



**INTELLIGENT SYSTEMS REFERENCE LIBRARY**  
**Volume 40**

Jeng-Shyang Pan  
Hsiang-Cheh Huang  
Lakhmi C. Jain  
Yao Zhao (Eds.)

# Recent Advances in Information Hiding and Applications

 Springer

## Editors-in-Chief

Prof. Janusz Kacprzyk  
Systems Research Institute  
Polish Academy of Sciences  
ul. Newelska 6  
01-447 Warsaw  
Poland  
E-mail: kacprzyk@ibspan.waw.pl

Prof. Lakhmi C. Jain  
School of Electrical and Information  
Engineering  
University of South Australia  
Adelaide  
South Australia SA 5095  
Australia  
E-mail: Lakhmi.jain@unisa.edu.au

Jeng-Shyang Pan, Hsiang-Cheh Huang,  
Lakhmi C. Jain, and Yao Zhao (Eds.)

# Recent Advances in Information Hiding and Applications

*Editors*

Prof. Jeng-Shyang Pan  
Department of Electronic Engineering  
National Kaohsiung University  
of Applied Sciences  
Kaohsiung  
Taiwan

Prof. Lakhmi C. Jain  
School of Electrical and Information  
Engineering  
University of South Australia  
Adelaide, South Australia  
Australia

Dr. Hsiang-Cheh Huang  
Department of Electrical Engineering  
National University of Kaohsiung  
Kaohsiung  
Taiwan

Prof. Yao Zhao  
Institute of Information Science  
Beijing Jiao Tong University  
Beijing  
P.R. China

ISSN 1868-4394

e-ISSN 1868-4408

ISBN 978-3-642-28579-0

e-ISBN 978-3-642-28580-6

DOI 10.1007/978-3-642-28580-6

Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012933769

© Springer-Verlag Berlin Heidelberg 2013

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media ([www.springer.com](http://www.springer.com))

---

## Preface

Information hiding techniques play a very important role in many areas including defence, business, copyright protection and many other areas where information needs to be protected. This book presents a sample of recent advances in information hiding techniques and their applications.

The book includes 10 chapters. Chapter 1 is on image data hiding scheme based on vector quantization and graph coloring. The scheme is capable to color each codeword in the codebook in different colors, each of which represents some bits of secret messages. The performance of the proposed scheme is validated to establish the superiority of the scheme.

Chapter 2 is on copyright protection system for Android platform. In the system, pre-specified copyright information is automatically embedded into pictures using digital watermark technology. The system is suitable for the protection of photographs taken by Android phones.

Chapter 3 is on reversible data hiding by coefficient adjustment algorithm. The authors have demonstrated the merit of the proposed algorithms using simulations. Chapter 4 is on Independent Component Analysis-Based Image and video watermarking. A number of ICA-Based image and video watermarking algorithms including case studies are presented.

Chapter 5 is on content based invariant image watermarking with high capacity. It is demonstrated experimentally that proposed scheme offers a better image quality and robustness. Chapter 6 is on single bitmap Block Truncation Coding (BTC) of color images using cat swarm optimization. It is demonstrated that the proposed scheme improves the decompressed BTC image quality.

Chapter 7 is on genetic-based wavelet packet watermarking for copyright protection. Chapter 8 is on lossless text steganography in compression coding. Chapter 9 presents a fast and low-distortion capacity adaptive synchronized acoustic-to-acoustic steganography scheme. The final chapter is on video watermarking with shot detection. Experimental results demonstrate the superiority of the scheme.

We are grateful to the authors and reviewers for their excellent contributions. The editorial support by Springer-Verlag is acknowledged.

Jeng-Shyang Pan, Taiwan  
Hsiang-Cheh Huang, Taiwan  
Lakhmi C. Jain, Australia  
Yao Zhao, China

## Related Books on Information Hiding

- Niu, X., Li, M., Suzuki, Y., Pan, J.S. and Jain, L.C. (Editors), *Intelligent Information Hiding and Multimedia Signal Processing, IIH-MSP 2011*, IEEE Press, USA, 2011.
- Echizen, I., Pan, J. S., Fellner, D., Nouak, A., Kuijper, A. and Jain, L.C. (Editors), *Intelligent Information Hiding and Multimedia Signal Processing, IIH-MSP 2010* IEEE Press, USA, 2010.
- Pan, J.S., Huang, H.C. and Jain, L.C. (Editors), *Information Hiding and Applications*, Springer-Verlag, Germany, 2009.
- Wang, F.H., Pan, J.S. and Jain, L.C., *Innovations in Digital Watermarking Techniques*, Springer-Verlag, Germany, 2009.
- Pan, J.S., Chen, Y.W. and Jain, L.C. (Editors), *Intelligent Information Hiding and Multimedia Signal Processing, IIH-MSP 2009*, IEEE Computer Society Press, USA, 2009.
- Pan, J.S., Niu, X.M., Huang, H.C. and Jain, L.C. (Editors), *Intelligent Information Hiding and Multimedia Signal Processing, IIH-MSP 2008*, IEEE Computer Society Press, USA, 2008.
- Pan, J-S. Huang, H-C, Jain, L.C. and Fang, W-C (Editors), *Intelligent Multimedia Data Hiding*, Springer-Verlag, 2007.
- Liao, B.-H., Pan, J.-S., Jain, L.C., Liao, M., Noda, H. and Ho, A.T.S. (Editors), *Intelligent Information Hiding and Multimedia Signal Processing, IIH-MSP 2007, Volume 1*, IEEE Computer Society Press, USA, 2007.
- Liao, B.-H., Pan, J.-S., Jain, L.C., Liao, M., Noda, H. and Ho, A.T.S. (Editors), *Intelligent Information Hiding and Multimedia Signal Processing, IIH-MSP 2007, Volume 2*, IEEE Computer Society Press, USA, 2007.
- Pan, J-S., Huang, H.C. and Jain, L.C. (Editors), *Intelligent Watermarking Techniques*, World Scientific Publishing Company Singapore, 2004.

---

# Table of Contents

|  |    |
|--|----|
| <b>1 An Image Data Hiding Scheme Based on Vector Quantization and Graph Coloring</b> |    |
| <i>Shuai Yue, Zhi-Hui Wang, and Chin-Chen Chang</i> .....                            | 1  |
| 1.1 Introduction .....   | 1  |
| 1.2 Related Works .....  | 2  |
| 1.2.1 Vector Quantization .....  | 2  |
| 1.2.2 Graph Coloring .....   | 3  |
| 1.2.3 Particle Swarm Optimization .....  | 4  |
| 1.3 Proposed Scheme .....  | 4  |
| 1.3.1 Embedding Phase .....  | 5  |
| 1.3.2 Extracting Phase .....   | 5  |
| 1.3.3 Embedding and Extracting Example .....   | 6  |
| 1.4 Experimental Results .....   | 8  |
| 1.5 Conclusions .....  | 16 |
| References .....   | 16 |
| <b>2 The Copyright Protection System for Android Platform</b>                        |    |
| <i>Yueh-Hong Chen and Hsiang-Cheh Huang</i> .....                                    | 19 |
| 2.1 Introduction .....   | 19 |
| 2.2 Related Works .....  | 20 |
| 2.3 Automatic Photograph Publishing System .....                                     | 21 |
| 2.3.1 Procedures for Automatic Photograph Publishing .....                           | 21 |
| 2.3.2 The Watermark Embedding Process .....  | 23 |
| 2.3.3 Optimal Method for Embedding Binary Value .....                                | 24 |
| 2.3.4 The Watermark Extraction Process .....   | 25 |
| 2.4 Experimental Results .....   | 26 |
| 2.5 Conclusion .....   | 31 |
| References .....   | 32 |



**3 Reversible Data Hiding By Coefficient Adjustment Algorithm**

*Ching-Yu Yang and Wu-Chih Hu* . . . . . 33

3.1 Introduction . . . . . 33

    3.1.1 Difference Expansion Techniques . . . . . 34

    3.1.2 Histogram-Based Schemes . . . . . 34

    3.1.3 Prediction-Based Methods . . . . . 35

    3.1.4 Interpolation-Based Algorithms . . . . . 36

    3.1.5 Robustness-Oriented Approaches . . . . . 36

    3.1.6 Human Visual System . . . . . 36

3.2 Coefficient Adjustment Algorithm . . . . . 37

    3.2.1 Bit Embedding . . . . . 37

    3.2.2 Bit Extraction . . . . . 38

    3.2.3 Overhead Information Analysis . . . . . 38

3.3 Data Hiding in the Spatial Domain . . . . . 38

    3.3.1 Use of a Codebook . . . . . 38

    3.3.2 Use of a Prediction Mean . . . . . 39

3.4 Data Hiding in the Frequency Domain . . . . . 41

3.5 Experimental Results . . . . . 42

    3.5.1 CA Algorithm with the Use of Codebook . . . . . 42

    3.5.2 CA Algorithm with the Use of Prediction Mean . . . . . 44

    3.5.3 Robust Version of the CA Algorithm . . . . . 45

    3.5.4 Performance Comparison and Discussions . . . . . 46

3.6 Conclusion . . . . . 50

References . . . . . 50

**4 ICA-Based Image and Video Watermarking**

*Jiande Sun and Ju Liu* . . . . . 53

4.1 Introduction of Independent Component Analysis . . . . . 53

4.2 Independent Feature of Image and Video . . . . . 55

    4.2.1 Independent Feature of Image . . . . . 55

    4.2.2 Independent Feature of Video . . . . . 60

4.3 ICA-Based Image Watermark . . . . . 64

    4.3.1 ICA-Based Watermark Detection and Extraction . . . . . 64

    4.3.2 Watermarking Based on Independent Image Block Feature . . . . . 66

    4.3.3 Watermarking Based on Independent Downsampling Feature . . . . . 69

4.4 ICA-Based Video Watermark Schemes . . . . . 73

    4.4.1 Watermarking Based on Independent Video Block Feature [35] . . . . . 73

    4.4.2 Watermarking Scheme Based on Independent Dynamic Feature [22] . . . . . 76

    4.4.3 Watermarking Based on Motion Location in Independent Dynamic Feature [54] . . . . . 86

    4.4.4 Watermarking Based on Independent Content Feature [60] . . . . . 93

References . . . . . 98

|   |     |
|---|-----|
| <b>5 Content Based Invariant Image Watermarking with High Capacity</b>                      |     |
| <i>Leida Li, Jianying Zhang, and Baolong Guo</i> .....                                      | 103 |
| 5.1 Introduction .....  | 103 |
| 5.2 Content Based Watermark Synchronization .....   | 105 |
| 5.3 High Capacity Watermark Embedding .....   | 107 |
| 5.3.1 Content Based Embedding .....   | 107 |
| 5.3.2 Partition Based Embedding .....   | 109 |
| 5.4 Simulation Results .....  | 111 |
| 5.4.1 Watermark Invisibility .....  | 111 |
| 5.4.2 Watermark Robustness .....  | 113 |
| 5.5 Conclusion .....  | 114 |
| References .....  | 116 |
| <b>6 Single Bitmap Block Truncation Coding of Color Images Using Cat Swarm Optimization</b> |     |
| <i>Shi-Yu Cui, Zhi-Hui Wang, Pei-Wei Tsai, Chin-Chen Chang, and Shuai Yue</i> ..            | 119 |
| 6.1 Introduction .....  | 119 |
| 6.2 Related Works .....   | 121 |
| 6.3 Proposed Scheme .....   | 122 |
| 6.4 Experimental Results .....  | 127 |
| 6.5 Conclusions .....   | 137 |
| References .....  | 137 |
| <b>7 Application of Genetic-Based Wavelet Packet Watermarking for Copyright Protection</b>  |     |
| <i>Hsiang-Cheh Huang and Yueh-Hong Chen</i> .....   | 139 |
| 7.1 Introduction .....  | 139 |
| 7.2 Genetic Watermarking in Wavelet Packet Transform .....                                  | 143 |
| 7.2.1 Best Basis Selection with GA .....  | 143 |
| 7.2.2 Adaptive Watermarking in Wavelet Packet Transform .....                               | 145 |
| 7.2.3 Limitations and Flexibilities with the Proposed Method .....                          | 146 |
| 7.3 Experimental Results .....  | 147 |
| 7.4 Conclusion .....  | 151 |
| References .....  | 152 |
| <b>8 Lossless Text Steganography in Compression Coding</b>                                  |     |
| <i>Chin-Feng Lee and Hsing-Ling Chen</i> .....  | 155 |
| 8.1 Background .....  | 155 |
| 8.2 Motivation .....  | 158 |
| 8.3 Related Work .....  | 160 |
| 8.3.1 Huffman Coding .....  | 160 |
| 8.3.2 The Scheme by Chang et al. ....   | 161 |
| 8.3.3 The Scheme by Shim et al. ....  | 164 |
| 8.4 Proposed Scheme .....   | 165 |
| 8.4.1 Embedding Procedure .....   | 165 |
| 8.4.2 Extraction and Recovery Procedure .....   | 169 |

|   |   |     |
|---|---|-----|
| 8.5   | Experimental Results and Discussion                                   | 171 |
| 8.6   | Conclusions   | 176 |
|   | References  | 176 |
| <b>9 A Fast and Low-Distortion Capacity Adaptive Synchronized Acoustic-to-Acoustic Steganography Scheme</b> |   |     |
|   | <i>Xuping Huang, Yoshihiko Abe, and Isao Echizen</i>                  | 181 |
| 9.1   | Introduction  | 181 |
| 9.2   | Problems with Conventional Methods                                    | 182 |
| 9.2.1   | Difference between Hiding Secret Messages in Images and Acoustic Data | 182 |
| 9.2.2   | Conventional Methods of Acoustic Steganography and Their Problems     | 183 |
| 9.2.3   | Contribution  | 184 |
| 9.2.4   | Example Application   | 185 |
| 9.3   | Synchronized Acoustic Steganography Using WAV Data                    | 186 |
| 9.3.1   | Statement of Purpose  | 186 |
| 9.3.2   | Acoustic Data Function  | 187 |
| 9.3.3   | Principle of Synchronized Steganography                               | 189 |
| 9.3.4   | Algorithms  | 190 |
| 9.4   | Experimental Evaluation   | 195 |
| 9.4.1   | Evaluated Data  | 195 |
| 9.4.2   | Evaluation Methods of Embedding Quality                               | 196 |
| 9.4.3   | Experiments and Results   | 197 |
| 9.4.4   | Evaluation of Real-Time Performance                                   | 201 |
| 9.5   | Conclusion  | 202 |
| 9.6   | Future Work   | 202 |
|   | Appendix  | 203 |
|   | References  | 208 |
| <b>10 Video Watermarking with Shot Detection</b>  |   |     |
|   | <i>Yueh-Hong Chen and Hsiang-Cheh Huang</i>                           | 211 |
| 10.1  | Introduction  | 211 |
| 10.2  | Embedding Algorithm   | 212 |
| 10.2.1  | Shot Detector   | 213 |
| 10.2.2  | Clustering  | 213 |
| 10.2.3  | DWT and DWT Coefficient Converter                                     | 213 |
| 10.2.4  | Watermark Embedder  | 215 |
| 10.3  | Extracting Algorithm  | 216 |
| 10.4  | Experimental Results  | 217 |
| 10.4.1  | Visual Quality  | 217 |
| 10.4.2  | Robustness  | 218 |
| 10.5  | Conclusion  | 226 |
|   | References  | 226 |
|   | <b>Subject Index</b>  | 229 |

---

## List of Contributors

### **Yoshihiko Abe**

Software Information Science Faculty,  
Iwate Prefectural University,  
152-52 Sugo, Takizawa,  
Iwate, Japan 020-0193  
yoshi@iwate-pu.ac.jp

### **Chin-Chen Chang**

Feng Chia University  
Taichung, Taiwan, R.O.C.  
ccc@cs.ccu.edu.tw  
[http://msn.iecs.fcu.edu.tw/  
~ccc/](http://msn.iecs.fcu.edu.tw/~ccc/)

### **Hsing-Ling Chen**

Graduate Institute of Informatics,  
Doctoral Program,  
Chaoyang University of Technology,  
Taichung County 41349,  
Taiwan, R.O.C.  
hsingling@cyut.edu.tw

### **Yueh-Hong Chen**

Far East University  
Tainan, Taiwan, R.O.C.  
yuehhong@feu.edu.tw

### **Shi-Yu Cui**

Dalian University of Technology  
Dalian, P.R. China  
cuishiyu0523@gmail.com

### **Isao Echizen**

National Institute of Informatics,  
2-1-2 Hitotsubashi, Chiyoda-ku,  
Tokyo, Japan 183-8512  
iechizen@nii.ac.jp

### **Bao-Long Guo**

Xidian University  
Xi'an 710071, P.R. China  
blguo@xidian.edu.cn

### **Wu-Chih Hu**

National Penghu University of Science  
and Technology  
Penghu, Taiwan, R.O.C.  
wchu@npu.edu.tw

### **Hsiang-Cheh Huang**

National University of Kaohsiung  
Kaohsiung, Taiwan, R.O.C.  
huang.hc@gmail.com  
[https://sites.google.com/site/  
hch888dr/](https://sites.google.com/site/hch888dr/)

**Xuping Huang**

School of Multidisciplinary Sciences,  
The Graduate University for Advanced  
Studies (SOKENDAI),  
2-1-2 Hitotsubashi, Chiyoda-ku,  
Tokyo, Japan 183-8512  
huang-xp@nii.ac.jp

**Lakhmi C. Jain**

University of South Australia  
Adelaide, SA, Australia  
Lakhmi.Jain@unisa.edu.au  
<http://www.kes.unisa.edu.au/>

**Chin-Feng Lee**

Department of  
Information Management,  
Chaoyang University of Technology,  
Taichung County 41349,  
Taiwan, R.O.C.,  
lcf@cyut.edu.tw

**Lei-Da Li**

China University of Mining and  
Technology  
Xuzhou, P.R. China  
reader1104@hotmail.com

**Ju Liu**

Shandong University  
Jinan, P.R. China  
jd\_sun@sdu.edu.cn

**Jeng-Shyang Pan**

National Kaohsiung University of  
Applied Sciences  
Kaohsiung, Taiwan, R.O.C.  
jspan@cc.kuas.edu.tw  
<http://bit.kuas.edu.tw/~jspan/>

**Jiande Sun**

Shandong University  
Jinan, P.R. China  
jd\_sun@sdu.edu.cn

**Pei-Wei Tsai**

National Kaohsiung University of  
Applied Sciences  
Kaohsiung, Taiwan, R.O.C.  
pwtsai@bit.kuas.edu.tw

**Zhi-Hui Wang**

Dalian University of Technology  
Dalian, P.R. China  
wangzhihui1017@gmail.com

**Ching-Yu Yang**

National Penghu University of Science  
and Technology  
Penghu, Taiwan, R.O.C.  
chingyu@npu.edu.tw

**Shuai Yue**

Dalian University of Technology  
Dalian, P.R. China  
peaceful1207@gmail.com

**Jianying Zhang**

China University of Mining and  
Technology  
Xuzhou, P.R. China  
zjycumt@126.com

**Yao Zhao**

Beijing Jiao Tong University  
Beijing, P.R. China  
yzhao@bjtu.edu.cn


# An Image Data Hiding Scheme Based on Vector Quantization and Graph Coloring

Shuai Yue<sup>1</sup>, Zhi-Hui Wang<sup>2</sup>, and Chin-Chen Chang<sup>3</sup>

<sup>1</sup> Department of Software,  
Dalian University of Technology,  
DaLian, China,  
peaceful1207@gmail.com

<sup>2</sup> Department of Software,  
Dalian University of Technology,  
DaLian, China,  
wangzhihui1017@gmail.com

<sup>3</sup> Department of Information Engineering and Computer Science,  
Feng Chia University,  
Taichung, Taiwan  
alan3c@gmail.com

**Summary.** Vector quantization, i.e., VQ, is an important image compression method. In the past, many data hiding schemes have been proposed based on VQ. However, none of them is graph coloring based, although graph coloring has been used in many applications. In this work, we propose a VQ-based data hiding scheme using graph coloring. The proposed scheme colored every codeword in the codebook in different colors, each of which represents some bits of secret messages. Then the scheme hidden data into the compression code through replacing one codeword in a color with another codeword in another color. The performance of the proposed scheme was evaluated on the basis of embedding capacity and imperceptibility, which proved the scheme's validity. 

## 1.1 Introduction

At the present time, computer science is making great advancements with the speed of computers and the bandwidth of the Internet seemingly increasing every day. This provides people more convenience in processing multi-media data, but, due to the great quantities of such data that are being transmitted, most of the image, sound, and video must be compressed before transmitting. Vector quantization is a popular image compression method that manipulates the fact that most blocks in an image are smooth and are similar to each other. VQ quantizes the image with a number of representative blocks, noted as codebook, so that the image can be represented

---

<sup>1</sup> Supported by the Fundamental Research Funds for the Central Universities.

as an index table of codewords in the codebook. Although VQ is simple, it can effectively reduce the size of the image while maintaining an acceptable quality of the compressed image.

Countless malicious attacks occur every day on the Internet, which is a public environment. Therefore, many techniques have been developed to protect data from illegal access. Data hiding in an image is one of those techniques that protect data by disguising the existence of the secret data by hiding data in the image with slight changes to the image on the condition that the modification cannot be perceived by the human eye [1, 2]. Data hiding in a VQ compressed image is more difficult than normal data hiding in the original image, since the size of the file that contains the VQ-compressed image is far smaller than the original image, and changes to the index table are usually more obvious.

In the past, many literatures related to VQ-based image data hiding have been published. Lin et al. [3] proposed a scheme in which the codebook,  $CB$ , is partitioned into two homogeneous parts, i.e.,  $CB_1$  and  $CB_2$ , where  $CB_1$  and  $CB_2$  are similar to each other. When '0' is to be embedded, the scheme finds the most similar codeword in  $CB_1$ , and when '1' is to be embedded, the scheme finds the most similar codeword in  $CB_2$ . Lin et al.'s scheme is similar to the least-significant-bit(LSB) substitution method. In 2005, Wu et al. [4] proposed a scheme that improved Lin et al.'s scheme by providing a codeword editing procedure to make  $CB_1$  and  $CB_2$  more similar to each other. However, both Lin et al.'s and Wu et al.'s methods can provide only limited embedding capacity. Later, Li et al. [5] proposed a scheme that divides the codebook into clusters according to the similarity between codewords. If the size of the cluster equals  $2^k$ , then  $k$  bits of secret message can be hidden through the replacement of the codewords in the cluster. However, the embedding capacity is still limited because Li et al.'s scheme requires the size of the cluster to be equal to  $2^k$ , which limited the number of clusters that can be used to embed data.

In the remainder of this work, we propose a scheme based on VQ and graph coloring. In Section 1.2 some works related to the proposed scheme are introduced. In Section 1.3 the proposed scheme is described. In Section 1.4 the experiment results are presented to show the validity of the proposed scheme. We conclude this chapter in Sec. 1.5.

## 1.2 Related Works

The proposed scheme makes use of vector quantization, graph coloring and particle swarm optimization, and these processes are described in this section.

### 1.2.1 Vector Quantization

Vector Quantization is a lossy data compression method that is used often [6]. It starts with the construction of a codebook that consists of codewords, each of which indicates a block of pixels. To compress an image, VQ scans the image in a Zig-zag manner, processing one block at a time with the block usually being  $4 \times 4$  pixels

in size. Figure 1.1 shows the encoding and the decoding procedure of VQ. When VQ process one block, it finds the most similar codeword in the codebook, which is determined by Euclidian distance, as shown in Equation (1.1):

$$D(\text{block}, \text{cw}) = \sqrt{\sum_{i=0}^{15} (\text{block}[i] - \text{cw}[i])^2}, \quad (1.1)$$

where block indicates the input block, and cw indicates the codeword in the codebook; block[i] and cw[i] indicate the  $i^{\text{th}}$  pixel value in the input block and in the codeword, respectively. Then, VQ outputs the index of the most similar codeword in the codebook to the index table as the compression code of the current input block. After processing every pixel block in the image, the compression code consists of two parts, i.e., the codebook and the index table. The decoder can recover the VQ indexed image by the codebook and the index table as Figure 1.1 shown.

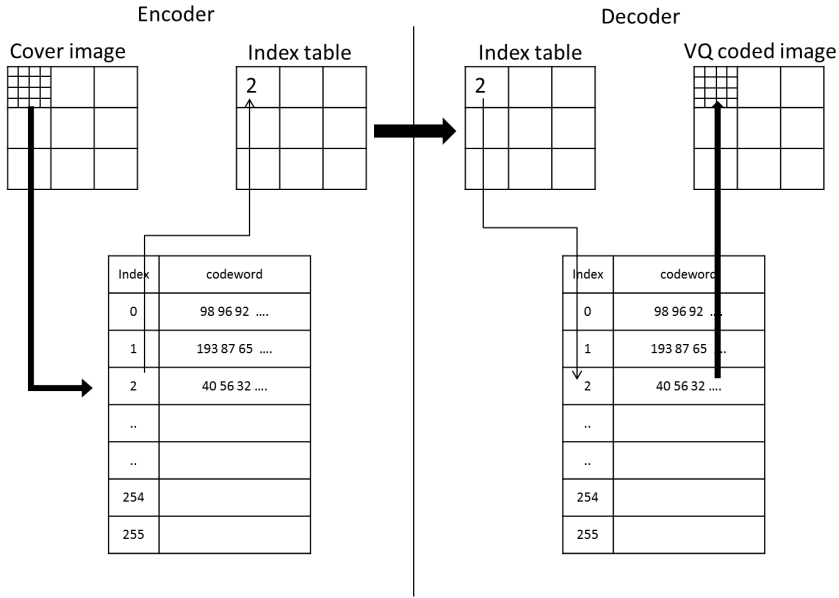


Fig. 1.1. The process of VQ compression.

### 1.2.2 Graph Coloring

Graph coloring includes vertex coloring, edge coloring, and face coloring [7]. In this work, graph coloring indicates vertex coloring, which assigns colors to the vertex of a graph so that none of the adjacent vertices has the same color. It is computationally difficult to color a graph. According to V. Guruswami et al.'s work [8], it is non-deterministic polynomial-time hard to color a three-colorable graph with four colors, meaning that it is impractical to color a graph that has a large number of vertices. So in this work, particle swarm optimization was selected to provide an approximate solution for a graph coloring problem [9].



### 1.2.3 Particle Swarm Optimization

Particle swarm optimization (PSO) is an evolutionary algorithm [10]. It maintains a lot of particles, each of which represent a candidate solution to the problem. A particle noted as  $p$  has two properties, i.e., position, noted as  $x$ , and velocity, noted as  $v$ . PSO optimizes the solution to a problem by improving every candidate solution iteratively, as shown in Equation (1.2a) and (1.2b):

$$v(t+1) = w \times v(t) + R(c) \times (p(t) - x(t)) + R(c) \times (g(t) - x(t)), \quad (1.2a)$$

$$x(t+1) = x(t) + v(t+1), \quad (1.2b)$$

where,  $x(t)$  and  $v(t)$  indicate the position and the velocity of the particle at the  $t^{\text{th}}$  loop, respectively;  $p(t)$  and  $g(t)$  are the previous best position of the particle and the previous best position of all the particles in the  $t^{\text{th}}$  loop, respectively;  $w$  and  $c$  are constant coefficients, and  $R(c)$  is the uniform distribution on  $[0, c]$ . At the beginning of the algorithm, PSO initiates all the informants randomly through a pseudo-random number generator, so that the results of different runs of PSO are different. So, in the implementation, a fixed number is selected to initiate the pseudo-random number generator. Due to the fact that the coloring result cannot be retrieved without this number, the fixed number can be treated as a key to protect the secret message, noted as *KEY*.

## 1.3 Proposed Scheme

In this section, the details of the proposed scheme are described.

Let  $I$  represent the cover image to be embedded in a binary message. The data hiding scheme begins after the construction of the codebook, noted as  $CB$ , by constructing a graph  $G$ . To construct  $G$ , the scheme adds a vertex,  $v$ , into graph  $G$  for every codeword in  $CB$ , so that every codeword has a corresponding vertex in  $G$ , noted as  $v(CW)$ . After adding every vertex into  $G$ , the scheme adds an edge into  $G$  for any two vertices  $v_1, v_2$  such that the distance between their corresponding codewords,  $D(cw_1, cw_2) \leq adj\_thresh$ , where  $adj\_thresh$  is a constant coefficient.

In the hiding scheme, every color can be denoted as an integer, so all the  $k$  colors, where  $k$  is the power of 2, can be represented as  $C = \{0, 1, \dots, k-1\}$ , so that every color corresponds to  $\log_2 k$  bits of secret message. For example, color 3 indicates '011' when  $k$  equals 8. After  $G$  is constructed, the scheme colors graph  $G$  with  $k$  different colors using the PSO algorithm, ensuring that any two adjacent vertices have different colors. The result of the coloring graph  $G$  is noted as  $Color(v)$ .

Next, graph  $G$  must be refined. The scheme checks every vertex,  $v$  in graph  $G$ , by iteratively testing whether a color can be found near  $v$  using a breadth-first search in a certain distance noted as  $bfs\_thresh$  from  $v$ . Any vertex  $v$  that has colors that cannot be found around it will be deleted from  $G$ . This procedure is repeated until no vertex is deleted.

### 1.3.1 Embedding Phase

The detailed embedding procedure is illustrated in pseudo-code in this subsection.

**Input:** a cover image  $I$  and a secret message  $S_d$

**Output:** codebook  $CB$  and a VQ compression code

**Step 1.** Generate codebook  $CB$  according to cover image  $I$  using *cell division*

**Step 2.** Construct a graph  $G$  with the following steps:

For every codeword  $CW$  in codebook  $CB$ ,

add a vertex  $v(CW)$  to graph  $G$ .

For any two vertices,  $v_1$  and  $v_2$ , in graph  $G$ ,

add an edge between them if their corresponding codewords  $cw_1, cw_2$  satisfying that  $D(cw_1, cw_2) \leq adj\_thresh$ .

**Step 3.** Color graph  $G$  with the PSO algorithm using a secret key  $KEY$  with  $k$  colors

**Step 4.** Refine graph  $G$

For every vertex  $v$  in the graph  $G$ :

use a breadth-first search to determine whether, within a certain distance from the vertex  $v$ , there are vertices colored with all the  $k$  colors. If not, delete  $v$  from graph  $G$ .

Repeat this step until no vertex must be deleted.

**Step 5.** For a block  $CW_{input}$  in cover image  $I$ ,

find its most similar codeword  $CB_{most\_similar}$  word in  $CB$ .

If  $v(CB_{most\_similar})$  exists in  $G$

use a breadth-first search to find a vertex  $v'$  that meets the following requirements:

$$Color(v') = S_d,$$

$$D(v, v') \leq bfs\_thresh.$$

output the index of the corresponding codeword of  $v'$  as the compression code of the current input block.

go to process  $S_{d+1}$  with the next input block.

Else

output the index of the current codeword  $CW_{most\_similar}$  as the compression code of the current input block.

go to process  $S_d$  with the next input block.

### 1.3.2 Extracting Phase

The extracting phase is just the inverse procedure of the embedding phase. The scheme constructs graph  $G$  according to the received codebook  $CB$ . Then, the scheme uses PSO to color graph  $G$  with the same  $KEY$  to initiate the pseudo-random generator. The coloring result will be exactly same as the result in the embedding phase. When the scheme meets a VQ-compression code,  $index$ , the scheme checks whether there is vertex  $v$  in graph  $G$  that corresponds the  $index^{th}$  codeword in the codebook. If there is,  $Color(v)$  is just the information hidden in the VQ-compression code. The corresponding block of the decompressed image is just the

$index^h$  codeword in the codebook which is different from most of the other VQ-based data hiding schemes whose index table cannot be decompressed by the normal VQ-decompression procedure.

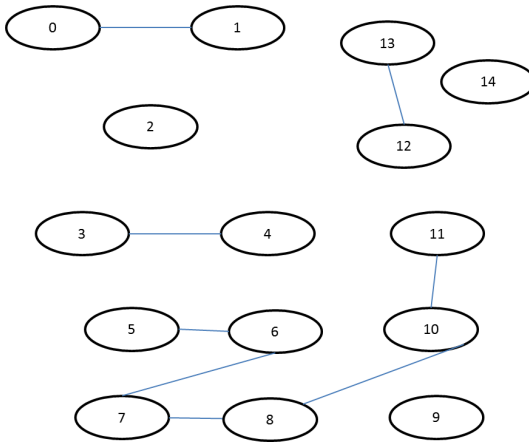
### 1.3.3 Embedding and Extracting Example

In this subsection, a detailed example of the proposed scheme is provided for illustration purposes.

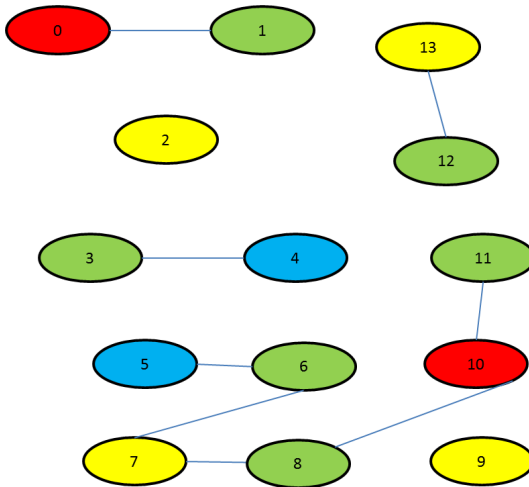
Assume that we are using a codebook as Table [1.1](#) shows. The codebook,  $CB$ , contains 14 codewords, which is far less than the usual size of the codebook, 256,

**Table 1.1.** Codebook of the embedding and extracting example.

| Index | Codeword   |
|-------|--|
| 0     | {183, 185, 185, 184, 184, 185, 185, 184, 184, 185, 185, 184, 184, 185, 185, 183} |
| 1     | {179, 181, 181, 180, 180, 181, 181, 180, 180, 181, 181, 180, 180, 181, 181, 179} |
| 2     | {59, 57, 57, 59, 57, 55, 55, 58, 57, 55, 54, 56, 60, 58, 58, 60}                 |
| 3     | {168, 169, 169, 168, 169, 170, 171, 169, 169, 171, 171, 169, 169, 170, 170, 167} |
| 4     | {164, 165, 165, 164, 165, 166, 167, 165, 165, 167, 167, 165, 165, 166, 166, 163} |
| 5     | {153, 155, 154, 153, 153, 155, 156, 154, 154, 156, 156, 154, 153, 154, 155, 153} |
| 6     | {149, 151, 150, 149, 149, 151, 152, 150, 150, 152, 152, 150, 149, 150, 151, 149} |
| 7     | {140, 141, 140, 138, 140, 141, 141, 140, 140, 142, 142, 141, 138, 140, 140, 139} |
| 8     | {136, 137, 136, 134, 136, 137, 137, 136, 136, 138, 138, 137, 134, 136, 136, 135} |
| 9     | {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}                                 |
| 10    | {126, 125, 125, 126, 127, 126, 126, 127, 128, 127, 127, 128, 127, 127, 127, 127} |
| 11    | {122, 121, 121, 122, 123, 122, 122, 123, 124, 123, 123, 124, 123, 123, 123, 123} |
| 12    | {102, 101, 102, 103, 101, 100, 100, 101, 102, 100, 99, 101, 102, 101, 101, 102}  |
| 13    | {98, 97, 98, 99, 97, 96, 96, 97, 98, 96, 95, 97, 98, 97, 97, 98}                 |



**Fig. 1.2.** Constructed graph  $G$ .



**Fig. 1.3.** Colored graph  $G$ .

but enough to illustrate our scheme. In the example, the  $adj\_thresh$  is set at 41 while the  $bfs\_thresh$  is set as 150. As in the first step, Graph  $G$  is constructed by adding a vertex for every codeword, so that every codeword has a corresponding vertex. The edges are then added to graph  $G$ . The proposed scheme iteratively tests whether the distance between two vertices is less than the  $adj\_thresh$ . If it is, the scheme adds an edge between the two vertices. For example,  $D(cw_0, cw_1)=16$  is less than  $adj\_thresh$ , 41, so there is an edge between them where  $cw_i$  indicates the codeword whose index is  $i$ . The constructed graph  $G$  is as Figure 1.2 shows.

Then graph  $G$  is then colored with  $k$  different colors, so that no adjacent two vertices have the same color. Here, due to the small size of  $CB$ , many methods can

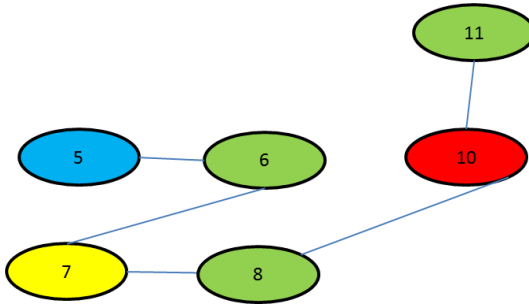


Fig. 1.4. Refined graph  $G$ .

be applied to resolve the graph coloring issue. However, the PSO can provide good security, so it is still adopted in this work. The PSO colored graph  $G$  is shown as Figure [1.3](#).

The next step in the process is to refine graph  $G$ . The proposed scheme deletes vertex,  $v$ , that there are colors that cannot be found using breadth-first search in a certain distance from  $v$ , such as vertex 0 and vertex 1. The result is as Figure [1.4](#).

After having been refined, graph  $G$  can be used to hide data. The proposed scheme scans the cover image in a Zig-zag manner, processing one block by one block. In the image ‘baboon’, the first block,  $cw_{input\_1}$ , is  $\{46, 46, 46, 63, 46, 46, 46, 63, 51, 51, 51, 51, 92, 92, 92, 53\}$ , due to  $D(cw_{input\_1}, cw_2)$  being the smallest one. The most similar codeword for  $cw_{input\_1}$  is  $cw_2$ , that has no corresponding vertex in refined graph  $G$ . Therefore, the scheme outputs 2 as the compression code. In the image ‘baboon’, the sixth block,  $cw_{input\_2}$  is  $\{115, 167, 169, 103, 115, 167, 169, 103, 96, 164, 197, 166, 71, 154, 196, 179\}$ , its most similar codeword is  $cw_6$ , which exists in graph  $G$ . Assuming the secret data is ‘00’ which is color 0, the scheme found a vertex around  $cw_6$  colored with color 0 which is vertex 10 in graph  $G$ . As a result, the scheme outputs 10 instead of 2 as the compression code that is embedded with secret data ‘00’.

In the extracting phase, the scheme constructs, colors and refines graph  $G$  exactly in the same manner as what the scheme has done in embedding phase. So in extracting phase, the decoder can achieve a same graph  $G$  as Figure [1.4](#). For compression code 2, if there is no vertex in graph  $G$  corresponding to codeword 2, there is no secret data inside the compression code; while for compression code 10, the scheme finds that vertex 10 exists, so  $color(v)$ , i.e. ‘00’ is the secret data and  $cw_{10}$  is the pixel block that should be output as the VQ-coded image.

## 1.4 Experimental Results

In this section, some experimental results are presented to prove the validity of the proposed scheme. The experiments were conducted on an Intel Core2 Duo P7450 computer at 2.13 GHz, and they were implemented in C using Intel Open CV 2.1.0 and Standard PSO in C [\[11\]](#). Figure [1.5](#) shows the cover images used in the

experiment in which Lena and Tiffany are smooth images; Baboon is a complicate image; the others are normal images. The cover images are all  $512 \times 512$  pixels in size.

In image steganography, embedding capacity indicates the number of secret bits that can be hidden in a cover image. Usually, peak signal-to-noise ratio (PSNR) is favored to evaluate the quality of the stego-images. The definition of PSNR is illustrated in Equation (1.3a).

$$\text{PSNR} = 10\log_{10} \left( \frac{255^2}{\text{MSE}} \right), \quad (1.3a)$$

$$\text{MSE} = \frac{1}{W \times H} \sum_{x=0}^{H-1} \sum_{y=0}^{W-1} (I(x,y) - I'(x,y))^2, \quad (1.3b)$$

where  $I$  and  $I'$  denote the cover image and the stego-image, respectively, both of which have the same size.  $I(x,y)$  denotes the pixel value of the cover image at the  $x^{\text{th}}$  row and  $y^{\text{th}}$  column.  $W$  and  $H$  denote the width and the height of the image, respectively. PSNR is widely used in the evaluation of image quality. A higher PSNR indicates better quality of the stego-image.

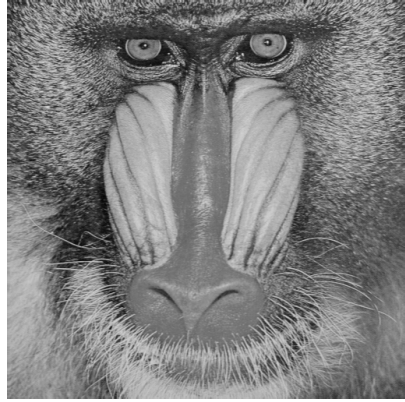
In the proposed scheme, 4 parameters, i.e.,  $k$ ,  $adj\_thresh$ ,  $bfs\_thresh$  and the size of the codebook, influence the experimental results. The symbol  $k$  controls the number of all colors used to color graph  $G$ ;  $adj\_thresh$  controls the distance between the vertices of graph  $G$ ;  $bfs\_thresh$  controls the radius of the breadth-first search, and ensures that the data hidden in the compression code is recoverable through the refine step; the size of the codebook influences the generation of codebook which is the base of graph  $G$ .

Figure 1.6 shows the influence of  $adj\_thresh$  on the proposed scheme. In this experiment, the size of codebook is set as 512. It can be observed that as the  $adj\_thresh$  increases, the embedding capacity also increases while the PSNR decreases either. However the increment of the embedding capacity is not stable, as such, sometimes the embedding capacity even decreases. This is a result of when  $adj\_thresh$  increases, more edges can be added to graph  $G$  leading to a more complicated graph  $G$  which is harder for the PSO to color.

Figure 1.7 illustrates the variation of the embedding capacity and the PSNR when  $bfs\_thresh$  varies. When  $k = 8$ , the variation is minimal. It is normal that the embedding capacity only minimally increases, while the PSNR also minimally decreases, because an increase in  $bfs\_thresh$  means that the radius of the breadth-first search is larger, so that the difference between the original index and the stego-index may be larger, which will finally lead to a decrease in PSNR. When  $k$  is set to 32, the variation is significant. In this case, when  $bfs\_thresh$  is in the range of 20 to 40, the embedding capacity is 0. However, when the  $bfs\_thresh$  is larger than 40, the embedding capacity increases stably. The PSNR also varies significantly, due to  $bfs\_thresh$  being less than 40. Because of this, there are not enough vertices around a vertex to be colored with 32 different colors, so that there is no data embedded and the PSNR is the same as the VQ-coded image's PSNR. This means that when  $k$  increases, the  $bfs\_thresh$  should also increase.



(a) Airplane



(b) Baboon



(c) Boat



(d) Pepper

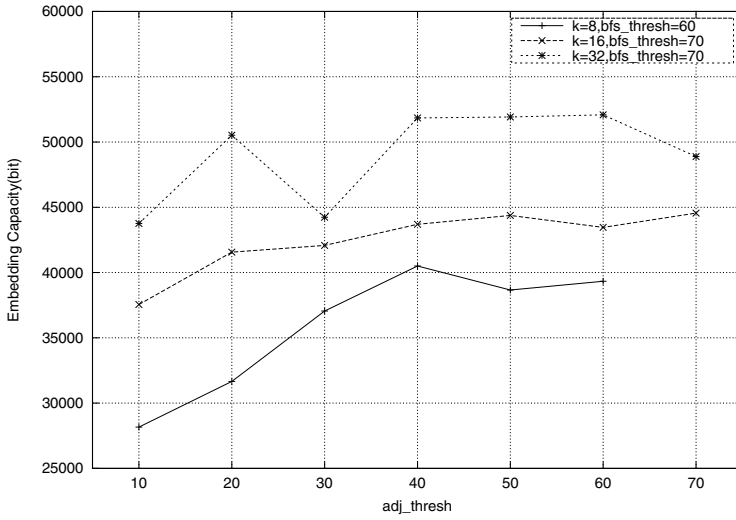


(e) Tiffany

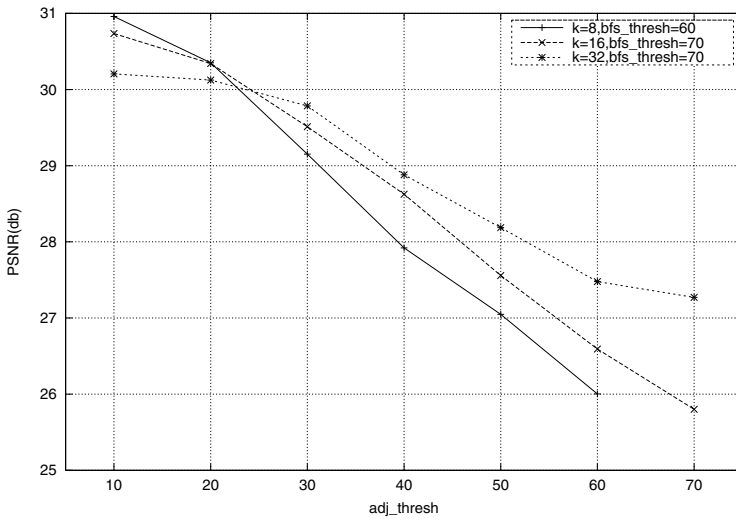


(f) Lena

**Fig. 1.5.** Images used to test the proposed scheme.



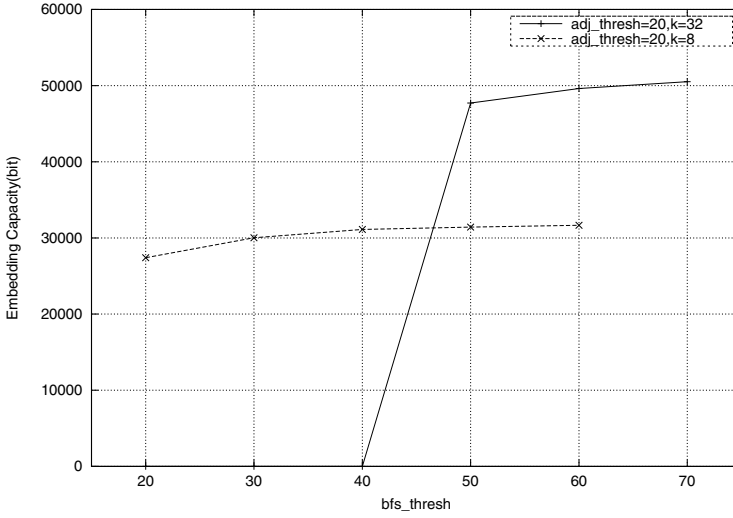
(a) Embedding capacity



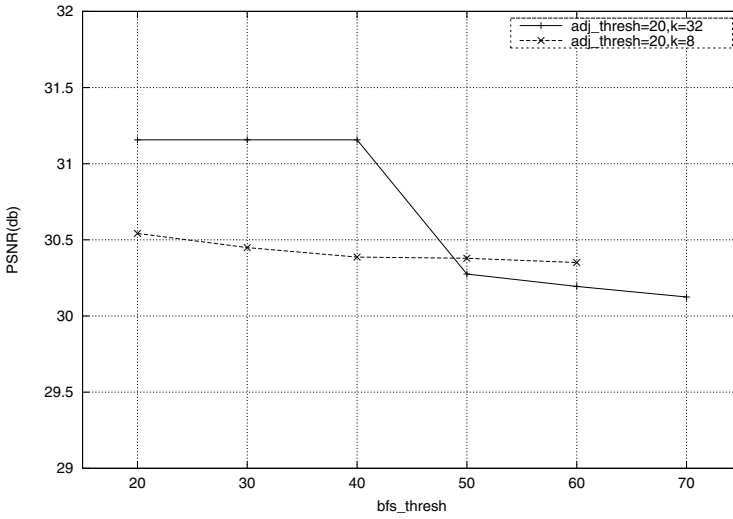
(b) PSNR

**Fig. 1.6.** Influence of  $adj\_thresh$  for image "Airplane".



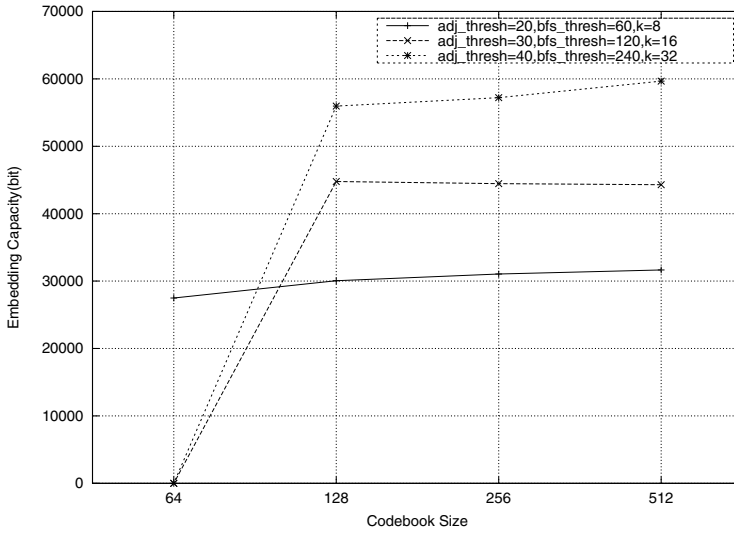


(a) Embedding capacity

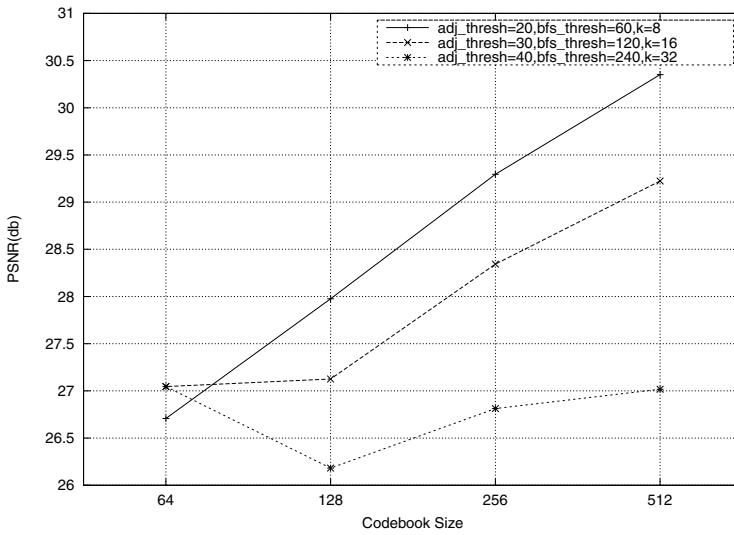


(b) PSNR

**Fig. 1.7.** Influence of  $bfs\_thresh$  for image "Airplane".

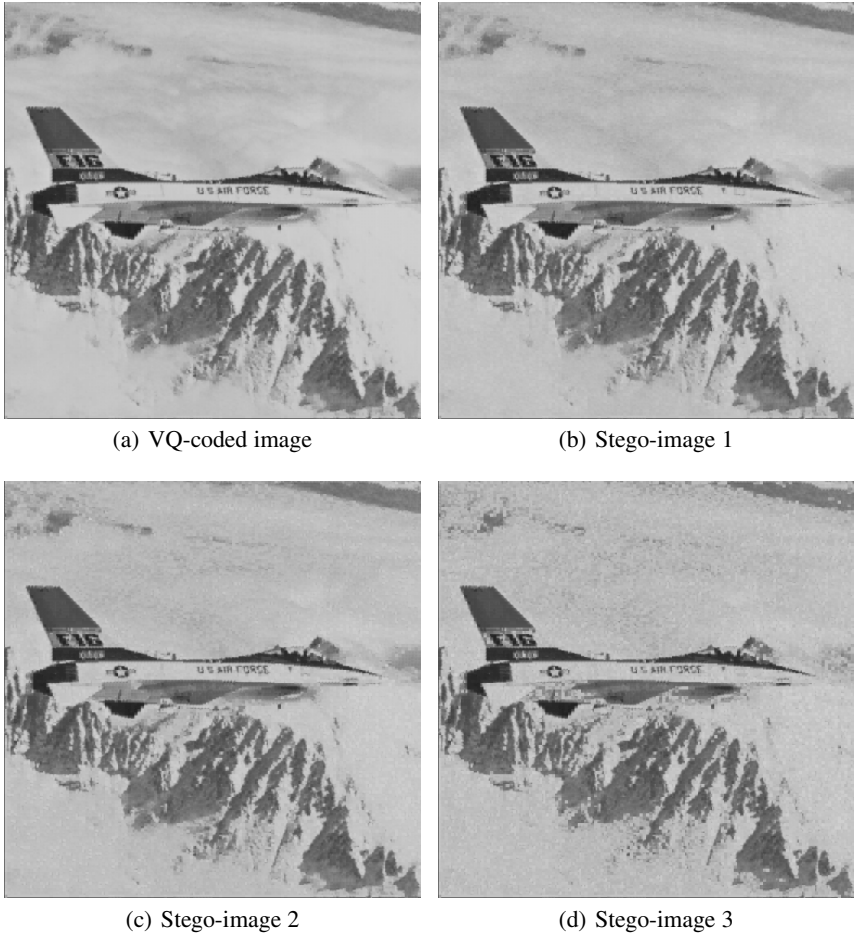


(a) Embedding capacity



(b) PSNR

**Fig. 1.8.** Influence of the size of codebook for image "Airplane".



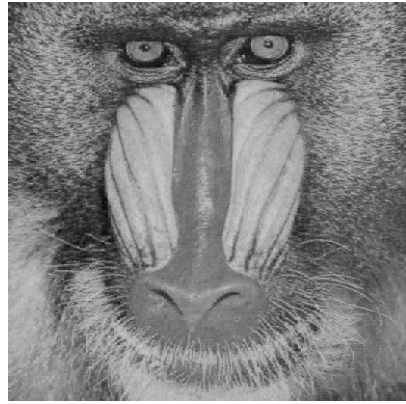
**Fig. 1.9.** Image “Airplane” and its stego-images.

The influence of the size of the codebook is shown in Figure 1.8. It can be observed that as the size of the codebook increases, the embedding capacity and the PSNR increases simultaneously. This result is different from the influence of  $bfs\_thresh$ . It occurs because the size of the codebook can improve the quality of the VQ-coded image and the improvement is usually larger than the negative influence caused by the embedding of the secret data. However, sometimes the embedding of secret data counterbalances the improvement brought about by the increase of codebook size which can be depicted in Figure 1.8 when  $adj\_thresh$ ,  $bfs\_thresh$ ,  $k$  and codebook size is 40, 240, 32, and 128, respectively.

In Figure 1.9, the resulting images of the proposed scheme with different parameters are presented. The sizes of the codebook of different images are all set as 512. Figure 1.9(a) shows the original VQ-coded image; stego-images 1, 2 and 3 are the



(a) Airplane



(b) Baboon



(c) Boat



(d) Pepper



(e) Tiffany



(f) Lena

**Fig. 1.10.** Stego-images.

results of the proposed scheme when  $adj\_thresh$  is 20, 30, 40;  $bfs\_thresh$  is 60, 120, 240;  $k$  is 8, 16, 32, respectively. Compared with the original VQ-coded image, the modification in stego-image 1 is totally imperceptible and the modification in stego-image 2 is still acceptable. However, the modification in stego-image 4 is too obvious. Figure 1.10 illustrates the stego-images of the proposed scheme with the same parameter as the stego-image 1 in Figure 1.9. It can be observed that our proposed scheme performs well on all the cover images. The embedding capacity and the PSNR of the images in Figure 1.10 are shown in Table 1.2.

**Table 1.2.** Performance of the proposed scheme.

| Picture | $adj\_thresh = 20,$<br>$bfs\_thresh = 60,$<br>$k = 8$ |           | $adj\_thresh = 30,$<br>$bfs\_thresh = 120,$<br>$k = 16$ |           | $adj\_thresh = 40,$<br>$bfs\_thresh = 240,$<br>$k = 32$ |           |
|---------|---|-----------|---|-----------|---|-----------|
|         | bits  | PSNR      | bits  | PSNR      | bits  | PSNR      |
|         | Airplane  | 31659     | 30.351415   | 44304     | 29.224346   | 59660     |
| Baboon  | 11772   | 24.373809 | 20332   | 23.95553  | 0   | 24.540745 |
| Boat    | 33492   | 28.447689 | 51116   | 27.212622 | 59660   | 25.451332 |
| Pepper  | 29829   | 29.387094 | 51744   | 27.496079 | 44845   | 25.822918 |
| Tiffany | 44499   | 31.341663 | 61480   | 29.638875 | 79550   | 27.532017 |
| Lena    | 39786   | 29.526763 | 46384   | 28.46273  | 78240   | 24.821300 |

## 1.5 Conclusions

In this work, we proposed a scheme based on VQ and graph coloring that can provide both good quality stego-images and good embedding capacity stego-images, which means our scheme is very flexible. The stego-index table that our proposed scheme outputs can be decompressed into a meaningful picture, while most of the other VQ-based data hiding schemes cannot accomplish this, which means our proposed scheme is valid and useful to steganography. Also, the proposed scheme manipulates PSO as a coloring method, which provides better security since the number to initiate the pseudo-random number generator is, in fact, a key. In addition, graph coloring was successfully applied in the image data hiding based on vector quantization. Therefore, our scheme provides flexibility, security, and innovation.

## References

1. Duric, Z., Jacobs, M., Jajodia, S.: Information hiding: Steganography and steganalysis. Handbook of Statistics 24, 171–187 (2005)
2. Tefas, A., Nikolaidis, N., Pitas, I.: Image watermarking: Techniques and applications. The Essential Guide to Image Processing 22, 597–648 (2009)

3. Lin, Y.C., Wang, C.C.: Digital images watermarking by vector quantization. *Optical Engineering* 3, 76–78 (1999)
4. Wu, H.C., Wu, N.I., Tsai, C.S., Hwang, M.S.: Image steganographic scheme based on pixel-value differencing and lsb replacement methods. *Vision, Image and Signal Processing* 152, 611–615 (2005)
5. Li, Y., Li, C.T.: Steganographic scheme for vq compressed images using progressive exponential clustering. In: *Proc. IEEE Int'l. Conf. Video and Signal Based Surveillance*, p. 85 (2006)
6. Cosman, P.C., Oehler, K.L., Riskin, E.A., Gray, R.M.: Using vector quantization for image processing. *Proceedings of the IEEE* 81, 1326–1341 (1993)
7. Yanez, J., Ramirez, J.: The robust coloring problem. *European Journal of Operational Research* 148, 546–558 (2003)
8. Guruswami, V., Khanna, S.: On the hardness of 4-coloring a 3-collorable graph. In: *Proc. 15th Annual IEEE Conf. Computational Complexity*, pp. 188–197 (2000)
9. Cui, G., Qi, L., Liu, S., Wang, Y., Zhang, X., Cao, X.: Modified PSO algorithm for solving planar graph coloring problem. *Progress in Natural Science* 18, 353–357 (2008)
10. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: *Proc. IEEE Int'l Conf. Neural Networks*, vol. 4, pp. 1942–1948 (1995)
11. Standard PSO (2007), <http://www.particleswarm.info/Programs.html>

---

# The Copyright Protection System for Android Platform

Yueh-Hong Chen<sup>1</sup> and Hsiang-Cheh Huang<sup>2</sup>

<sup>1</sup> Department of Computer Science and Information Engineering,  
Far East University,  
Tainan 744, Taiwan, R.O.C.  
yuehhong@gmail.com

<sup>2</sup> Department of Electrical Engineering,  
National University of Kaohsiung,  
Kaohsiung 811, Taiwan, R.O.C.  
huang.hc@gmail.com  
<https://sites.google.com/site/hch888dr/>

**Summary.** In recent years, Camera Smartphone has become a popular consumer electronics product. Young people like to use it to record their daily lives; moreover, they will share these photos and information to others. But the photos may be used without consent after they are uploaded to Internet. To avoid this problem, one can embed visible and invisible watermarks into images. However, an additional process for embedding watermarks should be performed before an image is uploaded. When the number of photos becomes large, the process for embedding watermarks will bother the user. Thus, in this chapter, we propose a copyright embedding system for Android platform. Using this system, pre-specified copyright information is automatically embedded into pictures with digital watermark technology when these pictures are taken. In addition, original images (i.e., images without watermarks) can be preserved selectively. Proposed system has following features: (1) computational complexity of watermark embedding process is possibly reduced for handheld mobile system; (2) the watermark can be extracted without the use of the original image; (3) the watermark embedded into an image would not be removed by commonly used image processing operations; (4) embedding copyright information, resizing the images, and uploading images to Internet are automatically performed without manual intervention. Therefore, this system is very suitable for the protection of the photographs taken by Android phones to prevent piratical behaviors.

## 2.1 Introduction

In recent years, the Camera Smartphone has become a popular consumer electronics product. Young people like to use that to record their daily lives; moreover, they will share these photos and information to others. However, the photos may be used without consent after they are uploaded to Blogs. Since digital images can easily be to redistributed without agreement, related researches on digital right protection

have attracted much attention in recent years. Among all schemes, digital watermarking [1]–[7] can still protect digital images when they are displayed. Thus, it can be used as one of the approaches to preventing image piracy. Digital watermarking is a process to embed some information (i.e., watermarks) to image data. The watermarked image can still be displayed, and the embedded information is used for owner identification in the future. In practice, the watermark embedding algorithm can be designed to resist re-encoding, compression, D/A converting and image processing operations. Therefore, digital watermarking has been considered as the underlying technology in several digital rights management (DRM) applications [2]. For instance, In copy prevention, digital watermarking may be used to embed license information so that hardware devices and software can detect illegal use of digital contents. In copyright protection applications, the watermark may be used to identify the copyright holder and ensure proper payment of royalties. Moreover, there are a number of other applications for which watermarking has been used or suggested. These include broadcast monitoring, transaction tracking, authentication, copy control, and device control [2].

Although one can embed visible and invisible watermarks into pictures to prevent image piracy, an additional process for embedding watermarks should be performed before a picture is uploaded while necessary. When the number of photos becomes large, the process for the embedding of watermarks will bother the user due to the routine process. Thus, in this chapter, we propose a copyright embedding system for Android platform. Using this system, pre-specified copyright information is embedded into pictures with digital watermark technology when these pictures are taken. In addition, original images (i.e., images without watermarks) can be preserved selectively. Since computational complexity of watermark embedding procedure is possibly reduced for handheld mobile system, processes including embedding copyright information and resizing the photos along with uploading these photos to Internet can be automatically performed. Thus, this system is very suitable for the protection of the photographs taken by Android phones to prevent piratical behaviors.

## 2.2 Related Works

Android is a software stack including an operating system, middleware and key applications. Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. Based on Linux kernel, Android provided a set of libraries, Android runtime and an application framework. For application developers, the Android SDK provides the tools and APIs necessary to develop applications on the Android platform using Java programming language.

Watermarking techniques can be briefly classified into *additive*, *multiplicative*, *quantization-based*, and *relationship-based* schemes. They are briefly described as follows.



1. In additive schemes, a very weak  $W$  is added into original signal  $x$ , as shown in Eq. (2.1):

$$Y = X + \alpha W, \quad (2.1)$$

where  $X$  is the original signal,  $Y$  is the watermarked signal and  $\alpha$  is a constant, referred to as *watermark strength*.

2. In multiplicative schemes, samples of the original data are multiplied by an independent signal  $(1 + \alpha W)$ . Precisely, multiplicative schemes can be described by Eq. (2.2):

$$Y = X \times (1 + \alpha W). \quad (2.2)$$

3. In quantization based watermarking schemes,  $X$  is modified such that the quantization indices imply a watermark with a certain quantization step  $q$ . For example, a binary watermark  $W$  can be embed into the signal  $X$  with following Eq. (2.3):

$$\begin{cases} \lfloor \frac{Y}{q} + 0.5 \rfloor \bmod 2 = 0, & \text{if } W = 0; \\ \lfloor \frac{Y}{q} + 0.5 \rfloor \bmod 2 = 1, & \text{if } W = 1. \end{cases} \quad (2.3)$$

In this example, signal  $X$  is modified into  $Y$  such that its quantization index is an even number to imply a binary value ‘0’, and vice versa.

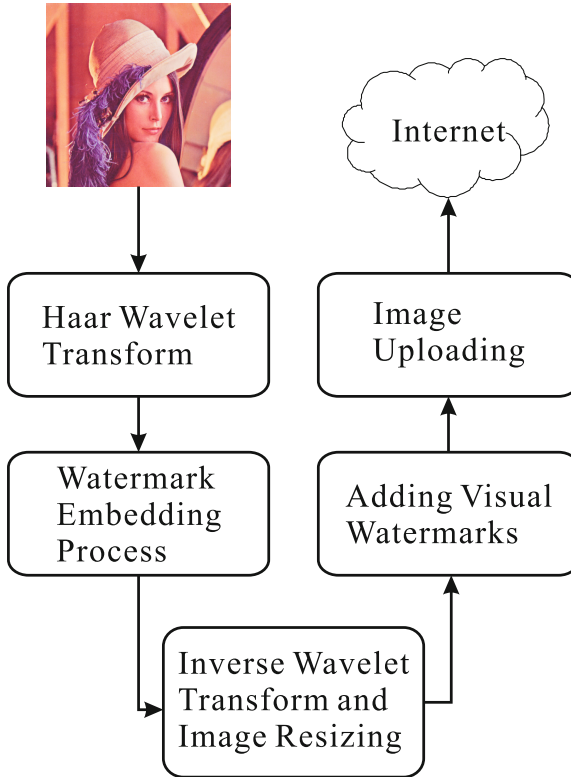
4. The basic idea of relationship-based watermarking is to use two pixel values or transform domain coefficients in an image to represent each bit of a binary watermark. If the first value is larger than the second, then an ‘1’ is encoded; otherwise, a ‘0’ is encoded. Hsu and Wu [8] proposed an approach using middle frequency coefficients chosen from one or more  $8 \times 8$  DCT blocks to embed watermarks. Quantization operation is taken into account in this approach so that watermarks can survive the JPEG lossy compression. In [9], six coefficients are selected from a DCT block and then the first coefficient is exchanged with the largest or smallest coefficient among them according to watermark bit value. A closely related approach was proposed in [10]. These approaches may also achieve good robustness.

## 2.3 Automatic Photograph Publishing System

In this section, we describe how to watermark a photograph with the proposed system, perform the resizing of images, and uploads the output image to Internet. The watermarking approach adopted in the system is also introduced briefly.

### 2.3.1 Procedures for Automatic Photograph Publishing

The automatic photograph publishing process proposed in this chapter involves three major tasks: (1) watermarking embedding, (2) image resizing, and (3) image uploading. These three tasks are integrated appropriately into the automatic photograph publishing process to reduce computational complexity. A flow diagram of the proposed automatic photograph publishing process is shown in Fig. 2.1. The



**Fig. 2.1.** Flow diagram of the proposed Automatic photograph publishing process

five steps included in the automatic photograph publishing process are described as follows.

1. Transform the photograph into Haar wavelet domain.
2. Embed the pre-specified watermark into the photograph.
3. Transform the photograph to pixel domain according a pre-specified resize factor.
4. Add a visible watermark (i.e., copyright information) to the photograph.
5. Upload the watermarked photograph to Internet.

Since Haar wavelet with some modifications may be performed without floating-point operations, it is very suitable for constrained devices such as handheld sets. Moreover, Haar wavelet transform is an orthonormal transform, so the visual quality of the watermarked image can be evaluated directly in the transform domain. In the second step, the watermarking scheme proposed in [11] is adopted to embed the watermark into photographs. This watermarking scheme will be introduced in follow subsections. After the watermark is embedded, the watermarked image may be resized so that it is applicable to the Internet environment. To resize the

watermarked image, we can use only the low and middle frequency coefficients in Haar wavelet domain to perform inverse Haar wavelet transform. Thus, if  $HH1$ ,  $HL1$  and  $LH1$  coefficients are discarded, the image is resized to half the original size. With the same manner, discarding  $HH1$ ,  $HL1$ ,  $LH1$ ,  $HH2$ ,  $HL2$  and  $LH2$  will result in a quarter-size watermarked image. Therefore, image resizing and watermark embedding can be applied simultaneously. If the resize factor specified by the user is not a power of 2, the resizing operation may not be performed by discarding some wavelet coefficients. In the circumstances, the method proposed in [12] is used to resize the image. Finally, we add a visible watermark with pixel domain image processing techniques and upload the watermarked image to a pre-specified web site.

### 2.3.2 The Watermark Embedding Process

To embed a watermark, an image is firstly transformed into Haar wavelet domain. For each bit of the watermark, a number of coefficients in pre-specified subband (e.g.,  $LH3$ ,  $HL3$  or  $HH3$ ) are then randomly chosen and modified. Finally, inverse wavelet transform is applied to obtain the watermarked image.

When one bit of the watermark is to be embedded, an user-specified number of coefficients are chosen randomly. These coefficients are then changed such that the first coefficient, in the order of being chosen, becomes the largest one if an '1' is to be embedded. If a '0' is to be embedded, the coefficients should be modified such that the first coefficient becomes the smallest one. Suppose  $c_i$ ,  $i = 1, \dots, n$ , are the chosen coefficients,  $n$  is the number of chosen coefficients,  $W = \{w_i \mid w_i \in \{1, 0\}, 1 \leq i \leq L\}$  is the watermark to be embedded, and  $L$  is the length of the watermark  $W$ . Precisely, after modification step, the relationship behind the coefficients is as Eq. (2.4)

$$\begin{cases} c'_1 \geq \max(c'_2, c'_3, \dots, c'_n) + \delta, & \text{if } w_i = 1, \\ c'_1 \leq \min(c'_2, c'_3, \dots, c'_n) + \delta, & \text{if } w_i = 0, \end{cases} \quad (2.4)$$

where  $c'_i$ ,  $i = 1, \dots, n$  are the modified coefficients,  $w_i$  is a particular bit of the watermark code, and  $\delta$ ,  $\delta \geq 0$ , is the strength parameter specifying the difference between the first coefficient and the largest (smallest) one among remaining coefficients. Intuitively, the larger the value of  $\delta$ , the more robust the watermark. However, the perceptual fidelity of the watermarked image will decrease when a larger  $\delta$  is adopted. For different applications, the value of  $\delta$  should be specified by the user.

To clarify the description, a simple example of implying a watermark bit with coefficients is given. Suppose an '1' is to be embedded and five coefficients,  $-5$ ,  $112$ ,  $-1$ ,  $107$  as well as  $13$ , are chosen randomly. A straightforward manner is to increase the value of the first coefficient,  $-5$ , to a value equal to or larger than  $112$ , and other coefficients are left unchanged. By this manner, the first coefficient,  $-5$ , should be increase to  $112 + \delta$  to embed an '1'.

### 2.3.3 Optimal Method for Embedding Binary Value

In order to obtain enhanced result in the image quality, more than one coefficient should be considered at the same time. To embed an '1', if the first coefficient  $c_1$  is increased to  $x + \delta$ , all coefficients larger than  $x$  should be decreased to  $x$  to fit the rule shown in Eq. (2.4). Therefore, it is possible to find the optimal value of  $x$  such that the watermarked image have the best quality according to an appropriate quality metric.

In this chapter, we adopt PSNR as the image quality metric due to its simplicity. If the mean square error (MSE) of the modified coefficients is minimized, the PSNR value is maximized simultaneously. Suppose a bit of '1' is to be embedded into  $n$  coefficients. If  $c_1$  is increased to  $x + \delta$  and all coefficients larger than  $x$  are decreased to  $x$ , the square error (SE) value can be calculated as Eq. (2.5):

$$SE(x) = ((x + \delta) - c_1)^2 + \sum_{c_i > x} (c_i - x)^2. \quad (2.5)$$

Then the minimum of  $SE(x)$  can be obtained by finding out the value of  $x$  where the first derivative of  $SE(x)$  is equal to 0. The first derivative of  $SE(x)$  is shown in Eq. (2.6), and the optimal value of  $x$  is shown in Eq. (2.7).

$$\frac{d}{dx}SE(x) = 2 \times (x + \delta - c_1) + 2 \times \sum_{c_i > x} (x - c_i), \quad (2.6)$$

$$x = \frac{\left( \sum_{c_i > x} c_i \right) + c_1 - \delta}{k + 1}, \quad i = 1, \dots, n, \quad (2.7)$$

where  $k$  is the number of coefficients larger than  $c_1$ . In Eq. (2.7), it is assumed that only  $k$  largest coefficients and  $c_1$  be modified. Therefore, the value of  $x$  should be larger than the  $(k + 1)$ -th largest coefficient but smaller than  $k$ -th largest coefficient. The algorithm to find the optimal value  $x$  is as follow:

Obtain  $d_1, d_2, \dots, d_n$  by sorting  $c_1, c_2, \dots, c_n$

such that  $d_1 \geq d_2 \geq \dots \geq d_n$

Suppose  $c_1$  is the  $(k + 1)$ -th largest value

If  $(k + 1) = 1$

$x_{opt} = d_2$ , Stop

End If

For  $i = 1$  to  $k$

$$x = \frac{(\sum_{j=1}^i d_j) + d_{k+1} - \delta}{i+1}$$

If  $d_{i+1} < x \leq d_i$

$x_{opt} = x$ , Stop

End If

End For

$x_{opt} = c_1$ , Stop

After completing the algorithm, the optimal value of  $x$  can be found.  $c_1$  can then be modified to  $x + \delta$ , and all coefficients larger than  $x$  be modified to  $x$  to embed a bit of '1'. A similar algorithm to find the optimal value to embed a '0' is as follow:

Obtain  $d_1, d_2, \dots, d_n$  by sorting  $c_1, c_2, \dots, c_n$   
 such that  $d_1 \leq d_2 \leq \dots \leq d_n$   
 Suppose  $c_1$  is the  $(k + 1)$ -th smallest value  
 If  $(k + 1) = 1$   
      $x_{opt} = d_2$ , Stop  
 For  $i = 1$  to  $k$   
      $x = \frac{(\sum_{j=1}^i d_j) + d_{k+1} + \delta}{i+1}$   
     If  $d_{i+1} > x \geq d_i$   
          $x_{opt} = x$ , Stop  
     End If  
 End For  
 $x_{opt} = c_1$ , Stop

Finally,  $c_1$  is decreased to  $x - \delta$ , and all coefficients smaller than  $x$  be increased to  $x$  to embed a bit of '0'.

Continuing the example in previous subsection, if  $\delta = 0$ , the  $SE(x)$  value is:

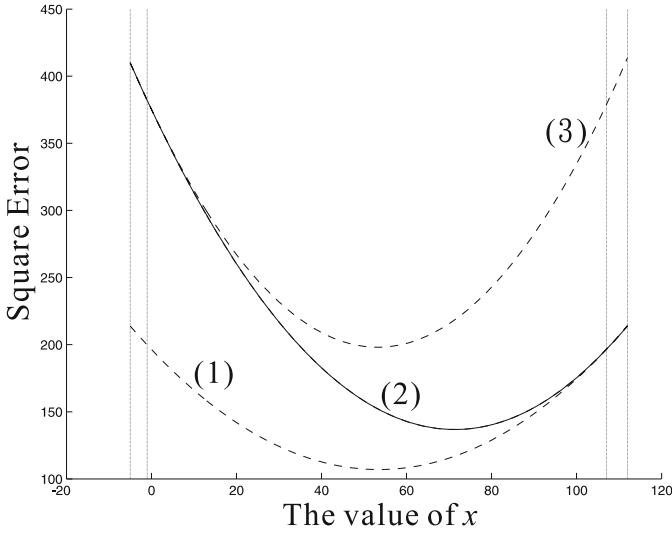
$$SE(x) = \begin{cases} (x - 112)^2 + (x + 5)^2 & \text{if } 107 \leq x < 112; \\ (x - 112)^2 + (x - 107)^2 + (x + 5)^2 & \text{if } -1 \leq x < 107; \\ (x - 112)^2 + (x - 107)^2 + (x + 1)^2 + (x + 5)^2 & \text{if } -5 \leq x < -1. \end{cases} \quad (2.8)$$

By applying the proposed algorithm, the optimal value of  $x$ , about 71.3, can be obtained. The curve of  $SE(x)$  is shown in Fig. 2.2. It is clear that  $SE(x)$  is the minimum when  $x = 71.3$ .

### 2.3.4 The Watermark Extraction Process

The simplest extracting method is to pick up the same coefficients and determine if the first coefficient is largest or smallest. However, the watermarked image may be distorted due to some image-processing operations, and the first coefficient is possibly no longer the largest or the smallest one. Hence, the purposed extracting method is to compare the first coefficient with the largest and smallest ones among remaining coefficients. If the value of the first coefficient is closer to the largest one among remaining coefficients, an '1' will be extracted; otherwise, a '0' will be extracted. This method can be described as Eq. (2.9):

$$w'_i = \begin{cases} 1, & \text{if } c_1'' \geq \frac{1}{2} (c_{\max}'' + c_{\min}'' ); \\ 0, & \text{otherwise.} \end{cases} \quad (2.9)$$



**Fig. 2.2.** Curves of  $(x-112)^2 + (x+5)^2$  (curve 1),  $(x-112)^2 + (x-107)^2 + (x+5)^2$  (curve 2) and  $(x-112)^2 + (x-107)^2 + (x+1)^2 + (x+5)^2$  (curve 3)

$$c_{\max}'' = \max(c_2'', c_3'', \dots, c_n'') \quad (2.10a)$$

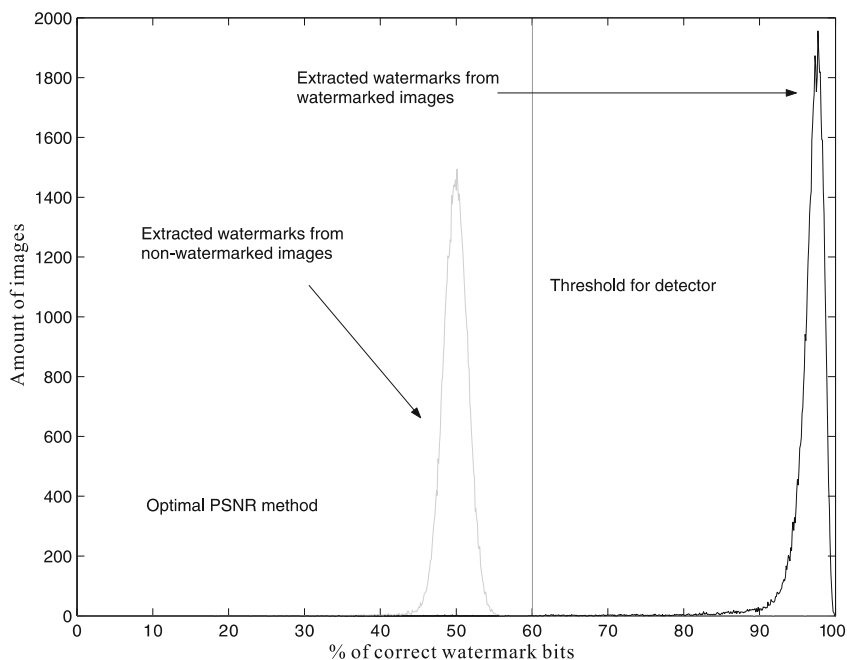
$$c_{\min}'' = \min(c_2'', c_3'', \dots, c_n'') \quad (2.10b)$$

where  $c_i'', i = 1, \dots, n$  are the coefficients obtained from an image to be judge, and  $w'_i, 1 \leq i \leq L$ , is the extracted binary value. A comparison between the extracted binary string  $W'$ ,  $W' = \{w'_i \mid w'_i \in \{1, 0\}, 1 \leq i \leq L\}$ , and the watermark  $W$  is then performed. Finally, the number of correct bit is compared with a certain threshold to determine if the watermark exists or not.

## 2.4 Experimental Results

In this section, some experimental results are presented to illustrate the practicality of our system. In the first part of the experiments, the robustness of the adopted watermarking algorithm was evaluated. An 1000-bit watermark was generated randomly and used in this experiment. To embed one bit of the watermark, twelve coefficients were chosen from *LH2*, *HL2* or *HH2* subband. In other words,  $n$  is equal to 12 in our experiments. The strength parameter  $\delta$  is assigned to 0.

To determine the threshold of the number of correct watermark bits, 58600 images chosen from Corel Gallery 1000000 were watermarked using the embedding algorithm described in Section 2.3.3. Then, the threshold was chosen such that watermarked and unwatermarked images could be well separated. The experimental result was shown in Fig. 2.3. According to the result, the value of threshold was assigned to 0.6 in all following experiments.



**Fig. 2.3.** Results of applying watermark detection process to 58600 watermarked and non-watermarked images

To evaluate the robustness of the proposed approach, six popular testing images: Lena, Baboon, F16, Fishing Boat, Pentagon, and Peppers are watermarked with the proposed watermarking approaches. Then, four image processing operations, JPEG compression, Gaussian filtering, sharpening, line removing and rescaling were applied on the watermarked images. The result are shown in Fig. 2.4–Fig. 2.7. As shown in Fig. 2.4, it is obvious that the watermark was still detectable until JPEG quality was lower than 15%. Similar results can be obtained after line removing, Gaussian filtering or sharpening as well as rescaling. These experimental results show that the adopted watermarking approach are robust on minimizing the perceptual distortion.

In the second part of experiments, an HTC desire HD smartphone with Android 2.3.3 was used as a test bed. A 512-bit watermark was used in the experiments, and every watermarked image was resized to quarter size of the original photograph. No visible watermark was added to the image before it was uploaded. In our experiment, six  $3264 \times 2448$  photographs were taken, watermarked, resized and uploaded to a pre-specific web site with the smartphone. In order to evaluate the image quality, the uploaded images were saved with lossless compression. The photographs uploaded to the web site are shown in Fig. 2.8. After JPEG compression was applied, the number and the percentage of correct bits of each watermark was presented in Table 2.1.

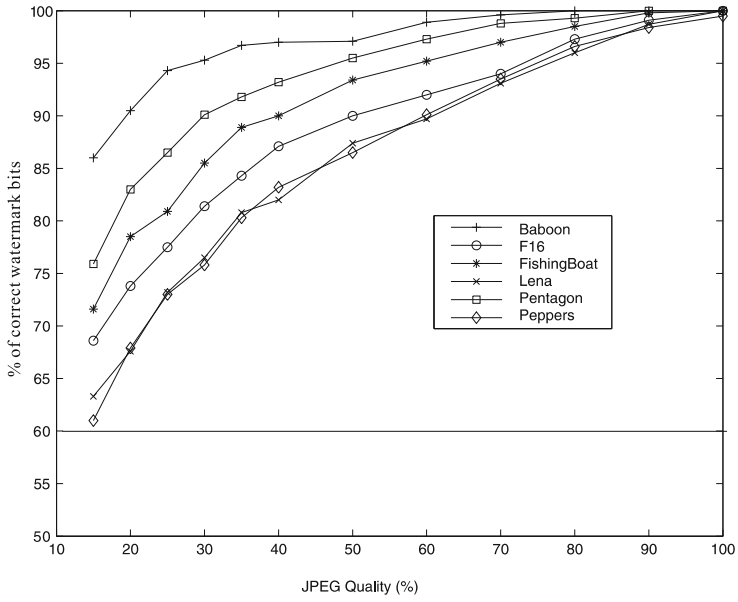


Fig. 2.4. Results of watermark detection after JPEG compression

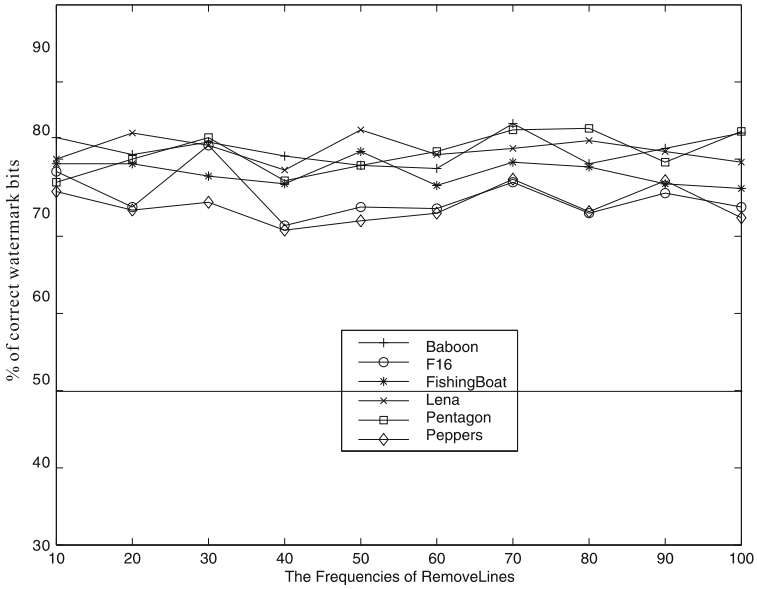


Fig. 2.5. Results of watermark detection after line removing attack



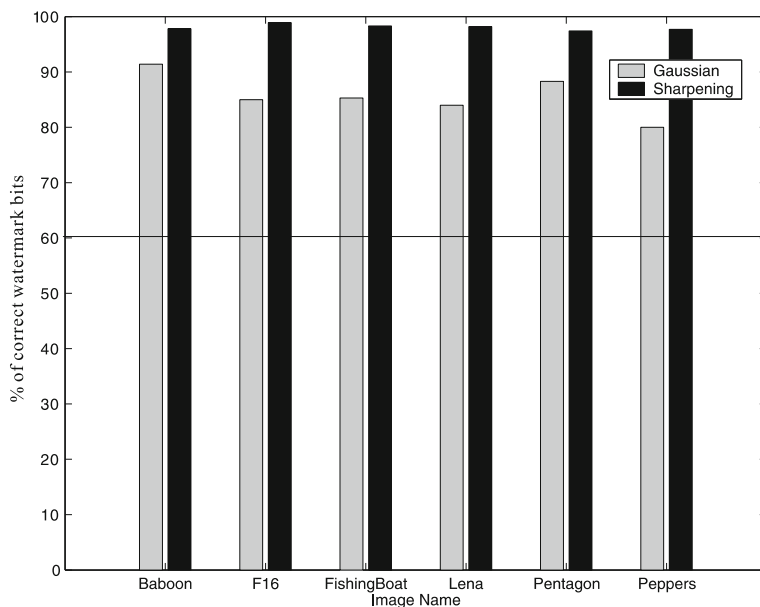


Fig. 2.6. Results of watermark detection after Gaussian filtering and sharpening

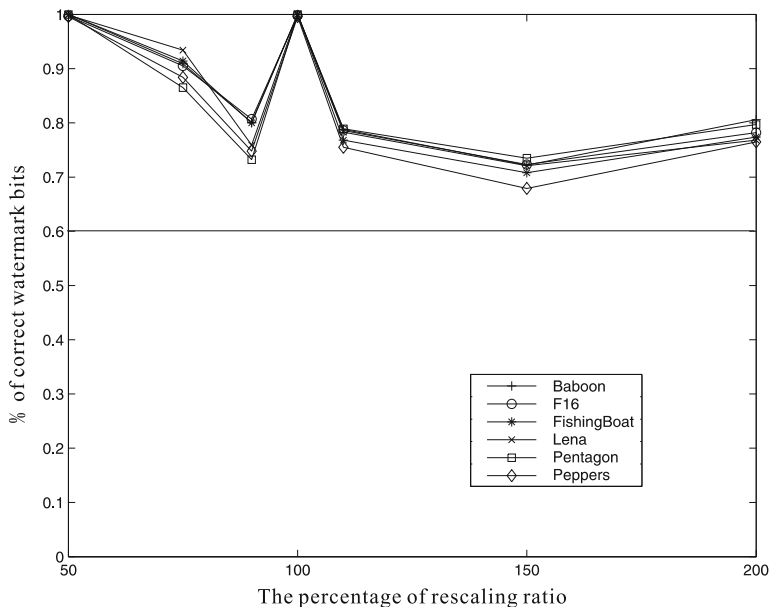


Fig. 2.7. Results of watermark detection after rescaling



(a)



(b)



(c)



(d)



(e)



(f)

**Fig. 2.8.** Images watermarked by the proposed system

**Table 2.1.** The number and the percentage of correct bits of each watermark embedded in the test image.

| Images used in<br>the experiment | Quality factor (Q) |       |      |       |
|----------------------------------|--------------------|-------|------|-------|
|                                  | Q=30               |       | Q=50 |       |
|                                  | #                  | %     | #    | %     |
| Fig. 2.8(a)                      | 423                | 82.6% | 473  | 92.4% |
| Fig. 2.8(b)                      | 427                | 83.4% | 475  | 92.8% |
| Fig. 2.8(c)                      | 415                | 81.1% | 466  | 91.0% |
| Fig. 2.8(d)                      | 413                | 80.7% | 465  | 90.8% |
| Fig. 2.8(e)                      | 437                | 85.4% | 486  | 94.9% |
| Fig. 2.8(f)                      | 434                | 84.8% | 481  | 93.9% |

As shown in Table 2.1, though JPEG compression was applied, about 85 percent of the watermark bits are still correct. If a larger quality factor was used, an image with better quality would be obtained, and the percentage of the correct watermark bits would increase to 95%. Since the image resizing was applied while inverse Haar wavelet transform was performed, these two operations can be completed in a few seconds. Thus, the developed system is a practical solution to protect the copyright of the photograph taken by Android smartphone.

## 2.5 Conclusion

In this chapter, we propose a copyright embedding system for Android platform. Using this system, pre-specified copyright information is automatically embedded into pictures with digital watermark technology when these pictures are taken. In addition, original images (i.e., images without modification) can be preserved selectively. This system has following features:

1. the watermarking approach based on Haar wavelet transform is adopted, and the watermark embedding process itself can also be performed without floating-point computation, so computational complexity of watermark embedding process is effectively reduced,
2. the watermark can be extracted without the use of the original image,
3. the watermark embedded into an image would not be removed by commonly used image processing operations, and
4. embedding copyright information, resizing the images, and uploading the images to Internet are automatically performed without manual intervention.

As shown in Section 2.4, this system is very suitable for the protection of the photographs taken by Android phones to prevent piratical behaviors. Therefore, our approach points out a practical application for smartphones.

## References

1. Cox, I.L., Miller, M.L., Bloom, J.A.: Digital Watermarking. Morgan Kaufmann, San Francisco (2001)
2. Cox, I.L., Miller, M.L., Bloom, J.A., Fridrich, J., Kalker, T.: Digital Watermarking and Steganography. Elsevier Science & Technology, London (2007)
3. Huang, H.C., Chen, Y.H., Abraham, A.: Optimized watermarking using swarm-based bacterial foraging. *Journal of Information Hiding and Multimedia Signal Processing* 1, 51–58 (2010)
4. Huang, H.C., Chen, Y.H.: Genetic fingerprinting for copyright protection of multicast media. *Soft Computing* 13, 383–391 (2009)
5. Huang, H.C., Fang, W.C.: Metadata-based image watermarking for copyright protection. *Simulation Modelling Practice and Theory* 18, 436–445 (2010)
6. Pan, J.S., Huang, H.C., Jain, L.C.: *Intelligent Watermarking Techniques*. World Scientific Publishing Company, Singapore (2004)
7. Pan, J.-S., Huang, H.-C., Jain, L.C. (eds.): *Information Hiding and Applications*. SCI, vol. 227. Springer, Heidelberg (2009)
8. Hsu, C.T., Wu, J.L.: Hidden digital watermarks in images. *IEEE Trans. on Image Processing* 8, 58–68 (1999)
9. Duan, F.Y., King, I., Chan, L.W., Xu, L.: Intra-block max-min algorithm for embedding robust digital watermark into images. In: *Proc. Multimedia Information Analysis and Retrieval*, pp. 255–264 (1998)
10. Bender, W., Gruhl, D., Morimoto, N., Lu, A.: Techniques for data hiding. *IBM Systems Journal* 35, 313–336 (1996)
11. Chen, Y.H., Su, J.M., Fu, H.C., Huang, H.C., Pao, H.T.: Adaptive watermarking using relationships between wavelet coefficients. In: *Proc. IEEE International Symposium on Circuits and Systems*, pp. 4979–4982 (2005)
12. Asamwar, R.S., Bhurchandi, K.M., Gandhi, A.S.: Interpolation of images using discrete wavelet transform to simulate image resizing as in human vision. *International Journal of Automation and Computing* 7, 9–16 (2010)

## Reversible Data Hiding by Coefficient Adjustment Algorithm

Ching-Yu Yang<sup>1</sup> and Wu-Chih Hu<sup>2</sup>

<sup>1</sup> National Penghu University of Science and Technology,  
Penghu, Taiwan  
chingyu@npu.edu.tw

<sup>2</sup> National Penghu University of Science and Technology,  
Penghu, Taiwan  
wchu@npu.edu.tw

**Summary.** This chapter presents three reversible data hiding schemes based on coefficient adjustment algorithm. The proposed coefficient adjustment (CA) algorithm mainly consists of three steps, which are block-mean removal, pixel squeezing, and pixel isolation. To provide a large hiding space, the mean removal firstly generates the differenced blocks from an input image. A pixel in the differenced blocks can be obtained by subtracting the original pixel value of the block from either the target codeword that was stored in a predetermined codebook or a prediction mean. Then, the pixel squeezing approach further provides hiding storage, while the pixel isolation approach minimizes distortion. Simulation demonstrated that the proposed method provided a larger payload than existing techniques at various embedding rates, and its corresponding PSNR was competitive. Moreover, a robust version of the proposed method can be achieved by employing the CA algorithm in the integer wavelet transform (IWT) domain. Experiments also confirmed that the resulting (marked) images were robust against manipulations of the image such as JPEG2000, JPEG, brightness adjustment, cropping, and inversion.

### 3.1 Introduction

Due to the steady development of network infrastructures and broadband services provided by the Internet Service Providers (ISPs), people can quickly and economically surf on the Internet. Several major services such as the world wide web (WWW), e-mail, real-time messaging, voice-over internet packet (VoIP), and video on demand (VOD) are widely used by individuals and organizations around the world. However, illegal eavesdropping or data tampering and falsification may occur during transmission. Encryption/decryption techniques such as secure socket layer (SSL), virtual private networks (VPNs), and digital signatures are often used to address these issues. The vulnerabilities of various operating systems (and applications), hosts (and servers) can be exploited by malicious users. When attackers compromised systems, confidential data, including credit card numbers, banking

accounts and passwords, and private documents, may not be adequately secure. Data hiding techniques [6, 20, 23] provide an alternative solution for protecting (or securing) data. Some data hiding schemes [5], [21, 22], [30], [32], [34] are irreversible, meaning that an original host medium cannot be completely recovered by the receiver after data have been extracted. Several authors [1]–[4], [7], [9]–[19], [24]–[28], [31], [33], [35]–[37] have suggested reversible data hiding to overcome this issue and preserve the originality of valuable (or priceless) media, including those associated with law enforcement, medical and military image systems, and geographic information. Six classifications of these methods are examined in the following subsections.

### 3.1.1 Difference Expansion Techniques

Tian [26] used a difference expansion (DE) technique to derive high capacity, low-distortion reversible data hiding. This technique divided the image into pairs of pixels, and a secret message was embedded into the difference between the pixels of each pair that was not expected to cause overflow/underflow. Simulation showed that the payload size and the perceived quality of the marked images generated by the technique were optimal results among existing research at that time. Alattar [1] used DE with vectors instead of pixel pairs to extend and improve the performance of Tian's algorithm. In a single pass, Alattar's algorithm could embed several bits into every vector. Applications of this algorithm could recursively occur across both grayscale and color images. Weng *et al.* [35] suggested an algorithm for lossless data hiding based on the invariability of the sum of pixel pairs and pair-wise difference adjustment (PDA). Higher peak signal-to-noise ratio (PSNR) can be achieved by making smaller modifications to pixel pairs. In addition, PDA was proposed to significantly reduce the overhead size. Hsiao *et al.* [9] suggested an algorithm to hide data bits in two areas: data embedding and auxiliary information areas. The former was further divided into three categories of the blocks. Namely, a message was hidden in both the smooth and normal blocks, whereas the complex blocks contained no data bit. The bitmap and other overhead were embedded in the auxiliary information area. To increase hiding capacity, Hu *et al.* [10] proposed a variant DE-based technique that improved the compressibility of the location map. Compared to conventional DE-based schemes, their technique provided increased embedding storage and performed well with a variety of image types.

### 3.1.2 Histogram-Based Schemes

Ni *et al.* [18] utilized the zero or minimum points of the histogram of an image and slightly modified the pixel values to embed data bits into the images without any loss. The PSNR generated by their method exceeded 48 dB. Based on the block difference histogram of a host image, Lin *et al.* [14] presented a high performance reversible data hiding. To achieve the goal, the algorithm selected a maximum point to embed data bits. Simulation indicated that an average embedding rate of 1.13 bpp can be achieved with a PSNR value around 30 dB. Lin and Hsueh [15] suggested a

lossless data hiding scheme that is based on the three-pixel block difference. In the host image, an absolute difference between a pair of pixels was used to embed data bits. The difference that was associated with the largest number of pixel pairs in the image was used. To embed a bit '1' or '0', the selected difference was increased by 1 or kept unchanged, respectively. An average hiding rate of 2.08 bpp (bit per pixel) can be achieved by the scheme with a peak signal-to-noise ratio (PSNR) value of 22.06 dB. Lin *et al.* [16] proposed a multilayer scheme for reversible data hiding based on modification of the difference histogram. By combining the peak point of a difference image concept with a multilevel hiding strategy, the scheme maintained high capacity while keeping distortion low. In first-level embedding, the low bound of PSNR reached 42.69 dB. Tai *et al.* [24] used histogram modification technique with a binary tree structure and presented a lossless data hiding approach. In addition, a histogram shifting technique was explored to prevent overflow/underflow. By using the difference between adjacent pixels rather than a single pixel, their algorithm operated at high capacity while maintaining low distortion. Based on modification of the difference histogram between sub-sampled images, Kim *et al.* [12] developed an efficient lossless data hiding algorithm. The algorithm shifted the difference histogram and then embedded data bits into the modified pixel values. The algorithm prevented any overflow/underflow issues and did not require overhead information during data extraction. Normally, the algorithm reached a payload size of 1.0 bpp. By using the interleaving maximum/minimum difference histogram with the shifting technique, Yang *et al.* [33] achieved a high-capacity reversible data hiding method. Owing to the interchangeable use of max-difference and min-difference, the distortion can be significantly reduced. Simulation indicated that the optimal embedding rate provided by the method was 1.120 bpp, with a PSNR value of approximately 30 dB.

### 3.1.3 Prediction-Based Methods

Based on prediction-error expansion and the histogram shifting technique, Thodi and Rodriguez [25] presented an effective and reversible method of data hiding. Simulation showed that prediction-error expansion doubled the maximum embedding capacity as compared to difference expansion. The perceived quality of the marked images was good at a moderate embedding capacity. Tsai *et al.* [27] presented a lossless data hiding approach using predicted coding and histogram shifting. The prediction technique first explored the similarities of neighboring pixels in the image, after which the residual histogram of the predicted errors of the host image hid the data bits. Compared to conventional histogram-based method, the resulting PSNR generated by the technique improved approximately 1.5 dB when embedding the same amount of data bits. Chen *et al.* [4] used the additive prediction-error expansion to embed a large amount data bits into images. Namely, the prediction-error was used to embed a bit '1' or '0' by expanding it additively or leaving it unchanged. Simulation showed that the scheme provided competitive performance compared with other existing techniques. Yang and Tsai [28] presented a reversible data hiding method by using interleaving prediction. All predictive values were

transformed into histogram to generate high peak values and improve hiding capacity. For a single pass embedding, the average PSNR of the marked images was larger than 48 dB. Lee *et al.* [13] developed an adaptive reversible data hiding approach based on the prediction of difference expansion. Simulation demonstrated that a high perceived quality of the marked image can be achieved by the approach. Moreover, the location map was not required during data extraction.

### 3.1.4 Interpolation-Based Algorithms

By using scaling-up neighbor mean interpolation, Jung and Yoo [11] proposed a new lossless data hiding method. The interpolation technique provided the advantages of low-time complexity and high-computing speed. Experiments indicated that their method can embed a large amount of bits into the host image while keeping distortion low. In addition, the resulting PSNR value was guaranteed to above 35 dB. Luo *et al.* [17] presented a novel reversible data hiding scheme using an interpolation technique, which can embed a secret message into images with imperceptible modification. Owing to the slight modification of pixels, high perceived of the resulting image was preserved. For single-layer embedding, the PSNR generated by the scheme was larger than 48.13 dB.

### 3.1.5 Robustness-Oriented Approaches

Ni *et al.* [19] developed a robust lossless data hiding technique based on the patchwork theory, the distribution features of pixel groups, error codes, and the permutation scheme. Although the payload size of the technique did not exceed 1,024 bits, the marked images contained no salt-and-pepper noise and the resulting PSNR exceeded 38 dB. Additionally, the marked images generated by the technique were robust against to JPEG/JPEG2000 compression. By shifting the mathematical difference values of a block, Zeng *et al.* [37] designed a lossless and robust data hiding method. The data bits were embedded into blocks by shifting mathematical difference values. Due to the separation of the bit-0-zone and the bit-1-zone, as well as the particularity of mathematical difference, the method was tolerant of non-malicious JPEG compression to some extent. Comparing the method proposed by Ni *et al.*, the obtained results showed that the method proposed by Zeng *et al.* increased embedding capacity but at sacrifice of bit error rate and perceived quality. On the other hand, Yang *et al.* [31] utilized the coefficient-bias scheme to propose the technique of combinational reversible data hiding in both spatial and frequency domains. The secret message (which can be divided into two parts) was embedded into the spatial domain and/or frequency domain of a host image. Although the approach has the capability of resisting manipulation, the pure payload size was not good enough.

### 3.1.6 Human Visual System

By considering the human visual system (HVS), Awrangjeb and Kankanhalli [23] presented a novel reversible data hiding algorithm. Because the message was



embedded into the host image with consideration of the HVS, the resulting marked images contained no perceptible artifacts. Simulation confirmed that the algorithm provided a higher payload size than other techniques at the time.

This chapter is organized as follows. In Sec. 3.2, we specify three reversible data hiding methods based on the CA algorithm. The first two versions of the proposed method were performed in the spatial domain in Sec. 3.3, whereas the third one of the proposed method was implemented in the frequency domain in Sec. 3.4. Both test results and performance comparisons were provided in Sec. 3.5. We conclude this chapter in Sec. 3.6.

## 3.2 Coefficient Adjustment Algorithm

The proposed coefficient adjustment (CA) algorithm mainly consists of three steps, which are block-mean removal, pixel squeezing, and pixel isolation. To provide a large hiding space, the mean removal firstly generates the differenced blocks from an input image. Then, the pixel squeezing approach further provides hiding storage, while the pixel isolation approach minimizes distortion. The details of the algorithm are specified in the following sections.

### 3.2.1 Bit Embedding

Let  $P = \{p_{ij}\}_{i=0}^{n \times n-1}$  be the  $j$ th non-overlapping block of size  $n \times n$  that divided from an input image. A (differenced) block can be obtained by  $\{\hat{p}_{ij}\}_{i=0}^{n \times n-1} = \{p_{ij}\}_{i=0}^{n \times n-1} - m_j$ , where  $m_j$  indicates either the prediction mean or the nearest neighbor of the the  $j$ th block-mean that can be found in a predetermined codebook. (The procedure of codebook generation will be described in Sec. 3.3.1). Then,  $\hat{p}_{ij}$  in a differenced block is shifted to a new (squeezed) value  $\tilde{p}_{ij}$  if it satisfies the following criteria:

$$\tilde{p}_{ij} = \begin{cases} \hat{p}_{ij} + \gamma, & \text{if } \hat{p}_{ij} < -\gamma \\ \hat{p}_{ij} - \gamma, & \text{if } \hat{p}_{ij} > \gamma \end{cases} \quad (3.1)$$

Finally, pixel isolation is carried out. Namely, a differenced (or squeezed) pixel  $\hat{p}_{ij}$ ,  $\hat{p}_{ij} \in \{\hat{p}_{ij}, \tilde{p}_{ij}\}$ , is adjusted to a new (isolated) value  $\check{p}_{ij}$  according to the following rules:

$$\check{p}_{ij} = \begin{cases} \hat{p}_{ij} - (2^k - 1)\beta, & \text{if } \hat{p}_{ij} \leq -\beta \\ \hat{p}_{ij} + (2^k - 1)\beta, & \text{if } \hat{p}_{ij} \geq \beta \end{cases} \quad (3.2)$$

Both  $\beta$  and  $\gamma$  are control parameters, and  $k$  is a positive integer. Notably, the isolated pixels carry no data bit. After these three steps, data bits are ready to be embedded into  $\tilde{p}_{ij}$  with  $-\beta < \hat{p}_{ij} < \beta$ , by modulo  $2^k$  substitution. A marked block is formed by adding  $m_j$  to each pixel in the differenced block. This procedure is repeated until all of the host blocks have been processed.

### 3.2.2 Bit Extraction

First, divide a marked image into a series of non-overlapping blocks that measure  $n \times n$ . Let  $Q = \{q_{ij}\}_{i=0}^{n \times n-1}$  be the  $j$ th hidden block of the marked image. Also let  $m_j$  be the prediction mean (or the codeword that was extracted from the codebook by a look-up table). The differenced pixels of the  $j$ th block are acquired using  $\{\hat{q}_{ij}\}_{i=0}^{n \times n-1} = \{q_{ij}\}_{i=0}^{n \times n-1} - m_j$ . Thereafter, data bits can be extracted from a differenced block. If  $-2^k\beta < \hat{q}_{ij} < 2^k\beta$ , then the data bits are obtained by applying modulo  $2^k$ . Subsequently, a differenced pixel that was originally less than  $-\beta$  is restored by adding  $\hat{q}_{ij}$  to  $(2^k - 1)\beta$  if  $\hat{q}_{ij} \leq -2^k\beta$ ; a differenced pixel that was originally larger than  $\beta$  is restored by subtracting  $\hat{q}_{ij}$  from  $(2^k - 1)\beta$  if  $\hat{q}_{ij} \geq 2^k\beta$ . Moreover, to restore a differenced pixel that contains no data bit,  $\hat{q}_{ij}$  is added to  $\gamma$  if  $\hat{q}_{ij} > 0$  and the corresponding flag in the bitmap is set to unity. Conversely,  $\hat{q}_{ij}$  is subtracted from  $\gamma$  if  $\hat{q}_{ij} < 0$  and the corresponding flag is set to 1. Finally, to restore the original host block, add all of the differenced pixels in the  $j$ th block to  $m_j$ . Repeat this procedure until all data bits have been extracted.

### 3.2.3 Overhead Information Analysis

A bitmap that indicates whether or not a differenced pixel has undergone the squeezing process is recorded during bit embedding. It is obvious that the overhead information used in the process of pixel squeezing is  $\lfloor \frac{M}{n} \rfloor \times \lfloor \frac{N}{n} \rfloor \times n^2 \leq MN$  bits, where the image size is  $M \times N$ . To help the decoder later extract the data bits, overhead information can be losslessly compressed [8] and sent by an out-of-band transmission to the receiver. From a data security point of view, the proposed algorithm provided a more secure manner than the methods which embedded a message in combination with the overhead information into a host medium.

To overcome the overflow/underflow issues, a similar pixel-shifting approach [13] can be performed in the spatial domain before embedding. Namely, if a pixel  $p$  in a host image satisfied either  $p < \phi_1$  or  $p > \phi_2$ ,  $p$  can be adjusted to a new value by adding to or subtracting from an integer offset  $\delta$ . Both  $\phi_1$  and  $\phi_2$  are two predetermined threshold values.

## 3.3 Data Hiding in the Spatial Domain

The proposed CA algorithm which embeds a secret message into a host medium in the spatial domain is specified in this section. First, the proposed CA algorithm embeds data bits into images with the use of a codebook was introduced. Then, the CA algorithm embeds data bits into images by using the prediction technique was described.

### 3.3.1 Use of a Codebook

As described previously, the mean removal of the CA algorithm firstly generated the differenced blocks from an input image. Namely, a pixel in the differenced blocks

can be obtained by subtracting a pixel value of the block from the codeword that was stored in a predetermined codebook. The codebook was generated by a hierarchical clustering algorithm using the radius weighted mean (RWM), in a two-class clustering technique [29].

The encoding part of the CA algorithm with the use of a codebook is virtually identical to the procedure of bit embedding specified in Sec. 3.2.1. Here,  $m_j$  represents the selected codeword. More specifically, the codeword is determined by finding the nearest neighbor of the block-mean that stored in a predetermined codebook. The decoding part of the CA algorithm can be similarly obtained from the bit extraction which listed in Sec. 3.2.2.

The numbers of overhead bits which including the bitmap and the index of the selected codeword for each block are (a)  $n^2 + 6$ , (b)  $n^2 + 5$ , (c)  $n^2 + 4$ , and (d)  $n^2 + 3$ , when the codebook is composed of 64, 32, 16, and 8 codewords, respectively. The size of a block is  $n \times n$ . Restated, the total number of bits associated with the overhead is (a)  $\frac{M}{n} \times \frac{N}{n} \times (n^2 + 6) = \left(1 + \frac{6}{n^2}\right) O_H$ , (b)  $\left(1 + \frac{5}{n^2}\right) O_H$ , (c)  $\left(1 + \frac{4}{n^2}\right) O_H$ , and (d)  $\left(1 + \frac{3}{n^2}\right) O_H$  respectively, when a codebook contains codewords with the above four sizes.  $O_H = MN$  denotes the size of an image.

### 3.3.2 Use of a Prediction Mean

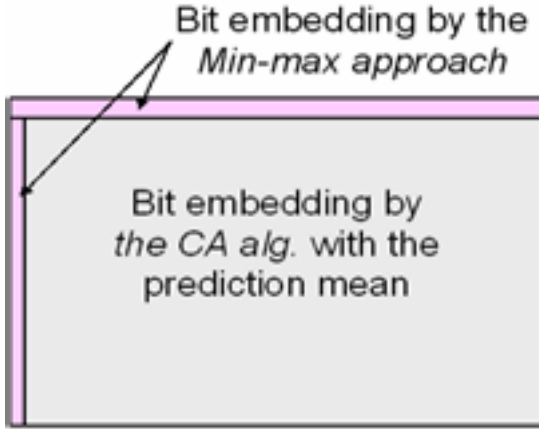
To reduce the overhead size, the CA algorithm with the use of the prediction mean was employed here. A small part of the secret message was first hidden in the first  $l$ -row and  $l$ -column of a host image by the min-max approach when host block size is  $l \times l$ . A predicted mean was then subtracted from the pixels in a block so as to generate a reduced block. In other words, most of the secret message is embedded into the reduced blocks by the CA algorithm. Fig. 3.1 gives an overview of the method.

#### Min-Max Approach

The concept of the min-max approach was to keep the minimum (or maximum) pixel of the (host) block unchanged. Without loss of generality, let  $p_{\min} = \arg \min \{p_j\}_{j=0}^{(l \times l)-1}$ , and let  $p_{\max} = \arg \max \{p_j\}_{j=0}^{(l \times l)-1}$  be the minimum and maximum pixel in a block  $P$  of size  $l \times l$ , respectively. Both  $\beta$  and  $\gamma$  are control parameters,  $k$  is a positive integer, and  $d$  is the input bit. The main steps of the min-max algorithm are summarized as follows:

**Step 1.** Input a non-overlapping block  $P$  which derived from the first  $l$ -row (or  $l$ -column) of a host image.

**Step 2.** Compute  $p_{\max}$  and  $p_{\min}$  of  $P$ , respectively.



**Fig. 3.1.** A schematic view of the proposed method uses the min-max approach and the CA algorithm with the prediction mean.

**Step 3.** If  $p_{\max} \geq 128$ , then subtract  $p_{\min}$  from  $p_j$ . Otherwise, subtract  $p_{\max}$  from  $p_j$  to obtain a reduced value  $q_j$ .

**Step 4.** If  $q_j > \gamma$ , then  $\tilde{q}_j = q_j - \gamma$ .

**Step 5.** If  $q_j \geq \beta$  (or  $\tilde{q}_j \geq \beta$ ), then compute  $q'_j = q_j + \beta$  (or  $q'_j = \tilde{q}_j + \beta$ ) (the pixels are unqualified for carrying data bits).

**Step 6.** If  $0 \leq q_j < \beta$  (or  $0 \leq \tilde{q}_j < \beta$ ), then evaluate  $\hat{q}_j = 2 \times q_j$  (or  $\hat{q}_j = 2 \times \tilde{q}_j$ ) and add  $\hat{q}_j$  to  $d$  (to form a hidden block).

**Step 7.** A marked block is introduced by adding (or subtracting)  $p_{\min}$  (or  $p_{\max}$ ) to (or from) a hidden block if  $p_{\max} \geq 128$  (or  $p_{\max} < 128$ ).

**Step 8.** Repeat Step 1 until all blocks are processed.

### Block-Mean Prediction

This section describes the three predictors [4] used in this approach, namely, the MEAN predictor, the median edge detector (MED) predictor, and a simplified version of the gradient-adjustment predictor (SGAP). Notably, the prediction is block-based rather than pixel-oriented in the proposed method.

The SGAP predictor of block  $C_j$  is determined by the following formula:

$$m_{SGAP} = \left\lfloor \frac{m_A + m_B}{2} \right\rfloor + \left\lfloor \frac{m_D + m_C}{4} \right\rfloor \quad (3.3)$$

where  $m_A$ ,  $m_B$ ,  $m_C$ , and  $m_D$  are the means computed from the respective four-neighbour blocks of  $C_j$ , as illustrated in Fig. 3.2

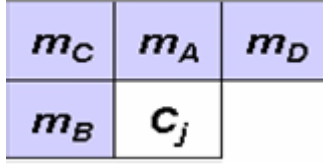


Fig. 3.2. A block  $C_j$  with its four neighboring blocks.

The MED predictor is

$$m_{MED} = \begin{cases} \min(m_A, m_B), & \text{if } m_C \geq \max(m_A, m_B), \\ \max(m_A, m_B), & \text{if } m_C \leq \min(m_A, m_B), \\ m_A + m_B - m_C, & \text{otherwise.} \end{cases} \quad (3.4)$$

The MEAN predictor is then defined by

$$m_{MEAN} = \left\lfloor \frac{m_A + m_B}{2} \right\rfloor \quad (3.5)$$

Simulation to test performance in hiding smooth images such as *Lena* and *Pep-pers* confirmed that the proposed method with SGAP predictor has better hiding performance. However, for hiding images with more texture (or edge) blocks than smooth blocks, MEAN predictor was superior. Although the proposed method with the MED predictor provided the best payload size among these three predictors at an average PSNR value above 47 dB, its performance degraded significantly as PSNR values approach 30 dB.

The encoding and decoding parts of the CA algorithm by using the prediction technique is identical to the procedures of bit embedding and bit extraction that described in Sec. 3.2.1 and Sec. 3.2.2, respectively, with the exception of  $m_j$ . Namely,  $m_j$  was replaced by a prediction mean. Notice that the overhead information for the coefficient adjustment algorithm with the use of prediction mean is  $\lfloor \frac{M-l}{T} \rfloor \times \lfloor \frac{N-l}{T} \rfloor \times l^2 < MN$  bits.

### 3.4 Data Hiding in the Frequency Domain

To achieve a method with a high hiding capacity and robust performance, we embed a message into the frequency domain based on the CA algorithm. Said more accurately, an input image was first decomposed to the integer wavelet transform (IWT) domain. The IWT coefficients can be consecutively acquired by using the following two formulas:

$$d_{1,k} = s_{0,2k+1} - s_{0,2k}, \quad (3.6)$$

and

$$s_{1,k} = s_{0,2k} + \left\lfloor \frac{d_{1,k}}{2} \right\rfloor, \quad (3.7)$$

where  $\lfloor x \rfloor$  is a floor function.

The encoding and decoding parts of the CA algorithm that performed in the IWT domain were similar to the procedures specified in Sec. 3.2.1 and Sec. 3.2.2. Note that only the last two steps of the CA algorithm, namely, pixel squeezing and pixel isolation were employed here during bit embedding. More specifically, data bits were embedded into the blocks which derived from the three high sub-bands, namely, the low-high (LH), high-low (HL), and high-high (HH) sub-bands of the IWT coefficients, respectively. The number of bits for the overhead information which used to signify whether or not a coefficient of the block undergone adjustment for the CA algorithm is  $\left\lfloor \frac{\frac{1}{2}M}{u} \right\rfloor \times \left\lfloor \frac{\frac{1}{2}N}{u} \right\rfloor \times u^2 \times 3 \leq \frac{3}{4}MN$ , where the block size is  $u \times u$ .

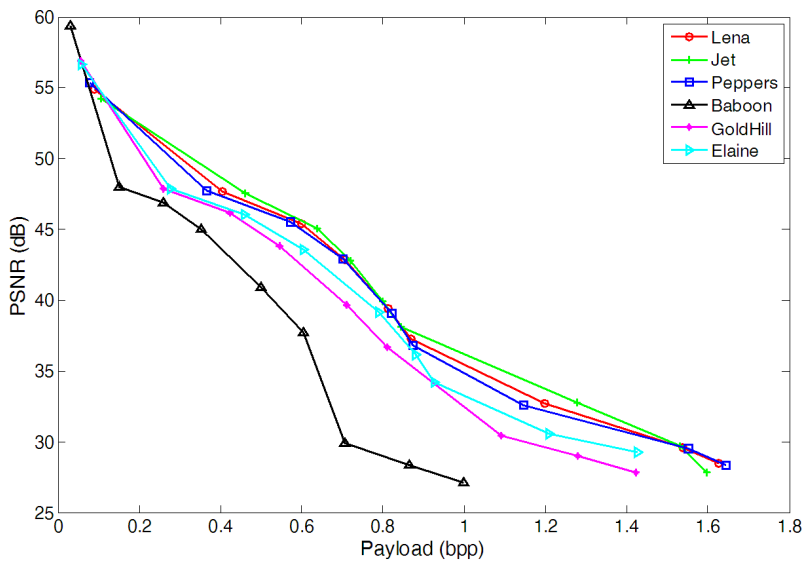
## 3.5 Experimental Results

Several greyscale images were used as host images, each with a size of  $512 \times 512$  pixels. A quarter of the host image *Lena* was used as the test data. To obtain a high PSNR performance, the integer  $k$  was set to 1, whereas a larger payload can be achieved by setting  $k$  to 2. To provide a variety of bit rate, the values of two control parameters  $\beta$  and  $\gamma$  were not fixed. In practical, the relationship between  $\beta$  and  $\gamma$  was  $|\beta| > |\gamma| > 0$ . Simulation generated by the proposed CA algorithm with the uses of codebook and prediction mean, respectively, were shown in the following subsections. The robustness of the CA algorithm performed in the IWT domain was also examined.

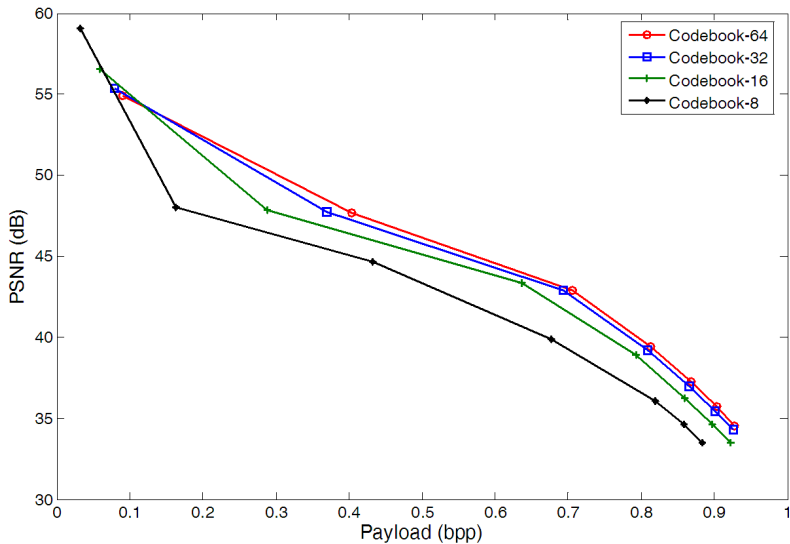
### 3.5.1 CA Algorithm with the Use of Codebook

Fig. 3.3 displayed the PSNR and payload that were achieved by the CA algorithm using the codebook of size 64. The block measured  $5 \times 5$ . The figure revealed that the PSNR was around 30 dB when the payload was about 1.6 bpp for images *Lena*, *Jet*, and *Peppers*. A maximum payload of 1.40 bpp was achieved for all images except *Baboon*.

To demonstrate the performance of the proposed method using codebooks of various size, Fig. 3.4 plotted the tradeoff between PSNR and payload for image *Lena*. Figure 3.4 demonstrated that a larger codebook corresponds to a larger PSNR achieved using the proposed method. In the legend to Fig. 3.4, ‘Codebook-64,’ ‘Codebook-32,’ ‘Codebook-16,’ and ‘Codebook-8,’ refer to applications of the CA algorithm using a codebook of 64, 32, 16, and 8 codewords, respectively.



**Fig. 3.3.** PSNR and payload yielded by CA algorithm using a codebook of size 64.



**Fig. 3.4.** PSNR and payload generated by the CA algorithm with codebooks of various sizes on image *Lena*.

The PSNR is defined by

$$\text{PSNR} = 10 \times \log_{10} \left( \frac{255^2}{\text{MSE}} \right), \quad (3.8)$$

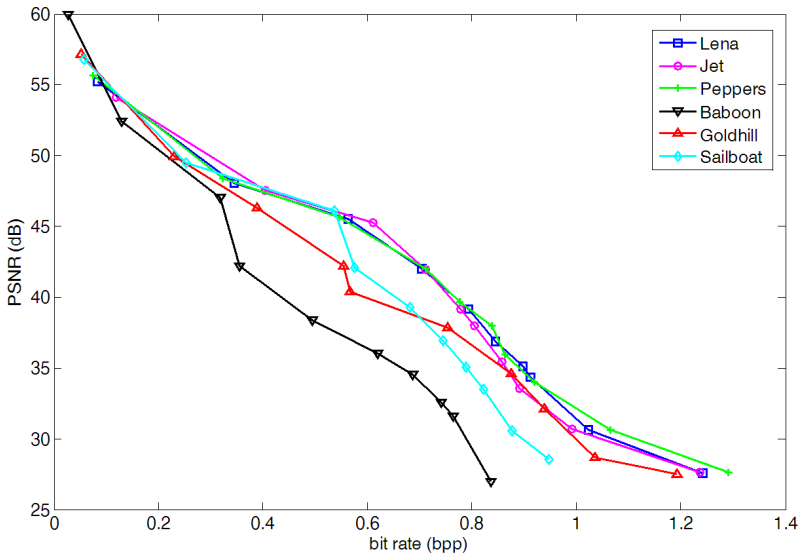
where

$$\text{MSE} = \frac{1}{MN} \sum_{i=1}^N \sum_{j=1}^M (\hat{x}(i, j) - x(i, j))^2$$

if the image size is  $M \times N$ . Here  $x(i, j)$  and  $\hat{x}(i, j)$  denote the pixel values of the original image and the marked image, respectively.

### 3.5.2 CA Algorithm with the Use of Prediction Mean

To demonstrate the hiding performance of the CA algorithm with the MEAN predictor, Fig. 3.5 depicted the relationship between the embedding rate and PSNR on several images. Block size was  $3 \times 3$ . The figure showed that, for all images except *Sailboat* and *Baboon*, the maximum hiding rate was 1.24 bpp with average PSNR value of 27.60 dB.



**Fig. 3.5.** PSNR and bit rate generated by the CA algorithm with the MEAN predictor on several images.



### 3.5.3 Robust Version of the CA Algorithm

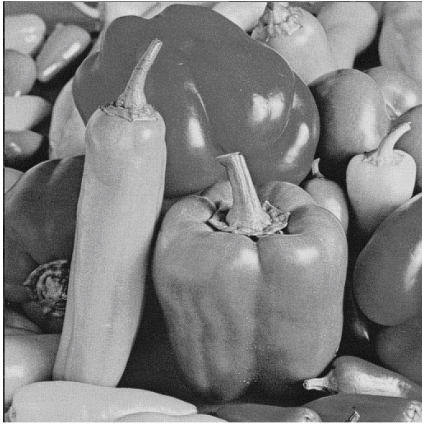
The marked images generated by embedding test data in host images via the CA algorithm implemented in the IWT domain were shown in Fig. 3.6. The block size was  $4 \times 4$ . It can be seen from Fig. 3.6 that the perceived quality was acceptable. Their average PSNR and bit rate were 30.97 dB and 1.235 bpp, respectively. Moreover, the relationship between PSNR and bit rate for the proposed method was depicted in Fig. 3.7. From the figure we can see that the maximum PSNR of value 48.30 dB can be achieved at the bit rate of 0.257 by the proposed method. On the other hand, the maximum hiding rate was 1.427 bpp with the PSNR of value 29.98 dB.



(a)



(b)

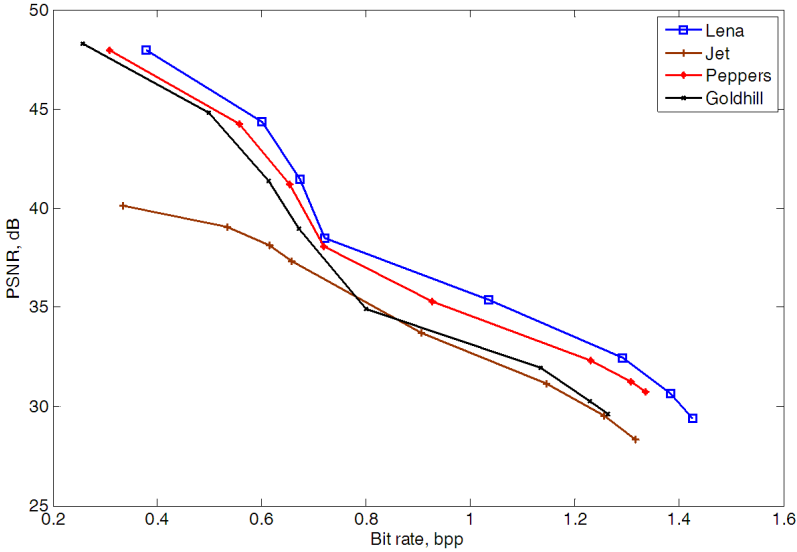


(c)



(d)

**Fig. 3.6.** The marked images (with PSNR and bit rate) generated by the CA algorithm performed in the IWT domain. (a) Lena (32.47 dB/1.293 bpp), (b) Jet (31.18 dB/1.147 bpp), (c) Peppers (31.77 dB/1.273 bpp), and (d) Goldhill (30.29 dB/1.229 bpp).



**Fig. 3.7.** Trade-off between PSNR and bit rate for the CA algorithm performed in the IWT domain.

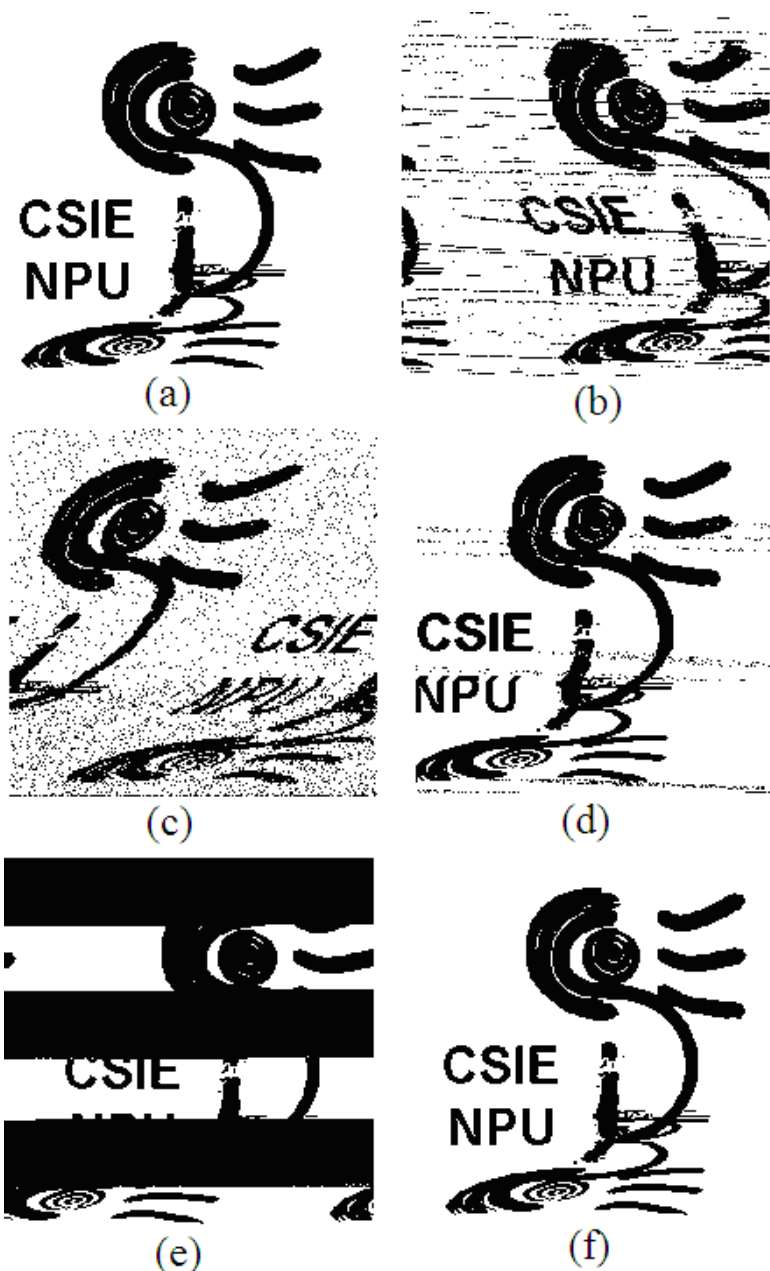
To reveal the robustness of the CA algorithm performed in the IWT domain, examples of extracted watermarks (size of  $210 \times 210$  pixels with 8 bits/pixel, 2 colours) after various manipulations of the image were depicted in Fig. 3.8. The bit correct ratio (BCR) was also included. The BCR is defined by

$$\text{BCR} = \left( \frac{\sum_{i=0}^{ab-1} w_i \oplus \tilde{w}_i}{a \times b} \right) \times 100\%, \quad (3.9)$$

where  $w_i$  and  $\tilde{w}_i$  represent the values of the original watermark and the extracted watermark, respectively, as well as the size of a watermark is  $a \times b$ . It is clear that the BCR for an extracted watermark is 100% when a marked image without being attacked, as shown in Fig. 3.8(a). Although the BCR in Figs. 3.8(b), 3.8(c), and 3.8(e) are not high, the extracted watermarks were identifiable. Figures 3.8(d) and 3.8(f) also confirm that the watermarks extracted from the manipulated images were recognizable.

### 3.5.4 Performance Comparison and Discussions

Several existing reversible data hiding techniques were compared with the proposed CA algorithms. Hiding performance using these techniques was tested in



**Fig. 3.8.** Examples of extracted watermarks after various manipulations. (a) Manipulation-free (BCR=100%); (b) Brightness (+50%) (BCR=66.6054%); (c) JPEG (CR=1.14) (BCR=67.6599%); (d) JPEG2000 (CR=1.37) (BCR=85.1111%); (e) Cropping (50%) (BCR=41.0113%); (f) Inversion (BCR=100%).

three images, namely, *Lena*, *Jet*, and *Baboon*, each of which had PSNR values of approximately 48 dB. Table 3.1 compared their hiding performance results. Note that the methods of ‘CA algorithm-A’ and ‘CA algorithm-B’ indicate the proposed CA algorithm using a codebook of size 64, and the MEAN predictor, respectively. Clearly, the average hiding capacity provided by the proposed methods including ‘CA algorithm-A’ and ‘CA algorithm-B’ yielded the largest payload among these methods with competitive PSNR values. Additionally, the hiding capacity provided by ‘CA algorithm-B’ was about three times that achieved by the Tai *et al.* approach [24] and four times that achieved by the Kim *et al.* scheme [12].

The payload and PSNR (around 30 dB) comparisons between the proposed CA algorithms and existing schemes were listed in Table 3.2. Note that ‘CA algorithm-C’ denotes the CA algorithm implemented in the IWT domain. Table 3.2 indicated that ‘CA algorithm-A’ provided the best performance of the average payload size, while the PSNR value yielded by ‘CA algorithm-C’ was superior.

**Table 3.1.** Hiding performance (Payload/ PSNR) comparison between various methods when PSNR value was approximately 48 dB.

| Methods                             | <i>Lena image</i>  | <i>Jet image</i>   | <i>Baboon image</i> | Average           |
|-------------------------------------|--------------------|--------------------|---------------------|-------------------|
| Lin <i>et al.</i> 's algorithm [16] | 65,349 /<br>48.67  | 69,941 /<br>48.67  | 38,465 /<br>48.67   | 57,919 /<br>48.67 |
| Tai <i>et al.</i> 's approach [24]  | 22,377 /<br>48.32  | 45,472 /<br>48.53  | 9,818 /<br>48.21    | 25,889 /<br>48.35 |
| Hong <i>et al.</i> 's technique [7] | 86,178 /<br>48.93  | 71,610 /<br>48.79  | 16,575 /<br>48.29   | 58,121 /<br>48.67 |
| Kim <i>et al.</i> 's scheme [12]    | 20,121 /<br>48.9   | 32,631 /<br>49     | 6,499 /<br>48.7     | 19,750 /<br>48.87 |
| CA algorithm-A                      | 105,925 /<br>47.65 | 120,914 /<br>47.56 | 39,431 /<br>48.01   | 88,757 /<br>47.74 |
| CA algorithm-B                      | 90,473 /<br>48.09  | 106,037 /<br>47.52 | 34,946 /<br>48.04   | 77,152 /<br>47.88 |

**Table 3.2.** Payload and PSNR comparison between the proposed method and other methods (with PSNR around 30 dB).

| Methods                              | <i>Lena image</i> | <i>Jet image</i> | <i>Peppers image</i> | <i>Goldhill image</i> | Average         |
|--------------------------------------|-------------------|------------------|----------------------|-----------------------|-----------------|
| Lin <i>et al.</i> 's technique [14]  | 231,971 / 30.2    | 289,877 / 30.1   | 268,042 / 30.2       | 234,160 / 30.1        | 256,013 / 30.15 |
| Hsiao <i>et al.</i> 's scheme [9]    | 303,700 / 30.00   | 286,488 / 30.00  | 303,736 / 30.00      | 245,370 / 30.00       | 284,824 / 30.00 |
| Zeng <i>et al.</i> 's algorithm [36] | 282,147 / 30.12   | 338,492 / 30.08  | 317,194 / 29.66/     | N/A                   | 310,681 / 30.30 |
| Yang <i>et al.</i> 's approach [31]  | 314,573 / 30.00/  | 311,404 / 30.71  | 317,194 / 29.66/     | 246,415 / 29.65/      | 297,397 / 30.01 |
| CA algorithm-A                       | 369,920 / 30.96   | 401,624 / 29.71  | 367,396 / 30.91      | 335,534 / 29.01       | 331,934 / 30.10 |
| CA algorithm-B                       | 325,825 / 27.59   | 324,309 / 27.66  | 338,578 / 27.65      | 312,907 / 27.49       | 325,405 / 27.60 |
| CA algorithm-C                       | 362,840 / 30.64   | 300,608 / 31.18  | 333,622 / 31.77      | 322,190 / 30.29       | 329,815 / 30.97 |

In addition, a comparison of the capacity against distortion for various methods in image *Lena* was depicted in Fig. 3.9. As shown in Fig. 3.9, 'CA algorithm-A' and 'CA algorithm-C' outperformed the other four schemes. The PSNR of 'CA algorithm-B' was better than those of other four schemes when the embedding rate was below 0.85 bpp.

From the above simulations we can see that 'CA algorithm-A' provided a larger payload than existing schemes at various embedding rates, and its corresponding PSNR is competitive. To further reduce overhead size, 'CA algorithm-B' utilized the prediction mean and yielded a good hiding capacity with high perceptual quality. Furthermore, 'CA algorithm-C' provided a high PSNR value when the embedding rate was larger than 0.36 bpp. Moreover, the marked images generated by 'CA

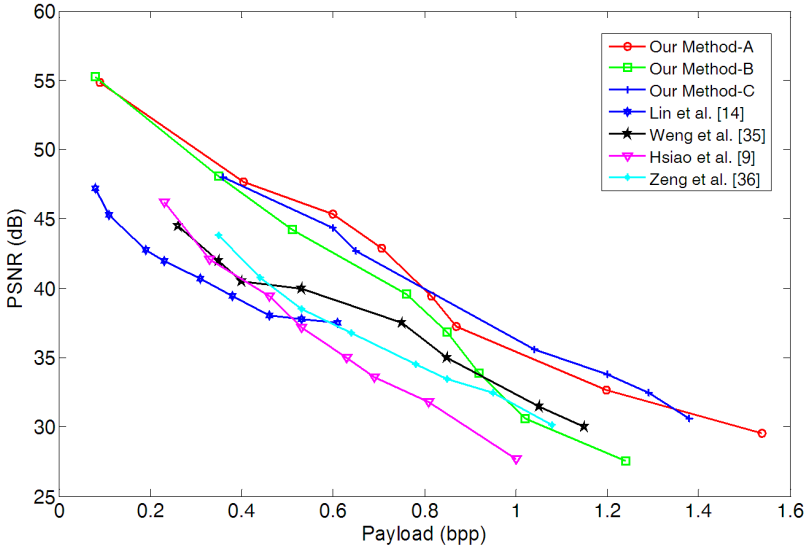


Fig. 3.9. Capacity against distortion comparison in image *Lena*.

algorithm-C' were robust against manipulations such as JPEG2000, JPEG, brightness adjustment, cropping, and inversion.

### 3.6 Conclusion

A simple, efficient reversible data hiding based on the coefficient adjustment (CA) algorithm was presented in this chapter. To yield a high hiding capacity, the mean removal firstly generated the differenced blocks from an input image. Then, the pixel squeezing approach further provided hiding storage while the pixel isolation keeping distortion low. Simulation demonstrated that the proposed CA algorithm with the uses of codebook and prediction mean provided a larger payload than existing schemes, and their corresponding PSNR were competitive. Moreover, the marked images generated by the robust version of the proposed CA algorithm can tolerate attacks from JPEG2000, JPEG, brightness adjustment, cropping, and inversion.

### References

1. Alattar, A.M.: Reversible watermark using the difference expansion of a generalized integer transform. *IEEE T. Image Processing* 13, 1147–1156 (2004)
2. Awrangjeb, M., Kankanhalli, M.S.: Lossless Watermarking Considering the Human Visual System. In: Kalker, T., Cox, I., Ro, Y.M. (eds.) *IWDW 2003*. LNCS, vol. 2939, pp. 581–592. Springer, Heidelberg (2004)

3. Awrangjeb, M., Kankanhalli, M.S.: Reversible watermarking using a perceptual model. *Journal of Electronic Imaging* 14, 013014-1-8 (2005)
4. Chen, M., Chen, Z., Zeng, X., Xiong, Z.: Reversible data hiding using adaptive prediction-error expansion. In: *The 11th ACM Workshop on Multimedia and Security*, pp. 19-24 (2009)
5. Chen, W.J., Chang, C.C., Le, T.H.N.: High payload steganography mechanism using hybrid edge detector. *Expert Systems With Applications* 37, 3292-3301 (2010)
6. Cox, I.J., Miller, M.L., Bloom, J.A., Fridrich, J., Kalker, T.: *Digital watermarking and steganography*, 2nd edn. Morgan Kaufmann, MA (2008)
7. Hong, W., Chen, T.S., Shiu, C.W.: Reversible data hiding for high quality images using modification of prediction error. *The Journal of Systems and Software* 82, 1833-1842 (2009)
8. Howard, P.G., Kossentini, F., Martins, B., Forchhammer, S., Rucklidge, W.J.: The emerging JBIG2 standard. *IEEE T. Circuits and Systems for Video Technology* 8, 838-848 (1998)
9. Hsiao, J.Y., Chan, K.F., Chang, J.M.: Block-based reversible data embedding. *Signal Processing* 89, 556-569 (2009)
10. Hu, Y., Lee, H.K., Li, J.: DE-based reversible data hiding with improved overflow location map. *IEEE T. Circuits and Systems for Video Technology* 19, 250-260 (2009)
11. Jung, K.H., Yoo, K.Y.: Data hiding method using image interpolation. *Computer Standard & Interfaces* 31, 465-470 (2009)
12. Kim, K.S., Lee, M.J., Lee, H.Y., Lee, H.K.: Reversible data hiding exploiting spatial correlation between sub-sampled images. *Pattern Recognition* 42, 3083-3096 (2009)
13. Lee, C.F., Chen, H.L., Tso, H.K.: Embedding capacity raising in reversible data hiding based on prediction of different expansion. *The Journal of Systems and Software* 83, 1864-1872 (2010)
14. Lin, C.C., Hsueh, N.L., Shen, W.H.: High-performance reversible data hiding. *Fundamenta Informaticae* 82, 155-169 (2007)
15. Lin, C.C., Hsueh, N.L.: A lossless data hiding scheme based on three-pixel block differences. *Pattern Recognition* 41, 1415-1425 (2008)
16. Lin, C.C., Tai, W.L., Chang, C.C.: Multilevel reversible data hiding based on histogram modification of difference images. *Pattern Recognition* 41, 3582-3591 (2008)
17. Luo, L., Chen, Z., Chen, M., Zeng, Q., Xiong, Z.: Reversible image watermarking using interpolation technique. *IEEE T. Information Forensics and Security* 5, 187-193 (2010)
18. Ni, Z., Shi, Y.Q., Ansari, N., Su, W.: Reversible data hiding. *IEEE T. Circuits and Systems for Video Technology* 16, 354-362 (2006)
19. Ni, Z., Shi, Y.Q., Ansari, N., Su, W., Sun, Q., Lin, X.: Robust lossless image data hiding designed for semi-fragile image authentication. *IEEE T. Circuits and Systems for Video Technology* 18, 497-509 (2008)
20. Pan, J.S., Huang, H.C., Jain, L.C.: *Intelligent Watermarking Techniques*. World Scientific (2004)
21. Pai, Y.T., Ruan, S.J.: A high quality robust digital watermarking by smart distribution technique and effective embedded scheme. *IEICE Trans. Fundamentals* E90-A, 597-605 (2007)
22. Qu, Z.G., Chen, X.B., Zhou, X.J., Niu, X.X., Yang, Y.X.: Novel quantum steganography with large payload. *Optics Communications* 283(13), 4782-4786 (2010)
23. Shih, F.Y.: *Digital watermarking and steganography: fundamentals and techniques*. CRC Press, FL (2008)
24. Tai, W.L., Yeh, C.M., Chang, C.C.: Reversible data hiding based on histogram modification of pixel differences. *IEEE T. Circuits and Systems for Video Technology* 19, 906-910 (2009)

25. Thodi, D.M., Rodriguez, J.: Expansion embedding techniques for reversible watermarking. *IEEE T. Image Processing* 16, 721–730 (2007)
26. Tian, J.: Reversible data embedding using a difference expansion. *IEEE T. Circuits and Systems for Video Technology* 13, 890–896 (2003)
27. Tsai, P., Hu, Y.C., Yeh, H.L.: Reversible image hiding scheme using predictive coding and histogram shifting. *Signal Processing* 89, 1129–1143 (2009)
28. Yang, C.H., Tsai, M.H.: Improving histogram-based reversible data hiding by interleaving predictors. *IET Image Processing* 4(4), 223–234 (2010)
29. Yang, C.Y., Lin, J.C.: Use of radius weighted mean to cluster two-class data. *Electronics Letter* 30(10), 757–759 (1994)
30. Yang, C.Y., Lin, J.C.: Image hiding by base-oriented algorithm. *Optical Engineering* 45, 117001–10 (2006)
31. Yang, C.Y., Hu, W.C., Lin, C.H.: Reversible data hiding by coefficient-bias algorithm. *Journal of Information Hiding and Multimedia Signal Processing* 1, 100–109 (2010)
32. Yang, C.Y., Hu, W.C., Hwang, W.Y., Cheng, Y.F.: A simple digital watermarking by the adaptive bit-labeling scheme. *International Journal of Innovative Computing, Information and Control* 6(3), 1401–1410 (2010)
33. Yang, H.W., Hwang, K.F., Chou, S.S.: Interleaving max-min difference histogram shifting data hiding method. *Journal of Software* 5, 615–621 (2010)
34. Wang, R.Z., Chen, Y.S.: High-payload image steganography using two-way block matching. *IEEE Signal Processing Letters* 13, 161–164 (2006)
35. Weng, S., Zhao, Y., Pan, J.S., Ni, R.: Reversible watermarking based on invariability and adjustment on pixels pairs. *IEEE Signal Processing Letters* 15, 721–724 (2008)
36. Zeng, X., Ping, L., Li, Z.: Lossless data hiding scheme using adjacent pixel difference based on scan path. *Journal of Multimedia* 4, 145–152 (2009)
37. Zeng, X.T., Ping, L.D., Pan, X.Z.: A lossless data hiding scheme. *Pattern Recognition* 43, 1656–1667 (2010)



---

## ICA-Based Image and Video Watermarking

Jiande Sun<sup>1</sup> and Ju Liu<sup>2</sup>

<sup>1</sup> School of Information Science and Engineering,  
Shandong University,  
Jinan, China  
jd\_sun@sdu.edu.cn

<sup>2</sup> School of Information Science and Engineering,  
Shandong University,  
Jinan, China  
juliu@sdu.edu.cn

**Summary.** With the development of computer science and information transmission, image and video have been applied more and more. Extraction of features match to human visual system and compact representation of images and videos are the key points in the field of image and video analysis. As an unsupervised learning method, Independent Component Analysis (ICA) has been used widely in image and video processing, because it is proved to match human visual system in image and video understanding. In this chapter, ICA model and FastICA algorithm are introduced firstly. And then several ICA-based image and video processing models are presented, and various independent features corresponding to these models are introduced. Finally, the similarity between the ICA- and watermarking-models is analyzed, and some representative ICA-based image and video watermarking algorithms are presented.

### 4.1 Introduction of Independent Component Analysis

Independent Component Analysis (ICA) is a method of signal processing and data analysis developed in the research of blind signal processing. It defines a generative model in which observed signals are expressed as a linear transform of components that are statistically independent from each other, and it can estimate these unknown independent components in the condition that the mixing system is also unknown [1]-[15].

Generally speaking, the simplest noise free ICA model can be represented by

$$\mathbf{X}(t) = \mathbf{A}\mathbf{S}(t), \quad (4.1)$$

where  $\mathbf{X}(t) = [\mathbf{x}_1(t), \mathbf{x}_2(t), \dots, \mathbf{x}_M(t)]^T$  is an observed signal vector at discrete time  $t$ ,  $\mathbf{S}(t) = [\mathbf{s}_1(t), \mathbf{s}_2(t), \dots, \mathbf{s}_M(t)]^T$  is a source signal vector. Each of the observations  $\mathbf{x}_i(t), i = 1, 2, \dots, M$ , is a linear combination of  $M$  unknown independent sources

$\mathbf{s}_i(t)$ , and  $\{\mathbf{s}_i(t), i = 1, 2, \dots, M\}$  are  $M$  mutually independent stochastic signals. The  $M \times M$  mixing matrix  $\mathbf{A} = (a_{ij})$  contains the mixture coefficients, where  $a_{ij}$  is the  $(i, j)^{th}$  element. Here, as in the follow  $\cdot^T$ , denotes transposition.

ICA is to estimate a linear system  $\mathbf{W}$  operating on  $\mathbf{X}(t)$ , so that

$$\mathbf{Y}(t) = \mathbf{W}\mathbf{X}(t), \quad (4.2)$$

where  $\mathbf{Y}(t) = [y_1(t), y_2(t), \dots, y_M(t)]^T$  is an estimation of  $\mathbf{S}(t)$  and is an  $M \times M$  separating matrix.

If both of the sources signal  $\mathbf{S}(t)$  and the mixing matrix  $\mathbf{A}$  are unknown, it is impossible to exactly reconstruct the sources without any a prior knowledge about  $\mathbf{A}$  and  $\mathbf{S}(t)$ , so in general, it is assumed that:

- (A1)  $\mathbf{A}$  must be a constant non-singular matrix.
- (A2) The source signals  $\{\mathbf{s}_i(t), i = 1, 2, \dots, M\}$  must be mutually statistically independent at each  $t$ .
- (A3) Each source signal  $\mathbf{s}_i(t)$  is a stationary zero-mean stochastic signal and only one of the source signals is allowed to have a Gaussian marginal distribution. In the following sections, the time index  $t$  will be dropped for simplification.

Many principles are used to estimate the model of ICA, which are based on Maximization of Non-Gaussianity [16], Maximum Likelihood Estimation [17, 18], Minimization of Mutual Information [19], and so on. Among them, The FastICA algorithm is a simple and efficient fixed-point algorithm [4]. It measures the independence between signals based on the non-Gaussian objective function. The observation  $X$  is pre-whitened by principal component analysis (PCA) through  $\mathbf{v} = \mathbf{T}\mathbf{X}$ , whose components are mutually uncorrelated and all have unit variance. Here the self-correlation matrix of  $\mathbf{v}$  is a unit matrix and  $\mathbf{T}$  is the whiten matrix that is usually defined as:

$$\mathbf{T} = \Lambda^{-1/2}\mathbf{E}^T, \quad (4.3)$$

where  $\Lambda = \text{diag}[\lambda_1, \lambda_2, \dots, \lambda_M]$ ,  $\lambda$  is the  $i^{th}$  largest eigenvalue of the covariance matrix of  $X$ , which is  $\mathbf{R}_X = E[\mathbf{X}\mathbf{X}^T]$ .  $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_M]$ , where  $\mathbf{e}_i$  is the corresponding eigenvector of  $\lambda_i$ . The eigenvectors corresponding to larger eigenvalues are called principal eigenvectors.

Then this problem can be resolved by searching a linear combination of  $\mathbf{v}$ , say,  $\mathbf{D}^T \mathbf{v}$ , such that it has maximal or minimal kurtosis. The objective function of FastICA by kurtosis is:

$$\begin{aligned} \text{kurt}(\mathbf{D}^T \mathbf{v}) &= E\{(\mathbf{D}^T \mathbf{v})^4\} - 3[E\{(\mathbf{D}^T \mathbf{v})^2\}]^2 \\ &= E\{(\mathbf{D}^T \mathbf{v})^4\} - 3\|\mathbf{D}\|^4, \end{aligned} \quad (4.4)$$

$$\mathbf{D}(k) = E\{\mathbf{v}(\mathbf{D}(k-1)\mathbf{v})^3\} - 3\mathbf{D}(k-1), \quad (4.5)$$

where  $k$  is the times of iteration. The final de-mixing matrix is  $\mathbf{W} = \mathbf{D}^T \mathbf{T}$ . So the estimation of source signal  $\hat{\mathbf{S}}$  can be obtained through  $\hat{\mathbf{S}} = \mathbf{Y} = \mathbf{W}\mathbf{X}$ .

In FastICA, the gradient-based algorithm is replaced by the fixed-point iteration algorithm to get the de-mixing matrix. This innovation makes the convergence of FastICA be more reliable and cubic. Comparison with gradient based algorithms shows that FastICA is usually 10 to 100 times faster and only 5 to 10 iterations seem to be enough to achieve the convergence. In addition, the independent components are obtained one after another in this algorithm. And before the estimation of each independent component, its corresponding column vector of de-mixing matrix is firstly orthogonalized and normalized. So the final de-mixing matrix is an orthogonal one.

If only the principal eigenvectors are retained to construct the whiten matrix  $\mathbf{T}$ , the dimension can be decreased and the optimal estimation of principal independent component can be achieved. This can be called Principal Independent Component Analysis (PICA). The difference between the PICA and ICA is whether the dimension is decreased during the pre-whitening with remaining only the principal components. During the pre-processing of PICA through PCA, the dimension can be decreased from  $M$  to  $P$ . Here  $P$  is the minimum that satisfies

$$\frac{\sum_{i=2}^P \lambda_i}{\sum_{i=2}^M \lambda_i} > g,$$

and  $g$  determines the number of decreased dimension.

Since ICA can reveal the hidden factors that underlie sets of the observed signals, it is reported that there are lots of real-world applications of it, including biomedical signal processing [20], speech signal separation [21], feature extraction [22, 23, 24], financial time series analysis [25, 26, 27], data mining [28, 29], to name just a few.

## 4.2 Independent Feature of Image and Video

As a signal processing method, ICA is to present a set of variables by the linear mixture of statistically independent variables [3]. It has two most important applications, i.e. Blind Signal Separation and Feature Extraction, which are meaningful for digital watermark. From the view of ICA-based image and video processing, how to derive the observed signals from only one image or video is what has to be resolved firstly. In the following sub-sections, the methods of ICA-based image and video feature extraction are presented respectively [30].

### 4.2.1 Independent Feature of Image

As far as we know, the methods of ICA-based image feature extraction can be classified into two kinds according to the way of deriving the observed signals of ICA model from a single image. One is based on blocking, and the corresponding extracted feature is called independent image block feature. The other is based on downsampling, and the corresponding extracted feature is called independent downsampling feature. Both of them are described in this part.

## Independent Image Block Feature

Hyvarinen A. et al applied ICA to feature extraction from images in [14, 15], and aimed to obtain some general features useful for potential image processing, such as image denoising, compression, etc. In his method, 1000 image blocks of  $16 \times 16$  pixels taken at random locations from natural images are used as the observed signals of ICA model, and FastICA algorithm is used to estimate the source signals, i.e., independent components. These obtained independent components shown in Fig. 4.1 describe spatial frequency information, edges with different orientation, etc. Hence they are considered as basis vectors which can be used for image reconstruction.

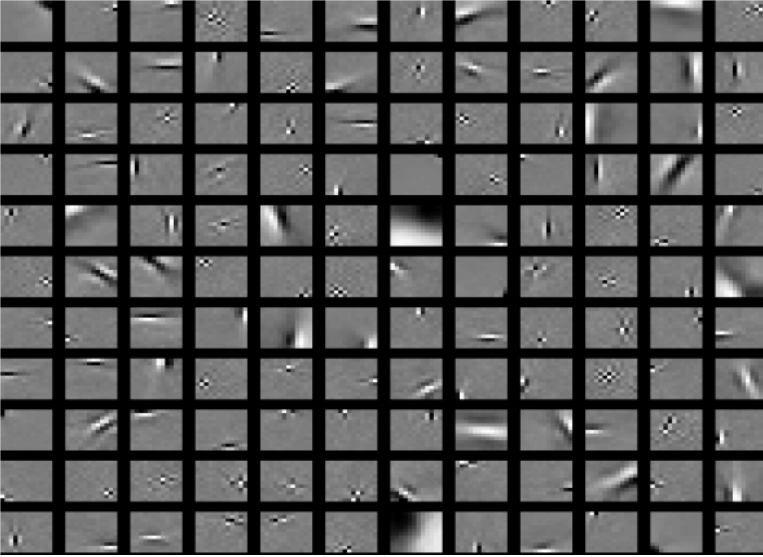


Fig. 4.1. ICA basis vectors [14, 15].

Though Hyvarinen A. did some corresponding experiments on the above features, and proved the representation of image based on this kind of basis vectors are similar to that in human visual system, this feature extraction was based on image database, not a single image. So in order to obtain the specific features for a single image, the image  $\mathbf{I}$  is divided into  $M$  non-overlapped blocks, each of which is expressed as a row vector  $\mathbf{x}_i$ ,  $i = 1, \dots, M$  of the observed signal  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M]^T$ . And after the operation of ICA, the estimated source signals  $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_M]^T$  can be obtained by Eq. (4.2), where  $\mathbf{y}_i^T$ ,  $i = 1, \dots, M$  is the derived independent image block feature [31, 32, 33]. This procedure is shown in Fig. 4.2

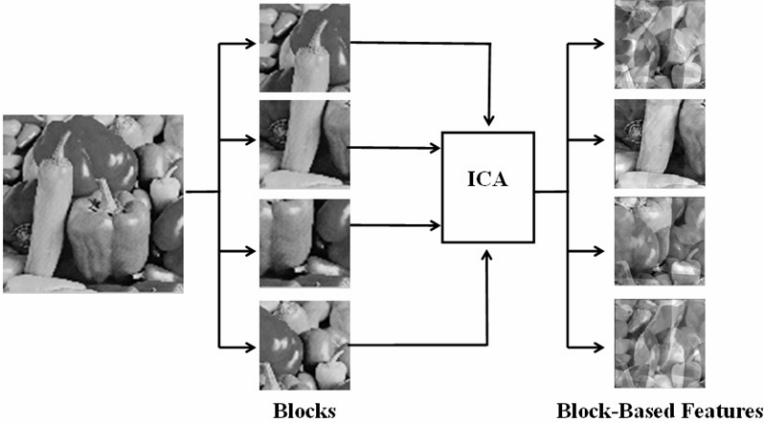


Fig. 4.2. Extraction of independent image block Feature.

### Independent Downsampling Feature

During the above extraction of independent image block feature, each block can only describe the regional properties of the image, and the global properties of the image are missed. Therefore, the obtained features can't represent the image globally. As an alternative method, downsampling can form some sub-images with lower resolution from one image, which can be taken as the observed signals in the model of ICA.

Assuming the size of the image  $\mathbf{I}$  is  $n \times m$ , after down-sampling with factor 2, the image is downsampled into 4 sub-images shown in Fig. 4.3. The 4 sub-images are:

$$\mathbf{I}_{sub1}(i, j) = \mathbf{I}(2i - 1, 2j - 1), \quad (4.6a)$$

$$\mathbf{I}_{sub2}(i, j) = \mathbf{I}(2i - 1, 2j), \quad (4.6b)$$

$$\mathbf{I}_{sub3}(i, j) = \mathbf{I}(2i, 2j - 1), \quad (4.6c)$$

$$\mathbf{I}_{sub4}(i, j) = \mathbf{I}(2i, 2j), \quad (4.6d)$$

where  $\mathbf{I}$  is the original image,  $i = 1, 2, \dots, \frac{n}{2}$ , and  $j = 1, 2, \dots, \frac{m}{2}$ . And the sub-images are taken as the observed signals to perform ICA. These sub-images are similar with each other, and preserve the global properties of the original image. This kind of processing is called ICA Transform (ICAT) in [34]. The 4 Feature Images (FI) obtained by using ICAT, 4 sub-bands like, are shown in Fig. 4.4.

Fig. 4.4 shows a result which is very similar to that of DWT. Therefore, a comparison is performed, in which  $512 \times 512$  standard image, peppers, is used as the original image, and Daubechies-4 as the mother wavelet is selected. And down-sampling with factor 2 is adopted in ICAT.

In order to analyze the property of the FIs extracted by ICAT, the approximate component is used to take the place of the details each at once, and the inverse transform is executed after each replacement. The results of inverse DWT (IDWT)

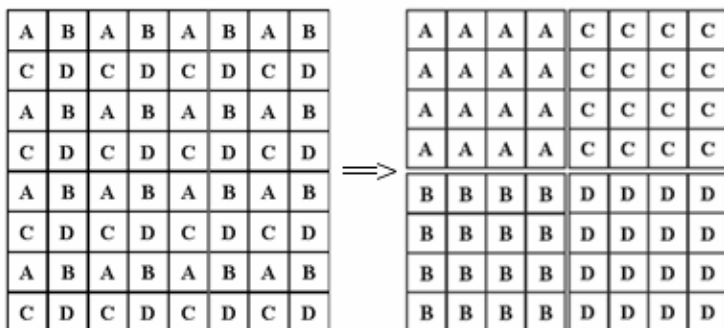


Fig. 4.3. The diagram of image down-sampling, which results four sub-images.

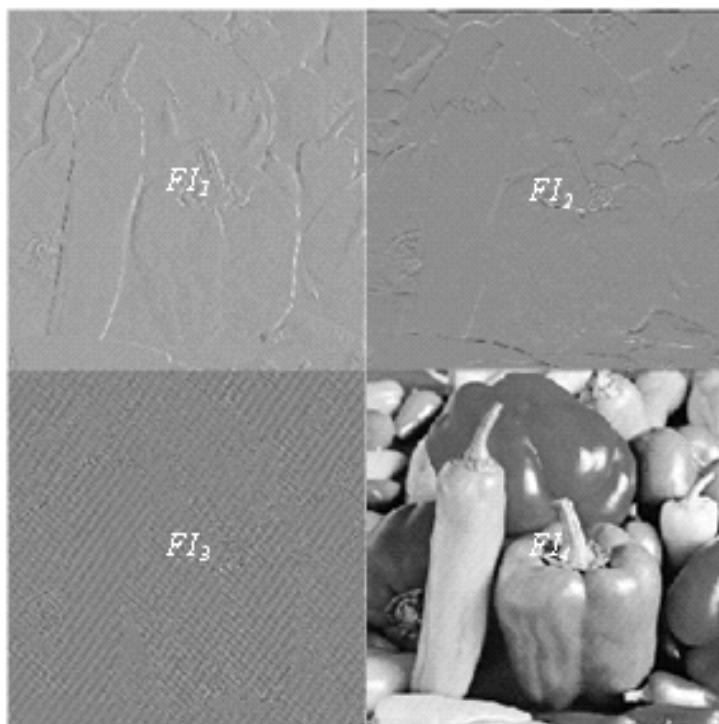
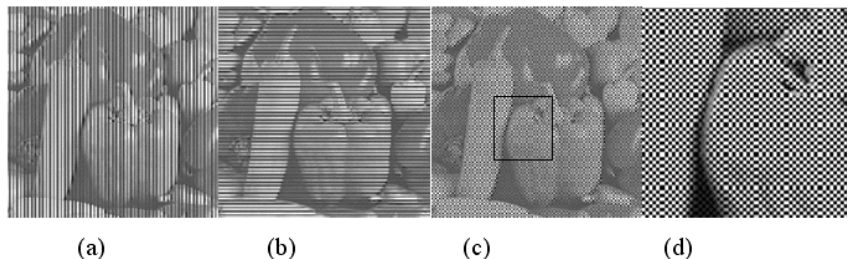
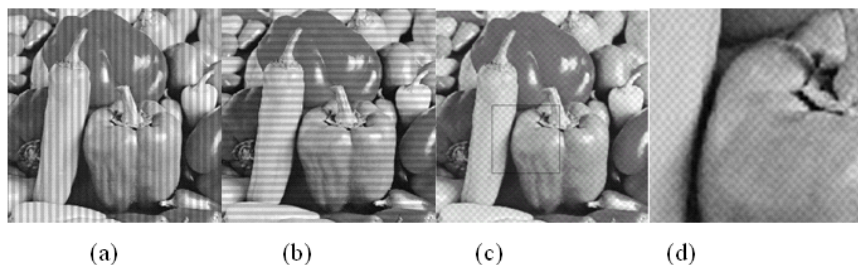


Fig. 4.4. The 4 FIs obtained by ICAT.  $FI_4$  is the approximate component of the original image, while the  $FI_1$ ,  $FI_2$ ,  $FI_3$  are the details of the original image.

and inverse ICAT (IICAT) are shown in Fig. 4.5 and Fig. 4.6 respectively. From Fig. 4.5, it can be seen that each result of such kind of procedure has the obvious textures in a certain direction, e.g. Fig. 4.5(a), which has heavy vertical textures, shows the vertical detail has been replaced by the approximate component.



**Fig. 4.5.** (a), (b), and (c) are the results of IDWT after the replacement of LH, HL, and HH respectively. And (d) is the enlarged of the rectangle part in (c).



**Fig. 4.6.** (a), (b), and (c) are the results of IICAT after the replacement of FI1, FI2, and FI3 in Fig. 4.3 respectively. And (d) is the enlarged of the rectangle part in (c).

Given the analysis of Fig. 4.5, Fig. 4.6 shows that FI1, FI2, and FI3 in Fig. 4.4 are also some directional details of the original image. Furthermore, DCT is performed on the approximate component obtained by DWT and ICAT, respectively, to analyze their frequency characteristics. Among the AC coefficients of the DWT approximate component, the first 5627 lowest frequency AC coefficients occupy the 95% in energy, while in the case of ICAT approximate component, the number is only 3451, which is only about 61% of that in DWT. It suggests that ICAT is potential in image and video compression.

### 4.2.2 Independent Feature of Video

Different from image, the ICA-based feature extraction from video is classified into spatial and temporal methods. The spatial method means in this kind of method, the video is divided spatially into some sub-videos, and these sub-videos are used to extract video features. However, the temporal method means the frames are processed by ICA temporally to extract video features. In this part, independent video block feature, independent dynamic feature, and independent content feature are described, which are obtained through spatial, temporal and temporal methods respectively.

#### Independent Video Block Feature

As an extension of independent image block feature extraction, a video is divided into 3D blocks and a kind of independent feature can be obtained in the same way as for images. Each frame of the video sequence is divided into blocks spatially. The blocks in the same position of each frame form the 3D video blocks. These 3D video blocks are considered as the observation  $\mathbf{X}$ , and decomposed by ICA into Independent Components (IC), which are some kind of video, and they are considered as independent video block features [35]. This process is shown in Fig. 4.7

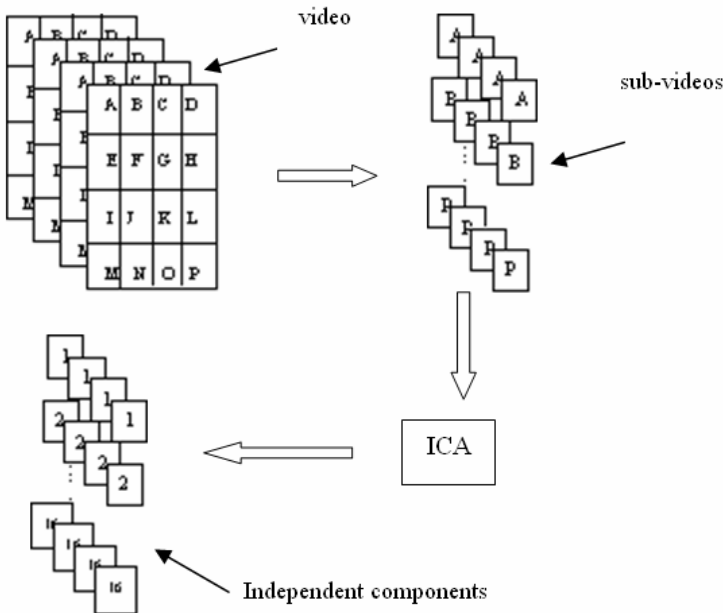


Fig. 4.7. Extraction of Independent Video Block Feature.



The ICs are proved to be the spatiotemporal constituents of the video blocks and the independent component filters are qualitatively similar to spatiotemporal properties of simple cells in primary visual cortex in [36].

### Independent Dynamic Feature from Successive Frames

As we all known, the pivotal technique in video compression is motion compensation based on block. And the relationship between corresponding blocks can be described as follows:

$$\mathbf{B}(i) = \mathbf{B}(i-1) + \mathbf{M}_B(i-1). \quad (4.7)$$

The current block  $\mathbf{B}(i)$  is the sum of the former one  $\mathbf{B}(i-1)$  and the motion compensation  $\mathbf{M}_B(i-1)$  between them. Here,  $\mathbf{B}(i-1)$  is considered as the static component of  $\mathbf{B}(i)$ , while  $\mathbf{M}_B(i-1)$  is the dynamic one. And  $\mathbf{M}_B(i-1)$  is independent of  $\mathbf{B}(i-1)$ . Given this, the relationship between consecutive frames can be described as follows:

$$\mathbf{F}(i-1) = k_{11}\mathbf{F}_s + k_{12}\mathbf{M}_F, \quad (4.8a)$$

$$\mathbf{F}(i) = k_{21}\mathbf{F}_s + k_{22}\mathbf{M}_F. \quad (4.8b)$$

Here  $\mathbf{F}(i)$ ,  $\mathbf{F}(i-1)$  are two consecutive frames.  $\mathbf{F}_s$ ,  $\mathbf{M}_F$  are respectively the static and dynamic components of  $\mathbf{F}(i)$ ,  $\mathbf{F}(i-1)$ , and it is reasonable to consider that  $\mathbf{F}_s$  and  $\mathbf{M}_F$  are independent from each other.  $k_{11}$ ,  $k_{12}$ ,  $k_{21}$ ,  $k_{22}$  are the mixing coefficients.

According to Eq. (4.8a), Eq. (4.8b) and Eq. (4.1), two consecutive frames  $\mathbf{F}(i)$ ,  $\mathbf{F}(i-1)$  can be considered as the observations  $\mathbf{X}_{F(i)}$ ,  $\mathbf{X}_{F(i-1)}$  of ICA model, and  $\mathbf{F}_s$ ,  $\mathbf{M}_F$  are the sources  $\mathbf{S}_{F_s}$ ,  $\mathbf{S}_{M_F}$  respectively. Thus the relationship between two consecutive frames can also be illustrated by the model of ICA,

$$\begin{bmatrix} \mathbf{x}_{F(i)} \\ \mathbf{x}_{F(i-1)} \end{bmatrix} = \mathbf{A} \begin{bmatrix} \mathbf{s}_{F_s} \\ \mathbf{s}_{M_F} \end{bmatrix}, \quad (4.9)$$

where  $\mathbf{x}_{F(i)}$ ,  $\mathbf{x}_{F(i-1)}$ ,  $\mathbf{s}_{F_s}$ ,  $\mathbf{s}_{M_F}$  are the row vectors of  $\mathbf{X}_{F(i)}$ ,  $\mathbf{X}_{F(i-1)}$ ,  $\mathbf{S}_{F_s}$ ,  $\mathbf{S}_{M_F}$  respectively. It clarifies that ICA can be applied to extract the static component,  $\mathbf{s}_{F_s}$ , and the dynamic component,  $\mathbf{s}_{M_F}$ , of two consecutive frames. Fig. 4.8 shows the extraction results. Two consecutive video frames in Fig. 4.8(a) and 4.8(b) are the observations, and Fig. 4.8(c) and 4.8(d) are the dynamic and static components respectively.

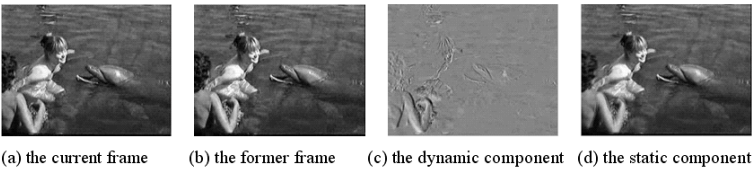


Fig. 4.8. ICA based separation results of two consecutive frames.

That Fig. 4.8(d) is similar to Fig. 4.8(a) and 4.8(b) shows that Fig. 4.8(d) is the common static component of the two frames. In Fig. 4.8(c), there is texture only in the region of motion. Furthermore, the edges indicate the relative motions. That is to say, conspicuous edges show great motions, the blurry ones mean slight motions, and no edges, no motions. To put it another way, the extracted dynamic component is able to describe the relative motions between the consecutive frames, and the extraction based on frames avoids the repetition of operation on blocks in the conventional method. Though the dynamic component is extracted wholly, it can also reflect the partial motion.

### Independent Content Feature of Video Shot

The frames in the same video shot are regarded as the observed signals of Eq. (4.1).

$$\mathbf{S}_f = \mathbf{W}_f \mathbf{X}_f, \quad (4.10)$$

where  $\mathbf{X}_f = [\mathbf{x}_f^1 \mathbf{x}_f^2 \cdots \mathbf{x}_f^K]^T$ , and  $\mathbf{x}_f^k$ , ( $1 \leq k \leq K$ ) is the lexicographical order column vector of  $k^{\text{th}}$  frame  $\mathbf{X}_F^k$ ,  $K$  is the total frame number in this shot.

Thus,  $\mathbf{x}_f^k = [\mathbf{X}_F^k(1, 1), \cdots, \mathbf{X}_F^k(M, 1), \mathbf{X}_F^k(1, 2), \cdots, \mathbf{X}_F^k(M, 2), \cdots, \mathbf{X}_F^k(M, N)]^T$ . Here  $\mathbf{X}_F^k(m, n)$  is the element of  $\mathbf{X}_F^k$  at the row  $m^{\text{th}}$  column  $n^{\text{th}}$ , and  $1 \leq m \leq M$ ,  $1 \leq n \leq N$ . The independent components are  $\mathbf{S}_f = [\mathbf{s}_f^1 \mathbf{s}_f^2 \cdots \mathbf{s}_f^L]^T$ , and  $L$  is the total number. Transform  $\mathbf{s}_f^l$ , ( $1 \leq l \leq L$ ) to matrix form

$$\mathbf{S}_F^l = \begin{bmatrix} \mathbf{s}_f^l(1) & \mathbf{s}_f^l(M+1) & \cdots & \mathbf{s}_f^l((N-1)M+1) \\ \mathbf{s}_f^l(2) & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{s}_f^l(3) & \mathbf{s}_f^l(2M) & \cdots & \mathbf{s}_f^l(NM) \end{bmatrix}_{M \times N}$$

that has the same size as frame, and  $\mathbf{S}_F^l$  is called Independent Feature Frame (IFF) here.

When the PCA pre-processing of PICA is done, the dimension is decreased from  $D$  to  $P$ , where  $D$  is the number of frames in a shot.  $P$  is the minimum that satisfies

$$\left( \sum_{j=2}^P \lambda_j / \sum_{j=2}^D \lambda_j \right) > 0.9.$$

$\lambda_j$  is the  $j^{\text{th}}$  largest eigenvalue of the covariance matrix of  $\mathbf{X}_f$ . And the obtained features, called Principal Independent Component Feature (PICF), IFF<sub>1</sub>-IFF<sub>6</sub>, are shown in Fig. 4.9.

In Fig. 4.9, the independent content features of the video of hand playing notes on the piano keyboard in [8] are shown as an example. From Fig. 4.9(a), it can be seen that IFF<sub>1</sub>-IFF<sub>5</sub> are corresponding to the fourth, the second, the first, the third and the fifth key respectively, and IFF<sub>6</sub> denotes the background. IFF<sub>6</sub> is proved as the background component of this video, for it corresponds to a constant weight in Fig. 4.9(b). The peaks and valleys in Fig. 4.9(b) provide a correct timing of each

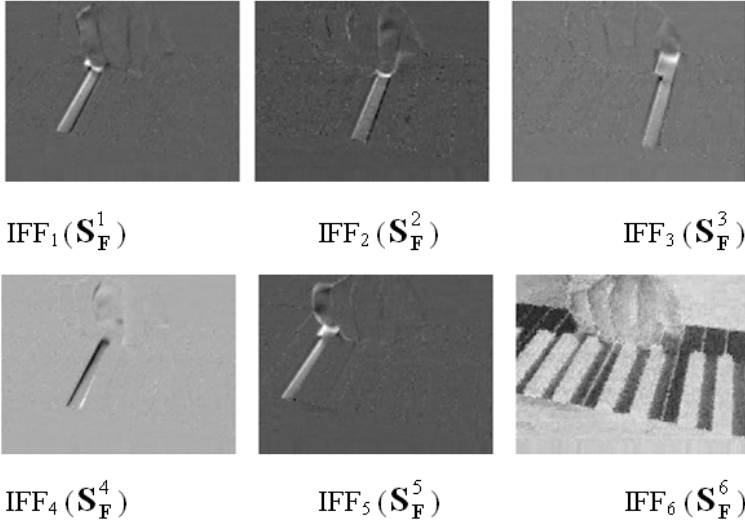
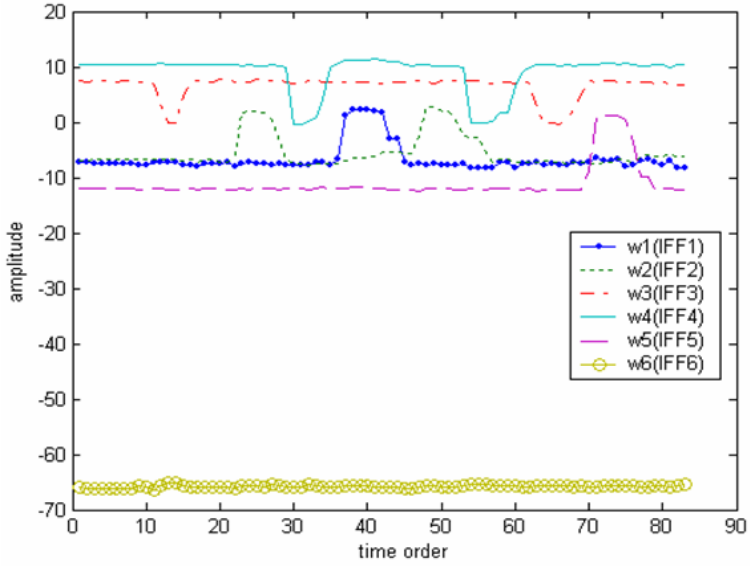
(a) IFF<sub>1</sub>- IFF<sub>6</sub> (S<sub>F</sub><sup>1</sup>-S<sub>F</sub><sup>6</sup>) extracted by ICA(b)  $w_1 - w_6$  of de-mixing matrix  $W_f$  corresponding to IFF<sub>1-6</sub>

Fig. 4.9. Independent feature frames and their weights.

pressed key in this playing. The peaks and valleys here are caused by the amplitude indeterminacy of ICA. Fig. 4.1 demonstrates that the extracted PICFs and their weights can represent the video content.

### 4.3 ICA-Based Image Watermark

Digital watermarking is to add the specific additional information into digital media, such as digital audio, image, video, etc., and extract/detect the information out for identification, authentication, and so on. The embedding and extraction in the model of digital watermarking are quite similar with the mixing and separation in the model of ICA, and the independent features mentioned in Sec. 4.2.2 can also be utilized as the watermarking domain. Therefore, lots of ICA-based watermarking schemes have been developed [31, 32, 33], [37]-[46]. Some representative schemes are presented in the following [30].

#### 4.3.1 ICA-Based Watermark Detection and Extraction

According to the similarity between the models of watermarking and ICA, the watermark image and the host image can be considered as two source signals. And with a  $2 \times 2$  mixing matrix, two mixed images are obtained, which are considered as the watermarked image and a key image for watermark extraction respectively. The scheme in [43] is presented here, which is based on this similarity.

#### Watermark Embedding

The host image and watermark image have the same size, and they are taken as two source signals in the model of ICA. The watermark embedding is

$$\begin{bmatrix} x_{\mathbf{I}_w} \\ x_{\mathbf{I}_k} \end{bmatrix} = \mathbf{A}_m \begin{bmatrix} s_{\mathbf{I}} \\ s_{\mathbf{w}} \end{bmatrix}, \quad (4.11)$$

where  $s_{\mathbf{I}}$  and  $s_{\mathbf{w}}$  are corresponding row vectors of the host image  $\mathbf{I}$  and watermark image  $\mathbf{w}$  respectively.  $x_{\mathbf{I}_w}$  and  $x_{\mathbf{I}_k}$  are corresponding row vectors of the watermarked image  $\mathbf{I}_w$  and the key image  $\mathbf{I}_k$  for watermark detection.  $\mathbf{A}_m$  is the mixing matrix, i.e. watermark embedding function. If  $\mathbf{A}_m = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ ,

$$a_{11}s_{\mathbf{I}} + a_{12}s_{\mathbf{w}} = x_{\mathbf{I}_w}, \quad (4.12a)$$

$$a_{21}s_{\mathbf{I}} + a_{22}s_{\mathbf{w}} = x_{\mathbf{I}_k}. \quad (4.12b)$$

With regards to the invisibility of watermark, it is necessary to set  $|a_{11}| \gg |a_{12}|$  to make the watermarked image have good quality. Both  $a_{21}$  and  $a_{22}$  are not zero.

## Watermark Detection

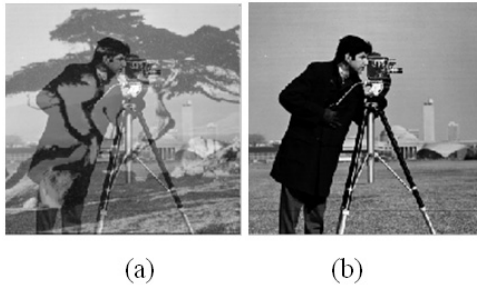
During watermark detection, for the to-be-detected watermarked image  $\mathbf{I}'_w$ , the detection is:

$$\begin{bmatrix} s'_I \\ s'_w \end{bmatrix} = \mathbf{W}_m \begin{bmatrix} x'_{I_w} \\ x_{I_K} \end{bmatrix}, \quad (4.13)$$

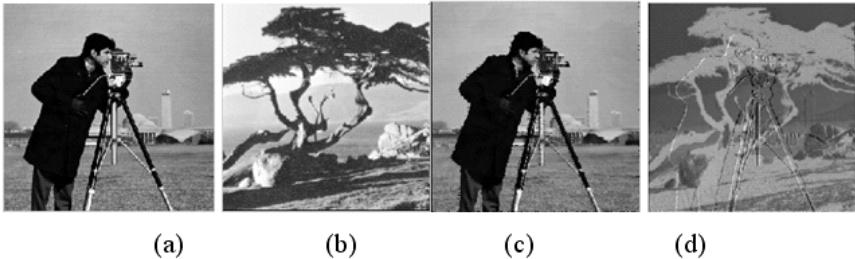
where  $x'_{I_w}$  is a row vector corresponding to  $\mathbf{I}'_w$ ,  $s'_w$  is a row vector corresponding to the detected watermark image  $w'$ , and  $s'_I$  is a row vector corresponding to an image  $I'$  similar to the host image.  $\mathbf{W}_m$  is the separation matrix for watermark detection.

## Experiments

In the experiments, the standard image, cameraman.tif in MATLAB, with size of  $256 \times 256$  is selected as the host image, and the image, tree.tif in MATLAB, which has the same size, is selected as the watermark image.  $\mathbf{A}_m = \begin{bmatrix} 0.7 & 0.026 \\ 0.72 & 0.6 \end{bmatrix}$  is generated randomly. The mixed images are shown in Fig. 4.10. Different attacks are added to the watermarked image. Fig. 4.11 shows these results.



**Fig. 4.10.** mixed images. (a) the key image, (b) the watermarked image (PSNR= 38.9557 dB).



**Fig. 4.11.** Experiment Results. (a) Compressed image by JPEG, (b) Extracted watermark from the compressed image, (c) Median filtered image, (d) Extracted watermark from the filtered image.

The experiment results demonstrate that this scheme can achieve high information embedding rate with good image quality and robustness. But this scheme is a non-blind one. Similar to the idea of this scheme, there are several other schemes such as the schemes described in [39, 41, 42, 44].

### 4.3.2 Watermarking Based on Independent Image Block Feature

As the independent image block feature is proved to be similar with that extracted by human visual system, this kind of feature is used widely in image processing. Furthermore, from the view of information embedding rate, Moulin et al proved that independent features are optimal for watermark embedding [47], so that some watermarking schemes based on independent image block feature have been proposed. The scheme in [31] is described here as an example.

#### Watermark Embedding

The host image  $\mathbf{I}$  is divided into  $M$  blocks, each of which is expressed as a row vector  $\mathbf{x}_{Bk}^T$ ,  $k = 1, \dots, M$ , and regarded as the observation vector, and then

$$\mathbf{S}_B = \mathbf{W}_B \mathbf{X}_B, \quad (4.14)$$

where  $\mathbf{S}_B = [\mathbf{s}_{B1}^T, \mathbf{s}_{B2}^T, \dots, \mathbf{s}_{BM}^T]^T$ ,  $\mathbf{X}_B = [\mathbf{x}_{B1}^T, \mathbf{x}_{B2}^T, \dots, \mathbf{x}_{BM}^T]^T$ , and  $\mathbf{s}_{Bk}^T$ ,  $k = 1, \dots, M$  is the derived IFC. Let  $\mathbf{W}_B = [\mathbf{W}_{B1}^T, \mathbf{W}_{B2}^T, \dots, \mathbf{W}_{Bm}^T]^T$  and  $\|\mathbf{W}_{Bt}^T\|_2 = \min_{1 \leq i \leq M} \|\mathbf{W}_{Bi}^T\|_2$ , and the IFC  $\mathbf{s}_{Bt}^T$  is selected to embed the watermark, and now

$$\mathbf{s}_{Bt}^T = \mathbf{W}_{Bt}^T \mathbf{X}_B. \quad (4.15)$$

Because the norm of  $\mathbf{W}_{Bt}^T$  is minimal, when  $\mathbf{X}_B$  is interfered, its influence to  $\mathbf{s}_{Bt}^T$  is also least. Thus better robustness can be gained.

Let  $\mathbf{c}_{Bt}^T = \text{DCT}(\mathbf{s}_{Bt}^T)$ , that is,  $\mathbf{c}_{Bt}^T$  is the DCT coefficient vector of  $\mathbf{s}_{Bt}^T$ . Then its intermediate frequency coefficients are selected as the embedding domain. Assume the watermark  $\mathbf{w}$  is a random sequence with value  $\pm 1$  and length  $l$ , that is,  $\mathbf{w} = \{w(i) | w(i) \in \{-1, +1\}, i = 1, \dots, l\}$ . If the watermarked DCT coefficient vector is  $\mathbf{c}_{wt}^T$ , then the embedding process can be written as

$$c_{wt}(K+i) = \text{sign}(w(i)) \cdot |c_{Bt}(K+i)|, \quad i = 1, \dots, l, \quad (4.16a)$$

$$\mathbf{s}_{wt}^T = \text{IDCT}(\mathbf{c}_{wt}^T), \quad (4.16b)$$

where  $K$  denotes the start position of embedding and is saved as a key.  $\mathbf{s}_{wt}^T$  is the marked IFC which is the inverse DCT of  $\mathbf{c}_{wt}^T$ .  $\text{sign}(\cdot)$  is an operator defined as

$$\text{sign}(x) = \begin{cases} +1, & x \geq 0; \\ -1, & x < 0. \end{cases} \quad (4.17)$$

Finally, let  $\mathbf{S}_w = [\mathbf{s}_{w1}^T, \dots, \mathbf{s}_{wt}^T, \dots, \mathbf{s}_{wM}^T]^T$ , and through

$$\mathbf{X}_w = \mathbf{W}_B^{-1} \mathbf{S}_w, \quad (4.18)$$

the marked image block can be obtained. And combining them together, the marked image is obtained.

## Watermarking Extraction

Corresponding to the embedding, during extraction, the to-be-detected image  $\mathbf{I}_R$  should be divided into blocks, and its IFCs  $\mathbf{S}_R = [s_{R1}^T, s_{R2}^T, \dots, s_{RM}^T]^T$  are calculated through ICA.

$$\mathbf{S}_R = \mathbf{W}_R \mathbf{X}_R, \quad (4.19)$$

Then, according to  $\|\mathbf{W}_{Rk}^T\|_2 = \min_{1 \leq i \leq M} \|\mathbf{W}_{Ri}^T\|_2$ , the may-be-watermarked IFC  $s_{Rk}^T$  would be found. Performed DCT,  $s_{Rk}^T$  is transformed into  $c_{Rk}^T$ . Using  $K$ , the may-be-marked intermediate frequency coefficients could be defined to extract the watermark  $\mathbf{w}'$ ,

$$w'(i) = \text{sign}(c_{Rk}(K+i)) \cdot 1, \quad i = 1, \dots, L. \quad (4.20)$$

Under the common image manipulations,  $\mathbf{I}_R \approx \mathbf{I}_w$ , where  $\mathbf{I}_w$  denotes the watermarked image. Hence,  $\mathbf{I}_R$  and  $\mathbf{I}_w$  can be regarded as different mixtures of the same IFCs. However, when copy attack exists,  $\mathbf{I}_R$  and  $\mathbf{I}_w$  are obviously different, so do their IFCs. That means the watermark can be extracted from the usual attacked images not the copy attacked images.

If  $\mathbf{W}_B$  and  $t$  obtained in the embedding process is assisted to perform extraction, better synchronization can be kept and higher robustness can be obtained. But they are not used here, because the following factors:

- (a) using  $\mathbf{W}_B$  and  $t$  can endanger the security of the algorithm, and
- (b) under copy attack, because of the using of  $\mathbf{W}_B$  and  $t$ , the information about the watermark is introduced to the illegal marked image. Consequently the watermark can be extracted well from the copy attacked image, resulting in the invalidity of algorithm.

To measure the similarity between the original watermark  $\mathbf{w}$  and the extracted watermark  $\mathbf{w}'$ , normalized cross-correlation (NC) is used as the objective evaluation criterion, which is defined as

$$\text{NC} = \frac{\sum_{i=1}^m \sum_{j=1}^n \mathbf{w}(i, j) \cdot \mathbf{w}'(i, j)}{\sqrt{\sum_{i=1}^m \sum_{j=1}^n \mathbf{w}^2(i, j) \sum_{i=1}^m \sum_{j=1}^n \mathbf{w}'^2(i, j)}}. \quad (4.21)$$

A threshold is give according to the false positive probability derived in [48] as Eq. (4.22).

$$p_{fp} = \frac{\text{erf}\left(\frac{1-\mu}{\sqrt{2}\sigma}\right) - \text{erf}\left(\frac{Th-\mu}{\sqrt{2}\sigma}\right)}{\text{erf}\left(\frac{1+\mu}{\sqrt{2}\sigma}\right) + \text{erf}\left(\frac{1-\mu}{\sqrt{2}\sigma}\right)}, \quad (4.22)$$

where

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

is the error function. The threshold for NC is 0.1, which can ensure that the false positive probability is about  $1.7 \times 10^{-5}$ .

Experiments

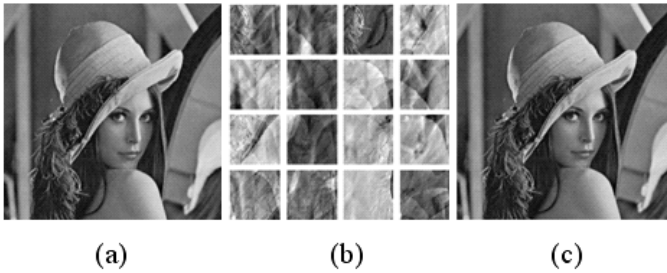


Fig. 4.12. (a) the host image, Lena, (b) the IFCs, and (c) the watermarked image.

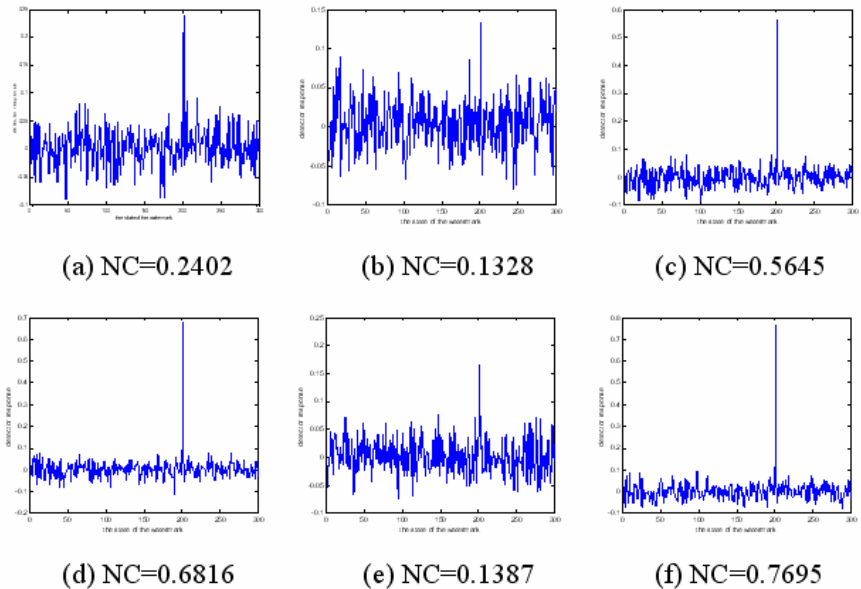


Fig. 4.13. Extraction results under common image manipulations.

Fig. 4.12 shows the host image, Lena, with size of  $256 \times 256$ , the IFCs of Lena with size of  $64 \times 64$ , and the watermarked image. To test the robustness of the watermark, 300 pseudo-random sequences with length of 1024 and value  $\pm 1$  are generated, and the 200th sequence is selected as the watermark.

Fig. 4.13 shows the extraction results under common image manipulations, such as (a) Gaussian noise with mean 0.1 and variance 0.01, (b)  $3 \times 3$  median filtering, (c) cropping the last 20 rows of "Lena" image, (d) JPEG compression (Compression rate = 9.393), (e) using the "bilinear" to extend the image as the fourth as the



original image first, and then resize it as the same as the original image, and (f) histogram equalization. It indicates that under above attacks the watermark can still be extracted successfully from the attacked images.

Several other watermarking schemes based on ICA block features are reported in the literatures [40, 46].

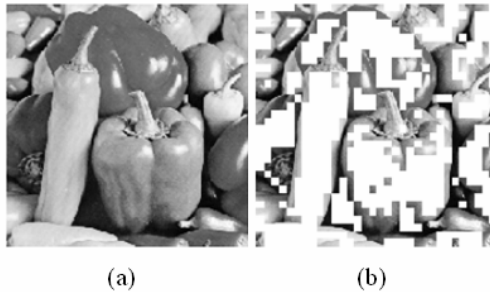
### 4.3.3 Watermarking Based on Independent Downsampling Feature

The independent component features described in Sec. 4.2.1 are used for watermark embedding. Among the extracted features, the low frequency component *F14* in Fig. 4.4 is selected to be watermarked, which has the largest variance, and is considered to embody more information. The scheme in [45] is presented in the following.

#### Watermark Embedding

As we all know, human visual system is more sensitive to the distortion in the smooth region than the rough, so the watermark should be embedded into the rough region to obtain better fidelity. *F14* is divided into blocks with the size of  $8 \times 8$ , and rough blocks are selected to embed watermark while the others are remained. Because the rough block has more AC energy than the others, and most of the energy is in the low-median frequency coefficients, so the rough block can be selected by comparing the L2-norm of these low and mid frequency coefficients.

In order to select the proper AC coefficients to determine the rough blocks, DCT coefficients of all blocks are analyzed. At first, the DCT coefficients are zigzag scanned, and the energy  $\|\mathbf{v}\|_2^2$  of the first 16 coefficients  $\mathbf{v}$  are calculated for each rough block with greater AC energy. Assuming the median of these values is  $m$ , the block is taken as rough one if its energy is larger than the median, otherwise the block is looked as the smooth one. According to the rule, Fig. 4.14 shows the selection results, where white blocks stand for the smooth ones, and the others are the selected rough ones.



**Fig. 4.14.** Rough blocks selection from *F14*. (a) *F14*, and (b) the selection result.

For each selected block, one bit of the watermark is embedded into its low and mid frequency DCT coefficients to obtain the watermarked coefficients  $v_w$  by using DC-QIM scheme as follows [49]:

Quantization [50]:

$$q = \begin{cases} \Delta \cdot \text{round}(\|v\|_2^2 / \Delta), & \text{for case1,} \\ \Delta \cdot \text{round}(\|v\|_2^2 / \Delta + 1), & \text{for case2,} \\ \Delta \cdot \text{round}(\|v\|_2^2 / \Delta - 1), & \text{for case3,} \end{cases} \quad (4.23)$$

where

$$\text{case 1 : } \text{round}(\|v\|_2^2 / \Delta) \% 2 = w;$$

$$\text{case 2 : } \text{round}(\|v\|_2^2 / \Delta) \% 2 \neq w \& \|v\|_2 \geq \Delta \cdot \text{round}(\|v\|_2^2 / \Delta);$$

$$\text{case 3 : } \text{round}(\|v\|_2^2 / \Delta) \% 2 \neq w \& \|v\|_2 < \Delta \cdot \text{round}(\|v\|_2^2 / \Delta).$$

The quantized value  $q$  is adjusted according to

$$q' = \begin{cases} (q + m + \Delta/2)/2 & \text{if } (q < m) \& (m < q + \Delta/2) \\ q + 7\Delta/4 & \text{if } (q < m) \& (m \geq q + \Delta/2) \\ q & \text{others.} \end{cases} \quad (4.24a)$$

$$\beta = \begin{cases} 0 & \text{if } (q < m) \& (m < q + \Delta/2) \\ 0.2 & \text{others.} \end{cases} \quad (4.24b)$$

The final obtained watermarked DCT coefficients  $v_w$ :

$$\begin{aligned} v_w &= v + \alpha v + \beta(v - (v + \alpha v)) \\ &= v + \alpha(1 - \beta)v \end{aligned} \quad (4.25)$$

where  $\alpha = (\sqrt{q} - \|v\|_2) / \|v\|_2$ ,  $w$  is a bit of the watermark  $\mathbf{w}$ ,  $\text{round}$ ,  $\%2$  and  $\|\cdot\|_1$  denote rounding, modulo 2 and  $L^2$ -norm operator, respectively.

The watermarked  $FI_{4w}$  can be reconstructed with combining the watermarked blocks and the original ones through 2D inverse DCT, and then the watermarked sub-image can be obtained. The watermarked image  $\mathbf{I}_w$  can be interpolated by these sub-images.

## Watermark Detection

The watermarked blocks are obtained just as what is done in embedding, and the detected watermark  $w'$  can be extracted as following:

$$w' = \text{round}(\|\mathbf{v}'\|_1 / \Delta) \% 2, \quad (4.26)$$

where  $\mathbf{v}'$  is the watermarked low and median frequency DCT coefficients of the selected block,  $w'$  is one bit of  $\mathbf{w}'$ .

To measure the similarity between the original watermark  $\mathbf{w}$  and detected watermark  $\mathbf{w}'$ , normalized cross-correlation is used as the objective evaluation criterion, defined as:

$$NC = \frac{\sum_{i=1}^L (2\mathbf{w}(i) - 1) \cdot (2\mathbf{w}'(i) - 1)}{\sqrt{\sum_{j=1}^L (2\mathbf{w}(j) - 1)^2 \sum_{j=1}^L (2\mathbf{w}'(j) - 1)^2}}, \quad (4.27)$$

where  $L$  is the length of the watermark. The watermark is considered to be exist in the testing image if the NC is large than the threshold  $Th$ . A failure probability is defined as in [9] for determining the threshold. Here, the threshold is set to 0.2, which means the failure probability is  $p_{fp} < 2.6 \times 10^{-6}$ .

## Experiments

In the experiments, the standard peppers image with size of  $512 \times 512$  is used as the host image. The watermark is a pseudo-random sequence, whose length is  $32 \times 32$  and state is 100. Block selection is based on the L2-norm of the low and mid coefficients of the block. The bit of the watermark at the same position with the rough block is embedded into this rough block.

In order to measure the robustness of the proposed watermarking scheme, the watermarked image is compressed by JPEG with different quality factors, and then watermark is detected from them. Fig. 4.15 describes the detection results after JPEG compression. The  $x$ -coordinate denotes the quality factor of JPEG compression, and  $y$ -coordinate shows the NCs. The results show that the proposed scheme is very robust to the JPEG compression. When the quality factor of JPEG compression isn't smaller than 15, the similarity is higher than 0.9, and it is 1 when the quality factor isn't smaller than 35. Fig. 4.16 shows the watermark detecting results of watermarked image attacked by collusion with 3 watermarked images averaging. These 3 watermarked images are watermarked with different watermarks. The 3 NCs are all above 0.2 demonstrates the watermarking scheme is robust to collusion attack. The  $x$ -coordinate denotes the state of pseudo-random sequence, and  $y$ -coordinate denotes NC.

Table 4.1 lists the detection NC of some common image processing and watermark attacks. The results show that the proposed scheme is robust to the listed image processing and attacks.

Besides the scheme presented above, the schemes described in [37] [38] are also based on the idea of downsampling ICA.

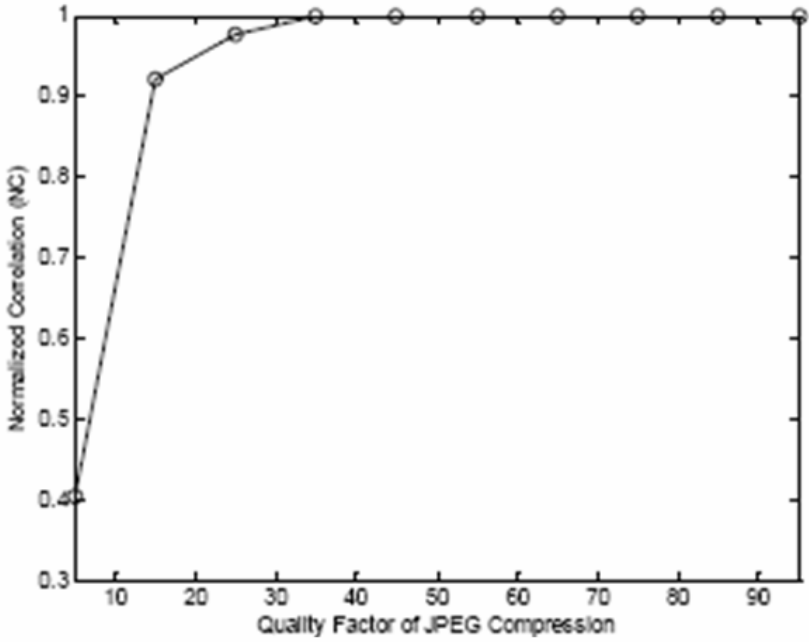


Fig. 4.15. Detection results after JPEG compression.

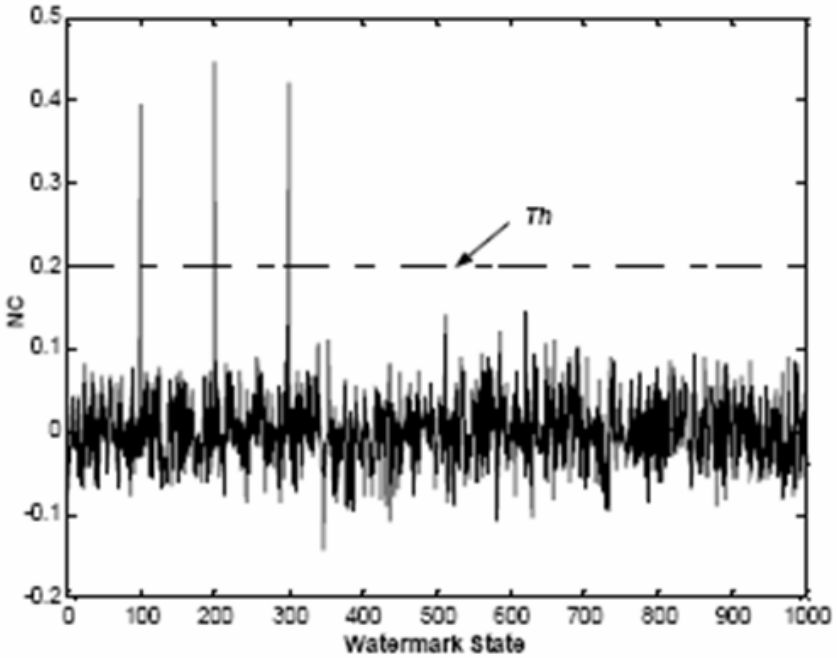


Fig. 4.16. Detection results after collusion attack.

**Table 4.1.** Detection NCs of some common image processing and watermark attacks.

|                  | Operation                    | NC     |
|------------------|------------------------------|--------|
| Noise Attack     | Gaussian (0, 0.003)          | 0.7969 |
|                  | Gaussian (0, 0.005)          | 0.6680 |
|                  | Pepper & Salt (0.005)        | 0.9648 |
|                  | Pepper & Salt (0.01)         | 0.8906 |
| Filtering Attack | Median (3×3)                 | 0.9766 |
|                  | Gaussian (3×3)               | 0.9883 |
|                  | Average (3×3)                | 0.6445 |
| Scaling Attack   | 2                            | 1.0000 |
|                  | 1/2                          | 0.6953 |
| Removing Attack  | 1/4                          | 0.6523 |
| Image Processing | Histogram Equalization       | 0.3594 |
|                  | Increasing Luminance [0.4-1] | 1.0000 |
|                  | Descending Luminance [0-0.8] | 1.0000 |

## 4.4 ICA-Based Video Watermark Schemes

### 4.4.1 Watermarking Based on Independent Video Block Feature [35]

The independent video block features are obtained as described in Sec. 4.2.2 and used as the watermark embedding domain. With regard to MPEG compression standard, GOP is adopted as the unit for dividing video into 3D blocks. And these 3D blocks are processed by ICA to get ICs.

#### Watermark Embedding

The frames of ICs are called slices here. The slices of the same order in their own ICs are collected, among which the one that has the maximum variance is selected to be embedded watermark. A pseudo-random 0, 1 sequence with special seed is used as watermark  $w_n$  to embed into the wavelet domain of the selected slices. All to-be-watermarked slices are watermarked in the same way. The 4-neighboring-mean-based algorithm presented in [51] is used as the embedding method.

The embedding algorithm is:

$$p'_{i,j} = p_{i,j} + \alpha(p_{i,j} - mp_{i,j}), \mathbf{F}_n(m) = 1 \begin{cases} p_{i,j} > mp_{i,j}, \mathbf{w}(m) = 0 \\ p_{i,j} < mp_{i,j}, \mathbf{w}(m) = 1 \end{cases} \quad (4.28a)$$

$$p'_{i,j} = p_{i,j} - \alpha(mp_{i,j} - p_{i,j}), \mathbf{F}_n(m) = 0 \begin{cases} p_{i,j} > mp_{i,j}, \mathbf{w}(m) = 1 \\ p_{i,j} < mp_{i,j}, \mathbf{w}(m) = 0 \end{cases} \quad (4.28b)$$

where  $p_{i,j}$  is the to-be-watermarked coefficient, and

$$mp_{i,j} = \frac{1}{4}(p_{i,j-1} + p_{i-1,j} + p_{i,j+1} + p_{i+1,j})$$

is the mean of its 4 neighboring coefficients, which will not be watermarked.  $m = 1, 2, \dots, r$ ,  $r$  is the length of watermark sequence,  $n$  is the slice order in the video ICs,  $\alpha$  is a weight.  $\mathbf{F}_n$  is a symbol to get the extracted watermark.

### Watermark Detection

When detecting watermark, the independent components of received video are gotten and the watermarked slices are picked out just like the embedding process. The detecting algorithm is:

$$\mathbf{w}_n^*(m) = 0 \begin{cases} p_{i,j}^* > mp_{i,j}^*, \mathbf{F}_n(m) = 0 \\ p_{i,j}^* < mp_{i,j}^*, \mathbf{F}_n(m) = 1 \end{cases} \quad (4.29a)$$

$$\mathbf{w}_n^*(m) = 1 \begin{cases} p_{i,j}^* > mp_{i,j}^*, \mathbf{F}_n(m) = 1 \\ p_{i,j}^* < mp_{i,j}^*, \mathbf{F}_n(m) = 0 \end{cases} \quad (4.29b)$$

where  $p_{i,j}^*$  and  $mp_{i,j}^*$  are the watermarked coefficient and its neighboring mean of the selected slices, respectively.  $\mathbf{w}_n^*$  is the extracted watermark of the slice  $n$ . The similarity between the extracted watermark and the original one is defined:

$$\text{Sim}(\mathbf{w}_n, \mathbf{w}_n^*) = 1 - \frac{\sum_{m=1}^r \text{XOR}[\mathbf{w}_n(m), \mathbf{w}_n^*(m)]}{r}. \quad (4.30)$$

A threshold is set to determine whether the extracted watermark is the embedded one. If the similarity exceeds the threshold, the slice is considered to be watermarked. If the mean of similarities of all slices exceeds the threshold, the video is considered to be watermarked.

### Experiments

The experimental video consists of 16  $256 \times 256$  frames, two of which are shown in Fig. 4.17. Each GOP is divided into 16  $64 \times 64 \times 12$  sub-videos, and 12 is the number of frames in each sub-video. 200 is the seed to generate a pseudo-random 0, 1 sequence to be the watermark. The threshold is 0.7. The video in the following two

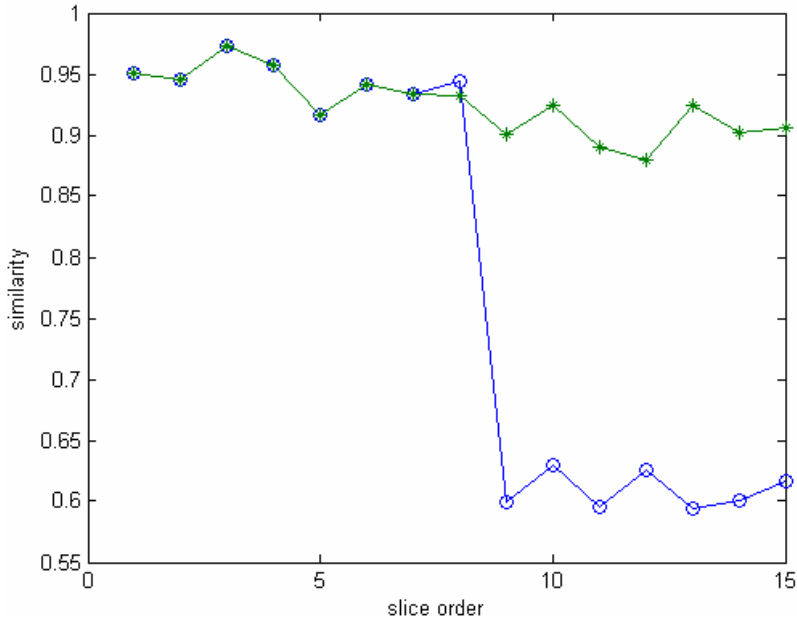


Fig. 4.17. Two frames of the experimental video.

experiments is decompressed after MPEG-2 compression. The detection results of temporal synchronization and collusion attacks experiments are shown in Fig. 4.18 and Fig. 4.19 respectively, in which the slice order is the order of the selected slice in its own IC and similarity measures whether the extracted watermark is the embedded one.

In Fig. 4.18, the  $-o-$  curve shows the detection results before synchronization, from which it can be estimated that the  $9^{th}$  frame of original video is the first dropped frame. The  $-*-$  curve shows the synchronization results. The first 8 points of the two curves are superposition, and the latter of the  $-*-$  curve show the extracted watermarks are the embedded ones. From both of the curves it can be seen that only the  $9^{th}$  frame is dropped. The experiment shows the synchronization can be achieved step by step whichever frame is dropped, and however many frames are dropped.

There are two types of collusion attacks. The type-I is the intra-video collusion. It is for similar frames watermarked different. The inter-video collusion is the type-II, which is for several differently watermarked copies of the same video. In Fig. 4.19, the lower  $-<-$  curve is the detection results of intra-video collusion and the upper four curves are the results of inter-video collusion, where  $-+-$ ,  $-o-$ ,  $-*-$  and  $-●-$  denote the collusion with seed 100, 150, 200, and 250 respectively. The intra-video collusion is performed by averaging consecutive frames. The points of  $-<-$  curve are almost all above 0.7, and their mean is 0.7625, which shows the detection is affected by the intra-video collusion, but the embedded watermark can still be extracted. For inter-video collusion experiments, 4 copies of the video are watermarked by different sequences, whose seeds are 100, 150, 200 and 250, respectively. The average of them is the detected video. All the points of the upper four curves are above 0.7 and the means of each curve are 0.9191, 0.9223, 0.9238 and 0.9288,



**Fig. 4.18.** Detection results comparison before and after temporal resynchronization. —○— detection before synchronization, and —\*— detection after synchronization.

respectively. Fig. 4.19 shows the watermark is robust to the two types of collusion attacks.

#### 4.4.2 Watermarking Scheme Based on Independent Dynamic Feature [22]

Compared with image watermarking, the issue of temporal synchronization is specific for video watermarking, and it is also essential one that has to be resolved first in video watermarking. Referring to MPEG compression standard, in order to embed the temporal synchronization information during watermark embedding, the 3-level wavelet approximate sub-image of the I-frame in a GOP is binarized to distinguish different GOPs [52], which is called GOP signature and is used as the watermark of this GOP.

The watermark of a GOP can be obtained by:

$$[\mathbf{LL}_{1,k}, \mathbf{LH}_{1,k}, \mathbf{HL}_{1,k}, \mathbf{HH}_{1,k}] = \text{DWT}(\mathbf{F}_{k,1}), \quad (4.31a)$$

$$[\mathbf{LL}_{2,k}, \mathbf{LH}_{2,k}, \mathbf{HL}_{2,k}, \mathbf{HH}_{2,k}] = \text{DWT}(\mathbf{LL}_{1,k}), \quad (4.31b)$$

$$[\mathbf{LL}_{3,k}, \mathbf{LH}_{3,k}, \mathbf{HL}_{3,k}, \mathbf{HH}_{3,k}] = \text{DWT}(\mathbf{LL}_{2,k}), \quad (4.31c)$$

$$\mathbf{w}_k = \text{BIN}_{median}(\mathbf{LL}_{3,k}), \quad (4.32)$$



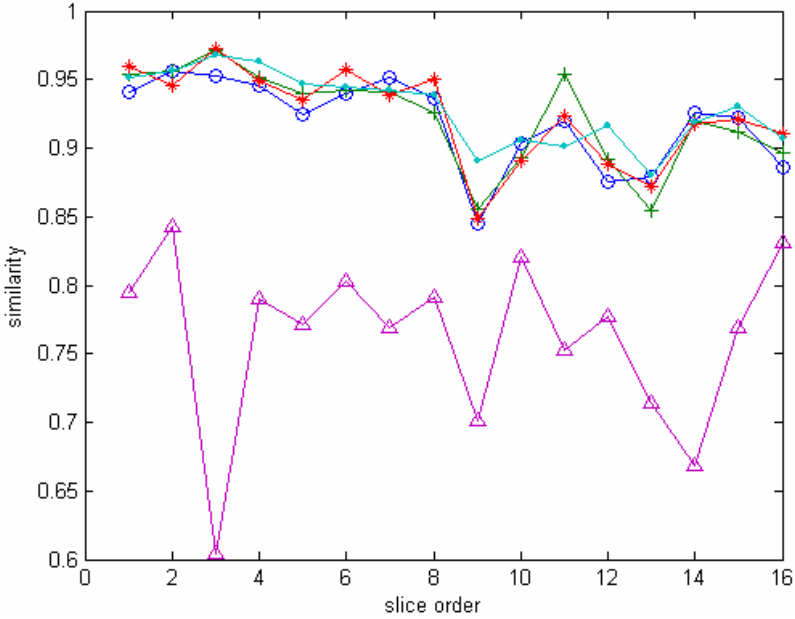


Fig. 4.19. Detection results after two kinds of collusion attacks.

where  $k$  is the order of GOP in video,  $\mathbf{F}_{k,1}$  is the I-frame of the  $k^{\text{th}}$  GOP, denotes 1-level discrete wavelet transform,  $\mathbf{LL}_{i,k}$  is the  $i$ -level approximate sub-image,  $\mathbf{LH}_{i,k}$ ,  $\mathbf{HL}_{i,k}$ ,  $\mathbf{HH}_{i,k}$  are respectively the  $i$ -level x-direction, y-direction and diagonal sub-images.  $BIN_{median}$  denotes binarization based on median, the watermark of the  $k^{\text{th}}$  GOP,  $\mathbf{w}_k$ , is a  $r \times c$  matrix.

The difference of different GOPs is defined by

$$D_G = \frac{H_D}{S_E}, \quad (4.33)$$

where  $H_D$  is the hamming distance between two GOP signatures.  $S_E$  is the total number of the elements in a GOP signature. According to randomness, the  $D_G$  of different GOP signatures is around 0.5. In the test, 1142 video frames of Bears, Earthquake, Volcano subjects of Discovery series are selected to extract their GOP signatures. The  $D_G$  histogram is shown in Fig. 4.20.

The mean of  $D_G$  is 0.519 and the variance is 0.0048. The results demonstrate the mean of  $D_G$  is close to 0.5 and gathers into a small area centered on 0.519. So the GOP signature can be used to distinguish two different GOPs.

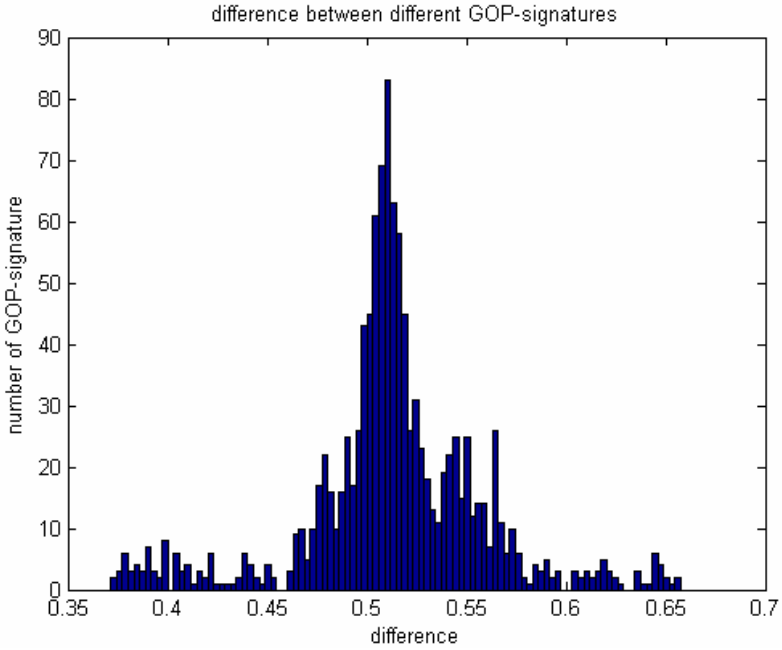


Fig. 4.20. The  $D_G$  histogram of 1142 GOP signatures.

## Watermark Embedding

According to the Sec. 4.2.2,  $\mathbf{s}_{F_S}$  and  $\mathbf{s}_{M_F}$  are gotten through FastICA:

$$\begin{bmatrix} \mathbf{s}_{F_S} \\ \mathbf{s}_{M_F} \end{bmatrix} = \mathbf{A}^{-1} \begin{bmatrix} \mathbf{x}_{F(i)} \\ \mathbf{x}_{F(i-1)} \end{bmatrix}. \quad (4.34)$$

And  $\mathbf{s}_{F_S}$  and  $\mathbf{s}_{M_F}$  can be obtained. And then  $\mathbf{s}_{M_F}$  is 2-level discrete wavelet transformed:

$$[\mathbf{LL}_1, \mathbf{LH}_1, \mathbf{HL}_1, \mathbf{HH}_1] = \text{DWT}[\mathbf{s}_{M_F}], \quad (4.35a)$$

$$[\mathbf{LL}_2, \mathbf{LH}_2, \mathbf{HL}_2, \mathbf{HH}_2] = \text{DWT}[\mathbf{LL}_1]. \quad (4.35b)$$

Here,  $\mathbf{LL}_i, \mathbf{LH}_i, \mathbf{HL}_i, \mathbf{HH}_i$  are respectively the  $i$ -level approximate, x-direction, y-direction and diagonal sub-images. In the 2-level approximate sub-image  $\mathbf{LL}_2$ , a random position sequence  $\text{Ind}_l$ , whose length is  $L = r \times c$ , is generated for each  $\mathbf{s}_{M_F}$  in a GOP. Its subscript  $l$  is the order of  $\mathbf{s}_{M_F}$  in GOP. The watermark is embedded into the same position in the  $\mathbf{s}_{M_F}$  of the same order in its own GOP.

The watermark is embedded into the 2-level DWT approximate sub-image of  $\mathbf{s}_{M_F}$  according to 4-neighboring-mean based algorithm [51]. Fig. 4.21 shows the whole embedding procedure.

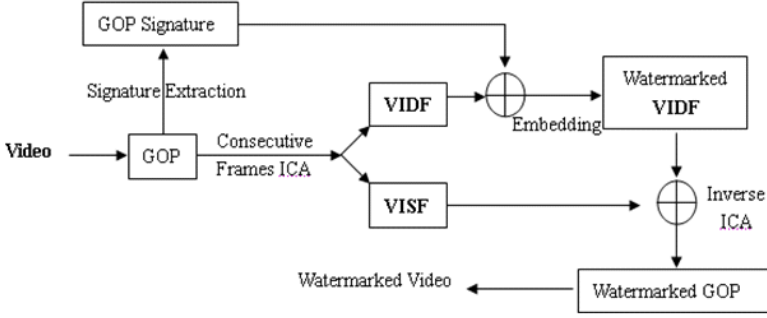


Fig. 4.21. Watermark Embedding.

Suppose that the coefficient  $p_{i,j} \in \mathbf{LL}_2$  is watermarked, the mean of its four neighboring coefficients is

$$mp_{i,j} = \frac{1}{4}(p_{i,j-1} + p_{i-1,j} + p_{i,j+1} + p_{i+1,j}),$$

and the four neighboring coefficients are not watermarked. For the  $l^{\text{th}}$   $\mathbf{S}_{M_F}$  of a GOP, its embedding-position sequence is  $Ind_l$ . The embedding algorithm is:

$$p'_{i,j} = p_{i,j} + \alpha(p_{i,j} - mp_{i,j}), \mathbf{F}_{k,l}(m,n) = 1 \begin{cases} p_{i,j} > mp_{i,j}, & \mathbf{w}_k(m,n) = 0 \\ p_{i,j} < mp_{i,j}, & \mathbf{w}_k(m,n) = 1 \end{cases} \quad (4.36a)$$

$$p'_{i,j} = p_{i,j} - \alpha(mp_{i,j} - p_{i,j}), \mathbf{F}_{k,l}(m,n) = 0 \begin{cases} p_{i,j} > mp_{i,j}, & \mathbf{w}_k(m,n) = 1 \\ p_{i,j} < mp_{i,j}, & \mathbf{w}_k(m,n) = 0 \end{cases} \quad (4.36b)$$

where  $(i,j) \in Ind_l$ ,  $m = 1, 2, \dots, r$ ,  $n = 1, 2, \dots, c$ ,  $\alpha$  is a weight.  $\mathbf{F}_{k,l}$  is a symbol to determine the existence of the embedded watermark.  $\mathbf{F}_{k,l}(m,n)$  is the element in  $m$  row  $n$  column of  $\mathbf{F}_{k,l}$ .

## Watermark Extraction

Before watermarking extraction, which frame of the original video is the first one of the received/detected video has to be known. After this, the watermark is temporarily extracted as if there is no temporal desynchronization, e.g. frame dropping, frame swapping, frame averaging. According to the original result, it can be judged that which case of temporal desynchronization happened, and the video is resynchronized temporally to get the correct extraction.

In this scheme, an improvement on watermark extraction is made, because the scheme in [51] can not achieve blind extraction. The similarity between the original symbol  $\mathbf{F}_{k,l}$  and the extracted one  $\mathbf{F}_{k,l}^*$  is used to measure whether there is the watermark or not. In this way, the original GOP-signature  $\mathbf{w}_k$  and the symbol  $\mathbf{F}_{k,l}$  need

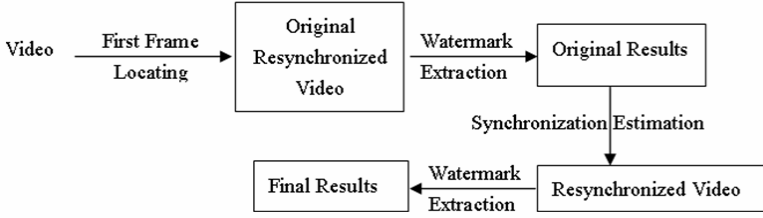


Fig. 4.22. Watermark extraction.

not be stored at the same time. Only  $\mathbf{F}_{k,l}$  need to be stored. The GOP-signature  $\mathbf{w}_k^*$  is gotten from the received video just as in embedding. And then the extracted symbol matrix  $\mathbf{F}_{k,l}^*$  is obtained.

$$\mathbf{F}_{k,l}^*(m,n) = 1 \begin{cases} p_{i,j}^* > mp_{i,j}^*, & \mathbf{w}_k^*(m,n) = 0, \\ p_{i,j}^* < mp_{i,j}^*, & \mathbf{w}_k^*(m,n) = 1, \end{cases} \quad (4.37a)$$

$$\mathbf{F}_{k,l}^*(m,n) = 0 \begin{cases} p_{i,j}^* > mp_{i,j}^*, & \mathbf{w}_k^*(m,n) = 1, \\ p_{i,j}^* < mp_{i,j}^*, & \mathbf{w}_k^*(m,n) = 0. \end{cases} \quad (4.37b)$$

Here,  $p_{i,j}^*$ ,  $mp_{i,j}^*$  are the watermarked wavelet coefficient and its 4-neighboring mean, respectively.

The similarity between the original symbol  $\mathbf{F}_{k,l}$  and  $\mathbf{F}_{k,l}^*$  the extracted one is:

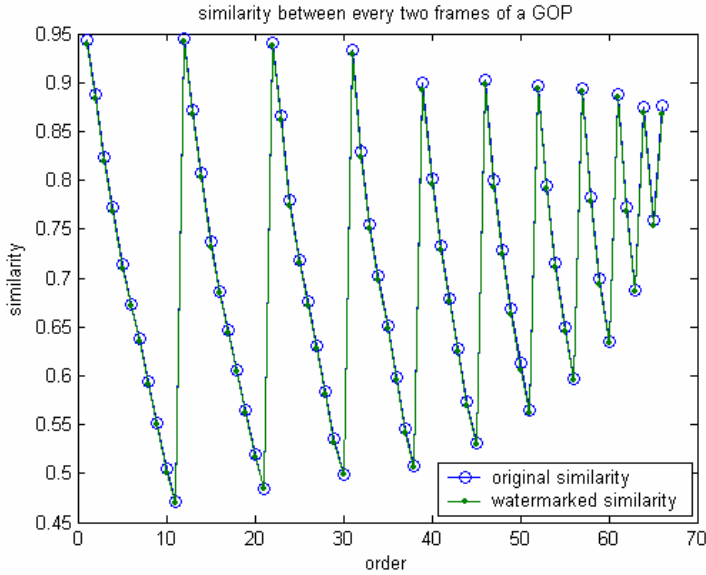
$$\text{Sim}(\mathbf{F}_{k,l}, \mathbf{F}_{k,l}^*) = 1 - \frac{\sum_{m=1}^r \sum_{n=1}^c \text{XOR}[\mathbf{F}_{k,l}(m,n), \mathbf{F}_{k,l}^*(m,n)]}{r \times c}. \quad (4.38)$$

A detecting threshold should be set. If the similarity exceeds the threshold, there is the watermark in this frame. And if the similarity mean of the frames in the received video exceeds the threshold, the video is the legal one and it is watermarked by the embedded one, vice versa.

## Experiments

The video used as an example here is 1s with frame rate 30fps. Its frame size is  $352 \times 272$ . And the frames in Fig. 4.8(a) and 4.8(b) are two of the video. The ICA algorithm used here is FastICA [4]. 0.7 is selected as the detecting threshold empirically. The GOP in this experiment consists of one I-frame, three P-frames and two B-frames between the I-frame and a P-frame or between two P-frames, i.e. the structure is IBBPBBPBBPBB.

The quality of the watermarked video is described in terms of PSNR. The minimum of the PSNRs is 31.21 dB, the maximum is 46.82 dB and the mean is 40.89 dB. It indicates that the embedded watermark does not degrade the video quality. And the subjective measure of the video quality is also good enough. The statistical invisibility shows whether the statistical property is affected by the watermark

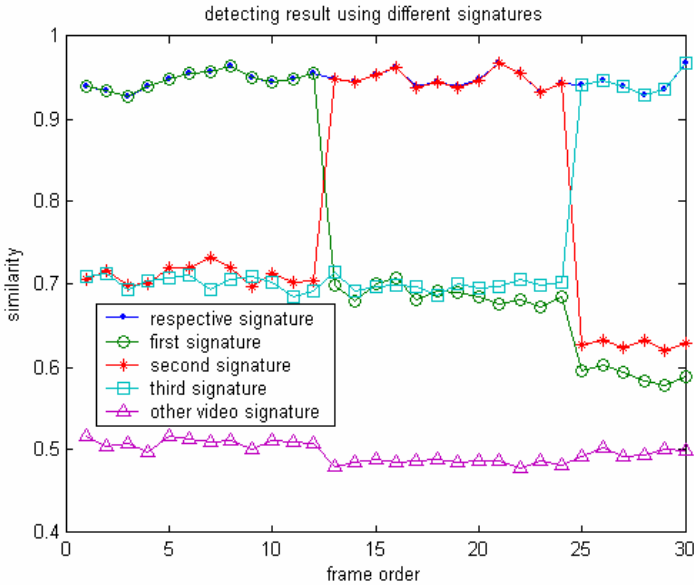


**Fig. 4.23.** Statistical invisibility of a GOP.

embedding. It can be measured by the difference between the correlation of every two original frames and that of the two corresponding watermarked frames [53]. Fig. 4.23 shows the statistical invisibility in a GOP. The mean of differences is 0.002 and its variance is  $3.66 \times 10^{-4}$ . That shows the watermark is statistically undetectable.

After MPEG-2 encoding and decoding, the detection is accomplished by using their respective GOP-signatures, only by the first GOP-signature, only by the second one, only by the third one and by a GOP-signature of another video. The results are in Fig. 4.24. When the watermark is detected by using the incorrect GOP-signatures, the detection curves descend abruptly, and the similarity descends greatly. That shows the watermark can be detected correctly only by using the correct GOP-signatures. The detection curve of using the GOP-signature of other video shows the scheme can resist copy attack. The similarity extracted by using the 1<sup>st</sup>, 2<sup>nd</sup>, 3<sup>rd</sup> GOP-signature is larger than that extracted by using the GOP-signature in another video, while smaller than that extracted by using the respective GOP-signature. That is because of the similarity between the GOP-signature in the same video.

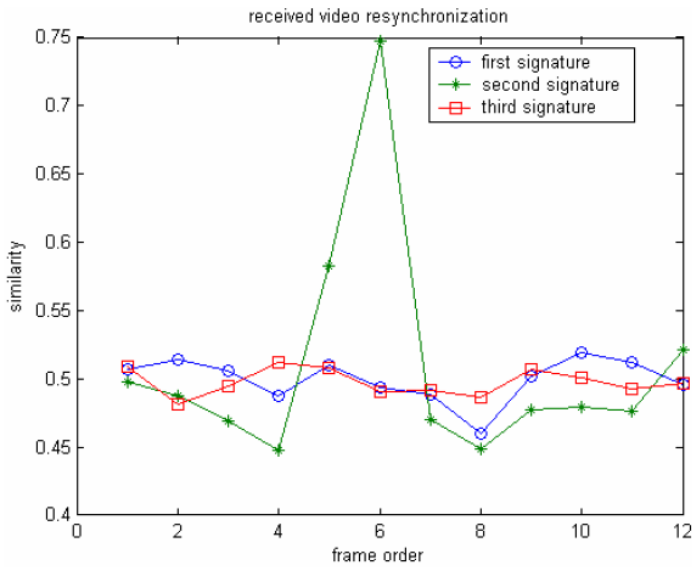
Fig. 4.25 demonstrates the result of location of the first frame in the received video. The first 7 frames are dropped at the receive terminal. In resynchronization, every frame in the first GOP of the received video is treated as the I-frame of a GOP, and then their corresponding GOP-signatures or watermarks are extracted. Thus a group of symbol matrices is achieved. The 3 curves in Fig. 4.25(a) shows the similarities between these symbols and the original symbols. There is a peak in the second GOP-signature detecting curve at the 6<sup>th</sup> frame. That shows the 6<sup>th</sup> of



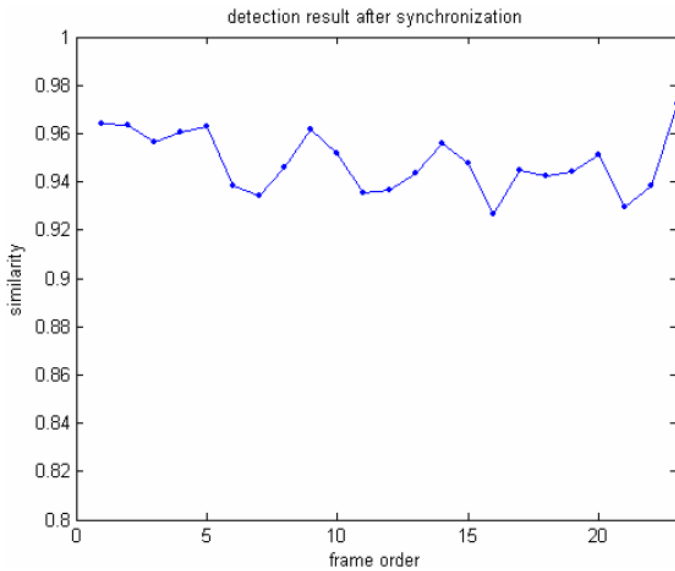
**Fig. 4.24.** Detection results with different GOP-signatures.

the received video is the I-frame of the second GOP in the original video and the received video begins from the 8<sup>th</sup> frame of the first GOP in the original video. So the resynchronization is achieved.

In Fig. 4.26, watermark is detected after fps changing, frame averaging, frame swapping and frame dropping. The total of frames is not changed, when fps changes from 30fps to 24fps. The detection curve shows that if the frame number of a video is not changed during fps changing, the watermark can all be detected correctly. During frame averaging, the 10<sup>th</sup> frame is replaced by the average of the 9<sup>th</sup>, 10<sup>th</sup> and 11<sup>th</sup> frames and the average of the 19<sup>th</sup>, 21<sup>st</sup> and 22<sup>nd</sup> frames is instituted for the 20<sup>th</sup> frame. From the curve, it can be known that if the average frames include the to-be-replaced frame, the similarity is slightly affected. At the same time, the watermark can be detected correctly. The 15<sup>th</sup> and 25<sup>th</sup> frames are exchanged. The corresponding curve drops sharply at the exchanged frames and the detection of the adjacent frames is affected. Because the 25<sup>th</sup> frame is the I-frame of the 3<sup>rd</sup> GOP, the detecting results of this GOP are damaged. The 10<sup>th</sup> frame is dropped in frame dropping experiment. From the detection curve, watermark cannot be detected correctly from the 10<sup>th</sup> frame. So it can be estimated that the 10<sup>th</sup> frame may be dropped. And then, the video is resynchronized according to this estimation. The detection curve is in Fig. 4.26(b). In Fig. 4.26(b), most of the detected results are higher than 0.7. So it can be seen that the 10<sup>th</sup> frame is really dropped, and only the 10<sup>th</sup> frame is dropped. Based on this algorithm, whichever frame is dropped, the video can be resynchronized frame by frame.



(a) Positioning of the first received frame

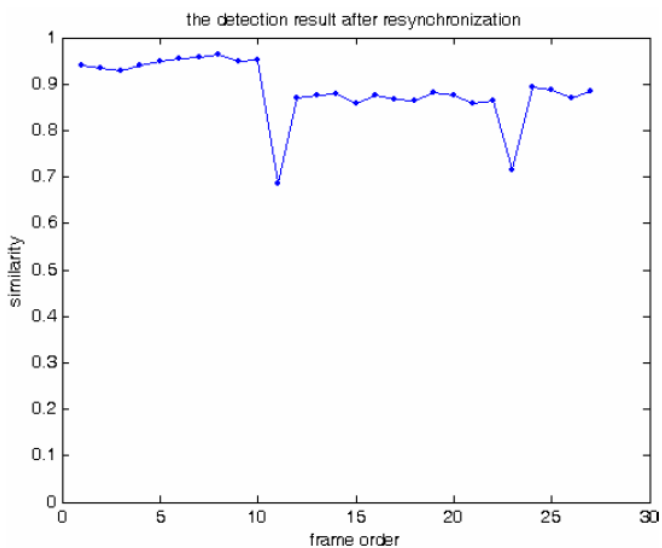


(b) Detection result after synchronization

**Fig. 4.25.** Resynchronization after the first 7 frames dropping.



(a) Detection results after attacks



(b) Resynchronization result after frame dropping

Fig. 4.26. Detection results after temporal desynchronization.



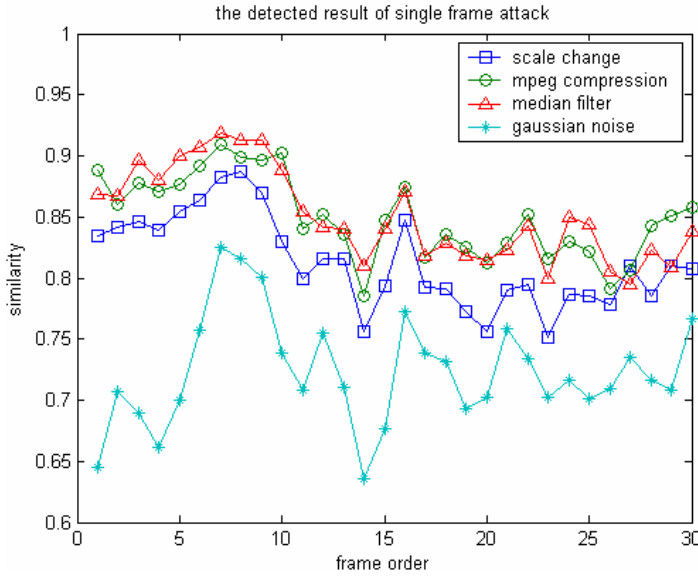


Fig. 4.27. Detection results after single frame attacks.

In Fig. 4.27 every single frame is attacked by common processing for images, which includes  $3 \times 3$  median filtering, scale changing with the factor 2, standard MPEG-2 compression and Gaussian noising whose mean is 0 and variance is 0.03. The detection curves show that the watermark is robust to the former three attacks, while the detection is affected by the Gaussian noise as shown by the lower curve. But the mean of all the similarities in the lower curve is 0.724, it is still over 0.7. That shows the watermark is also detected correctly.

Because the watermark is based on the video content, different watermarked copies of the same video are watermarked same and different videos have different watermarks. That is to say, there is no possibility for the inter-video attack. In a video, the frames in different GOPs are treated as different frames and those in a GOP are treated as similar frames. In this scheme, all of the frames are differently watermarked, i.e. different watermarks or different positions. So there is only a case for the intra-video collusion attack. It is the frames in a GOP that are treated as the same frames differently watermarked. This kind of collusion is often performed by low-pass filter. Here, 1-D temporal wavelet is applied to every four frames of the video and the original frame is replaced by the approximate one. The detection result is shown in Fig. 4.28 after the entire replacement of the frames. By the way, the video used in this experiment is the football.avi video. The mean of the detection result is 0.7892. This experiment indicates that the watermark is robust to the two types of collusion attacks.

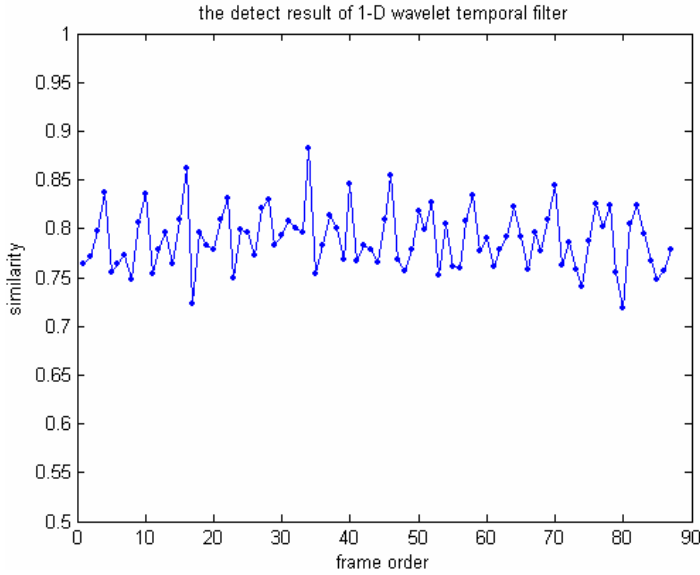


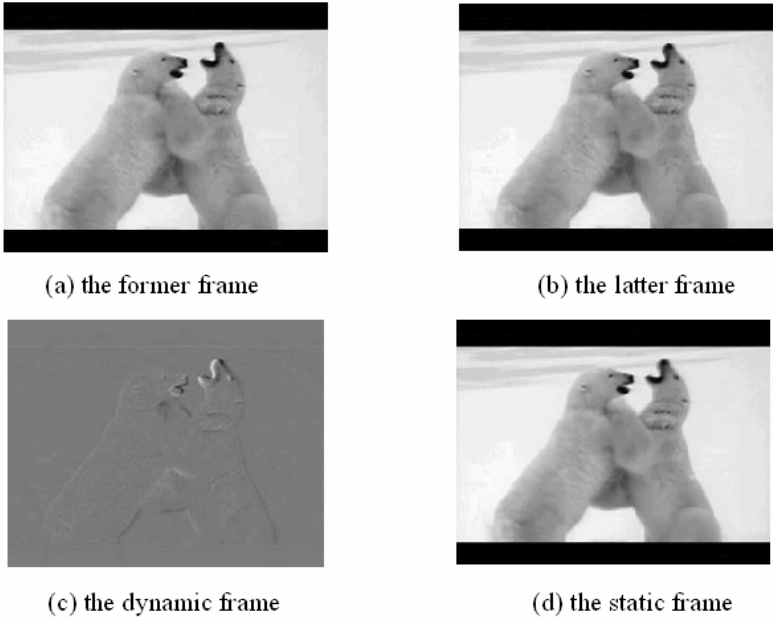
Fig. 4.28. Detection results after intra-video collusion attack.

#### 4.4.3 Watermarking Based on Motion Location in Independent Dynamic Feature [54]

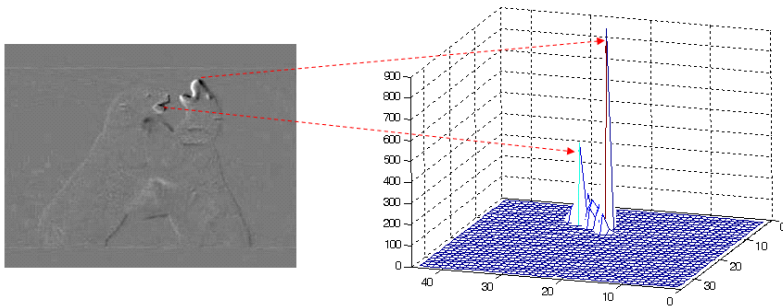
With the development of the technology of video coding, the motive objects and the background of video have been encoded respectively. Such framework is not only suitable for some applications, but also enhances the coding efficiency. Therefore, the extraction of motion information has been essential for further video processing. So as an improvement of the scheme presented in Sec. 4.4.2, the motion regions are located in the independent dynamic features, and watermark is only embedded into these corresponding motion regions of the former frame. Fig. 4.29 shows the independent dynamic features extraction by using ICA.

Because the variance of the region, where the texture is more conspicuous, is much larger, variance of  $8 \times 8$  block is taken as the standard to measure how great the motion is in the dynamic frame. Fig. 4.30 shows the relationship between the variance and the texture. It clarifies that the motion region can be located with the help of variance.

Fig. 4.31 describes the procedure of motion location. Here the motion location is divided into two stages. In the first stage, macro block is used to locate the motive object primarily. The sum of variance (SoV) of each  $8 \times 8$  block in the macro block is used to measure how great the motion is in this macro block. If the SoV of a macro block is great, the macro block is classified as a motion macro block (MMB). Then the MMB which has the largest SoV is selected, and a motion region (MR) centered by this MMB is defined. And the watermark will be embedded into the same place in original frame where the MR is located in the dynamic frame.



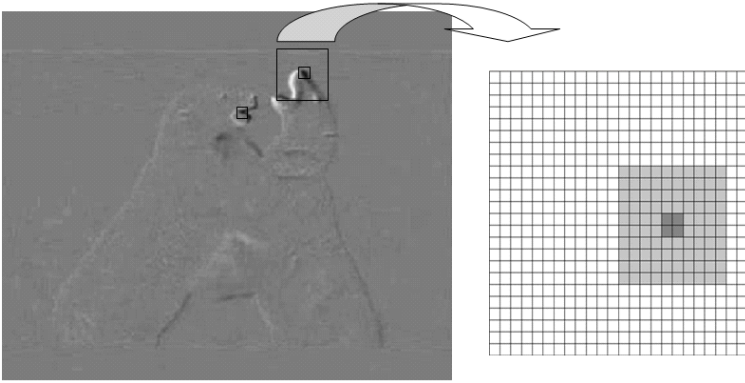
**Fig. 4.29.** The dynamic frame (c) is extracted using ICA from two successive frames ((a) and (b)). And (d) is the static frame.



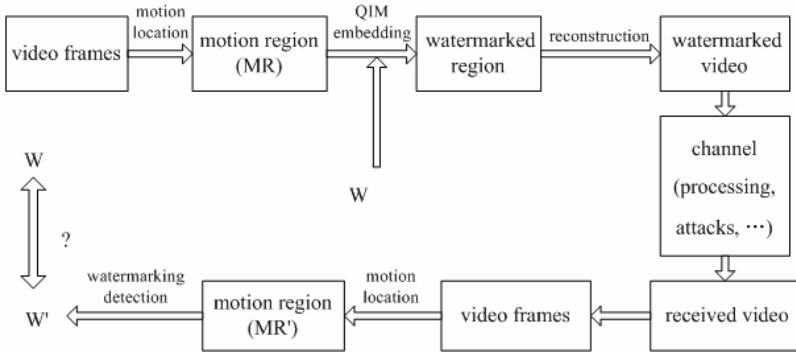
**Fig. 4.30.** The relationship between the variance and the texture.

### Watermark Embedding

Since human eyes are less sensitive to the error in the region with/near great motion, the region where MR is located is selected as watermarking domain. Apart from considering perceptual quality, the Quantization index modulation (QIM) method is used to guarantee the robustness of the proposed scheme. Fig. 4.32 shows the framework. Through comparing the extracted watermark  $w'$  with the embedded watermark  $w$ , one can know whether the digital video is watermarked or not.



**Fig. 4.31.** This figure describes the procedure of locating motion region. The smaller squares stand for the MMBs, while the bigger square is the desired MR.



**Fig. 4.32.** Framework of the scheme.

QIM is a computationally efficient method for watermarking with side information [55, 56]. When embedding a message, QIM firstly modulates an index or sequence of indices with a message to be embedded, and then quantizes the host signal with the associated quantizer or sequence of quantizers. A quantizer is a function that maps a value to the nearest point belonging to a class of pre-defined discontinuous points. Assume that function  $\text{round}(\cdot)$  denotes rounding value to the nearest integer, and  $\mathbf{x}$  is the host signal. Then the standard quantization operation with step size  $\Delta$  can be obtained by

$$Q(\mathbf{x}, \Delta) = \text{round}\left(\frac{\mathbf{x}}{\Delta}\right) \Delta. \tag{4.39}$$

The traditional QIM utilizes a fixed quantization step size, which may lead to poor fidelity in some areas of the content. Therefore, Li and Cox [55] proposed an ameliorative QIM based on perceptual models. The QIM method in our scheme also

**Table 4.2.** The frequency sensitivity  $t(i, j)$  in  $(8 \times 8)$  DCT sensitivity model.

| $t(i, j)$         | $j = 1, \dots, 8$ |      |      |      |       |       |       |       |
|-------------------|-------------------|------|------|------|-------|-------|-------|-------|
|                   | 1.40              | 1.01 | 1.16 | 1.66 | 2.40  | 3.43  | 4.79  | 6.56  |
|                   | 1.01              | 1.45 | 1.32 | 1.52 | 2.00  | 2.71  | 3.67  | 4.93  |
|                   | 1.16              | 1.32 | 2.24 | 2.59 | 2.98  | 3.64  | 4.60  | 5.88  |
|                   | 1.66              | 1.52 | 2.59 | 3.77 | 4.55  | 5.30  | 6.28  | 7.60  |
| $i = 1, \dots, 8$ | 2.40              | 2.00 | 2.98 | 4.55 | 6.15  | 7.46  | 8.71  | 10.17 |
|                   | 3.43              | 2.71 | 3.64 | 5.30 | 7.46  | 9.62  | 11.58 | 13.51 |
|                   | 4.79              | 3.67 | 4.60 | 6.28 | 8.71  | 11.58 | 14.50 | 17.29 |
|                   | 6.56              | 4.93 | 5.88 | 7.60 | 10.17 | 13.51 | 17.29 | 21.15 |

integrates perceptual model for good fidelity. Dissimilarly, only the sensitivity function of Watson's perceptual model [57] is used for calculating quantization step size.

Watson's model is an  $8 \times 8$  block-based perceptual model in DCT domain. It consists of a sensitivity function, two masking components based on luminance and contrast masking respectively, as well as a combine component. Here only the sensitivity function is used for calculating quantization step size.

Sensitivity model defines a frequency sensitivity table  $\mathbf{t}$  [57, 58], which is shown in Table 4.2. For each DCT coefficient  $(i, j)$ , where  $i, j = 1, 2, \dots, 8$ , each table entry  $\mathbf{t}(i, j)$  is approximately the smallest discernible change without any masking noise. Therefore, the step size used to do quantization operation to DCT coefficient  $(i, j)$  is given by

$$\text{step}(i, j) = \beta \times \mathbf{t}(i, j), \quad (4.40)$$

where  $\beta$  is used to control the strength of watermarking. Incorporated with Watson's perceptual model, the QIM method can achieve better fidelity.

During watermarking the original video, all frames of the original video (except the last one) will be watermarked in case of frame dropping or false location in certain frames when detecting. As a consequence, the determined regions for embedding in different frames may be different from each other.

Let  $\mathbf{c}_o(i, j, k)$  denotes the DCT coefficient  $(i, j)$  of the  $k^{\text{th}}$  block,  $\mathbf{c}_w(i, j, k)$  denotes the modified DCT coefficient  $(i, j)$  of the block  $k^{\text{th}}$ , and  $\mathbf{w}$  denotes the binary pseudo-random watermark.

$$\mathbf{q}_e(i, j, k) = \frac{|\mathbf{c}_o(i, j, k)|}{\text{step}(i, j)}, \quad (4.41)$$

$$\mathbf{m}_e(i, j, k) = \text{round}(\mathbf{q}_e(i, j, k)), \quad (4.42)$$

$$\delta = \mathbf{m}_e(i, j, k) - \mathbf{q}_e(i, j, k), \quad (4.43)$$

where  $|\delta| \leq \frac{1}{2}$ .

During watermark embedding,

- Case 1:  $\text{mod}(\mathbf{m}_e(i, j, k), 2) = w$

$$\mathbf{m}_e(i, j, k) = \text{round}(\mathbf{q}_e(i, j, k)). \quad (4.44)$$

- Case 2:  $\text{mod}(\mathbf{m}_e(i, j, k), 2) \neq w$ 
  - if  $\delta \geq 0$

$$|\mathbf{c}_w(i, j, k)| = (\mathbf{m}_e(i, j, k) + 1) \times \text{step}(i, j). \quad (4.45)$$

- if  $\delta < 0$

$$|\mathbf{c}_w(i, j, k)| = (\mathbf{m}_e(i, j, k) - 1) \times \text{step}(i, j). \quad (4.46)$$

### Watermark Detection

During watermark detecting,

$$\mathbf{w}' = \text{mod}(\mathbf{m}_d(i, j, k), 2), \quad (4.47)$$

where  $\mathbf{w}'$  denotes the extracted watermark. And  $\mathbf{m}_d(i, j, k)$  is given by

$$\mathbf{q}_d(i, j, k) = \frac{|\mathbf{c}_w(i, j, k)|}{\text{step}(i, j)}, \quad (4.48)$$

$$\mathbf{m}_d(i, j, k) = \text{round}(\mathbf{q}_d(i, j, k)). \quad (4.49)$$

In the experiments, the normalized correlation is selected as the measurement of watermark detection:

$$\text{NC} = \frac{\sum_{i=1}^L (2\mathbf{w}(i) - 1) \cdot (2\mathbf{w}'(i) - 1)}{\sqrt{\sum_{j=1}^L (2\mathbf{w}(j) - 1)^2 \cdot \sum_{j=1}^L (2\mathbf{w}'(j) - 1)^2}}. \quad (4.50)$$

where  $L$  is the length of watermark.

The formulas above are just the rules for extracting watermark from a single frame. And a so-called voting scheme is employed to improve performance in the mass.  $m$  watermarks are extracted from  $m$  frames, and then each bit of the watermark is summed up. If the sum of certain bit is bigger than  $\lfloor \frac{m}{2} \rfloor$ , this bit is adjudged to 1; otherwise 0. And this procedure is expressed by

$$\mathbf{w}'(i) = \begin{cases} 1, & \text{if } \sum_{j=1}^m \mathbf{w}'_j(i) \geq \lfloor \frac{m}{2} \rfloor, \\ 0, & \text{if } \sum_{j=1}^m \mathbf{w}'_j(i) < \lfloor \frac{m}{2} \rfloor, \end{cases} \quad (4.51)$$

where  $i = 1, 2, \dots, L$ .

The voting scheme used in watermark extraction greatly improves the veracity of detector. And its good performance is proved in the experiments.

## Experiments

An uncompressed Bears video from Discovery Series is used as the experimental video, whose frame size is  $272 \times 352$ . Besides, this raw video consists of 50 frames, and its frame rate is 25 fps. The watermark  $w$  is a pseudorandom sequence composed of 0 and 1, whose length is  $n^2$  (in experiments,  $n = 10$ ). When embedding, it is moderate to watermark with  $\beta = 18$ , which makes watermark robust enough on the premise that the video still keeps high perceptual quality. And in the detecting procedure, the threshold is set to 0.7. Fig. 4.33 shows the PSNR of each frame in the watermarked video. According to this figure, the curve fluctuates around 46 dB, and the watermarked video has a high PSNR, which reflects high fidelity.

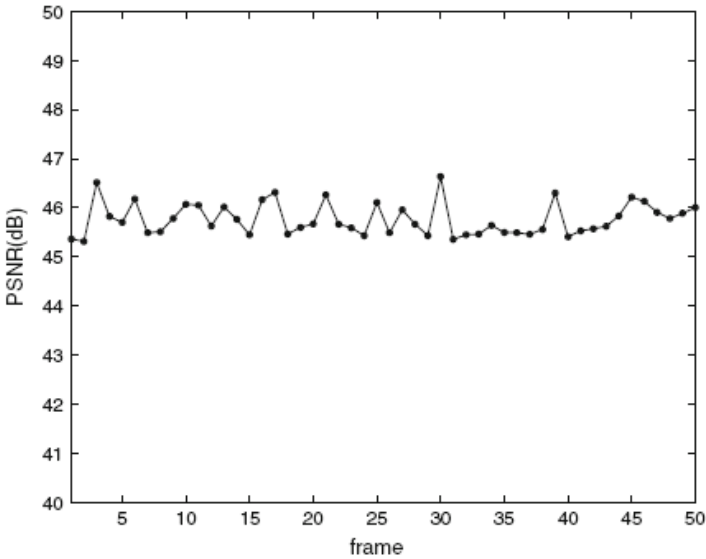


Fig. 4.33. PSNR of the watermarked video.

In order to demonstrate the robustness of the proposed scheme, the watermarked video is processed with some general information processing means. And the corresponding NC via various attacks is listed in Table 4.3. As shown in the table, Gaussian white noise of mean 0 and variance  $\text{Var}$  is added to the video, and the NC is still higher than the threshold when the  $\text{Var}$  is set to 0.002. In the frame dropping experiment, 5 frames (10% of the total frames) are dropped randomly. The dropped frames in cases 1-4 are frames 3, 10, 29, 32, 42, frames 6, 12, 15, 38, 48, frames 3, 4, 8, 14, 34, and frames 5, 28, 31, 32, 40. According to the table, the results are still 1. Additionally, frame cropping does not change the NC in first group (cropping the video from the top down, from the bottom up, from left to right, and from right to left). However, it does affect the NC in the second group (cropping random rows/columns in each frame of the video), because certain watermarked region is

**Table 4.3.** Detection of results after various attacks.

| Gaussian noise           | Mean=0        |              |                  |                 |
|--------------------------|---------------|--------------|------------------|-----------------|
| Var                      | 0.0005        | 0.0010       | 0.0015           | 0.0020          |
| NC                       | 1             | 1            | 1                | 0.8824          |
| Frame dropping           | Case 1        | Case 2       | Case 3           | Case 4          |
| NC                       | 1             | 1            | 1                | 1               |
| Frame cropping           | Crop orderly  |              |                  |                 |
| Selected rows or columns | First 10 rows | Last 10 rows | First 10 columns | Last 10 columns |
| NC                       | 1             | 1            | 1                | 1               |
| Selected rows/columns    | Case 1        | Case 2       | Case 3           | Case 4          |
| NC                       | 0.9167        | 1            | 0.9592           | 1               |

cropped too. Here, the cropped rows/columns in cases 1-4 are rows 6, 32, 58, 83, 110, 132, 163, 185, 248, 269, rows 27, 29, 30, 38, 44, 55, 67, 74, 109, 118, columns 151, 169, 215, 267, 299, 301, 302, 310, 316, 339, and columns 7, 16, 66, 73, 160, 174, 209, 210, 215, 226. Consequently, if the rows/columns cropped in this experiment include the region where the watermark is embedded, the watermark detecting will be influenced or even disturbed.

Table 4.4 shows comparisons on the detection results under MPEG-2 compression between the proposed scheme and the one presented in [59]. In the case of keeping original duration, the video is compressed under different frame rates, and the compression rate is 34:1 accordingly.

In the case of keeping original number of frames, the compression ratio is 33:1, while the frame rate is set to 23.976 and 24 fps, and 41:1, while the frame rate is set to 29.97 and 30 fps. From the table, it can be seen that the proposed scheme has



**Table 4.4.** Detection of results under MPEG-2 compression.

| Frame rate              | 23.976 fps | 24 fps | 29.97 fps | 30 fps |
|-------------------------|------------|--------|-----------|--------|
| Duration fixed          |            |        |           |        |
| NC with proposed scheme | 1          | 1      | 1         | 1      |
| NC with [22]            | –          | –      | –         | –      |
| Frame number fixed      |            |        |           |        |
| NC with proposed scheme | 1          | 1      | 1         | 1      |
| NC with [22]            | 0.8826     | 0.9298 | 0.8670    | 0.8598 |

– Denotes that the watermark cannot be detected.

fairly good robustness against MPEG-2 compression, and is superior to the scheme presented in [59] especially in the case of frame rate changing with keeping the same duration.

#### 4.4.4 Watermarking Based on Independent Content Feature [60]

In the view of digital watermark, the content feature is the most consistent feature because both the attackers and owners of the host media don't want to cause any distortion in the media content, which is easy to be perceived. The PICF described in Sec. 4.2.2 is such a kind of feature. The framework of the watermarking scheme based on PICF is described in Fig. 4.34. Firstly, the original video is segmented into shots. Here, the global histogram comparison approach is selected to segment video into shots [61].

After the PICFs having been obtained, they can be taken as images and watermarked according to the existing image watermark schemes, thus the video watermarking can be completed.

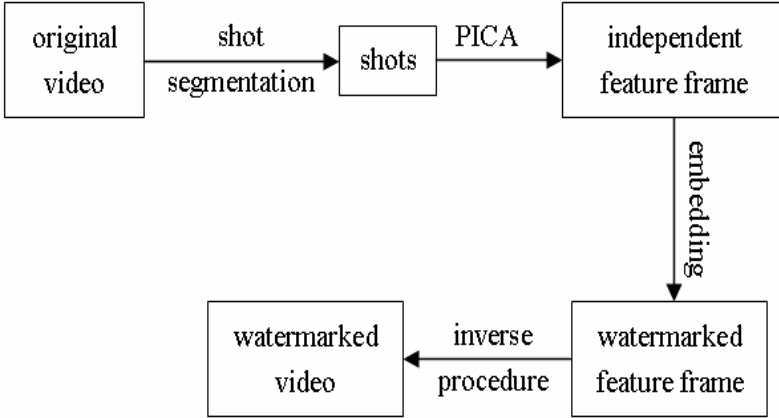


Fig. 4.34. Framework of watermarking scheme.

### Watermark Embedding

Since the PICF is robust to various video processing, any existing blind image watermarking method can be used. Here the Single Watermark Embedding (SWE) scheme is used as an example.

The SWE scheme [50] is simple and robust to compression. In this scheme, two keys are used, which are  $\mathbf{D}=[d_1, d_2, \dots, d_N | d_i \in \mathfrak{R}^+ \leq i \leq N]$  and  $\mathbf{K}=[k_1, k_2, \dots, k_M]$  to embed binary watermark  $\mathbf{w}$  into the host vector  $\mathbf{H}$  to form the watermarked vector  $\mathbf{H}'$ .  $\mathbf{H}$ ,  $\mathbf{H}'$  and  $\mathbf{K}$  are split into  $N$  subvectors of equal length  $L = \lfloor \frac{M}{N} \rfloor$  with the  $i^{\text{th}}$  subvector denoted as  $\mathbf{H}_i$ ,  $\mathbf{H}'_i$  and  $\mathbf{K}_i$ , respectively. Its embedding algorithm is:

$$\mathbf{H}'_i = \mathbf{H}_i + \alpha_i \mathbf{K}_i. \quad (4.52)$$

Here,

$$\alpha_i = \begin{cases} \frac{d_i \cdot \text{round}\left(\frac{z}{d_i}\right) - z}{\|\mathbf{K}_i\|_2^2}, & \text{if } \text{round}\left(\frac{z}{d_i}\right) \% 2 = w(i), \\ \frac{d_i \cdot \left(\text{round}\left(\frac{z}{d_i}\right) + 1\right) - z}{\|\mathbf{K}_i\|_2^2}, & \text{if } \text{round}\left(\frac{z}{d_i}\right) \% 2 \neq w(i) \text{ and } z \geq d_i \cdot \text{round}\left(\frac{z}{d_i}\right), \\ \frac{d_i \cdot \left(\text{round}\left(\frac{z}{d_i}\right) - 1\right) - z}{\|\mathbf{K}_i\|_2^2}, & \text{if } \text{round}\left(\frac{z}{d_i}\right) \% 2 \neq w(i) \text{ and } z < d_i \cdot \text{round}\left(\frac{z}{d_i}\right), \end{cases}$$

where  $z = \langle \mathbf{H}_i, \mathbf{K}_i \rangle$ ,  $\text{round}(\cdot)$ ,  $\%$  and  $\|\cdot\|_2$  are rounding, modulo 2 and  $L^2$ -norm respectively. After being watermarked, the original PICFs are replaced by the watermarked ones, and the inverse ICA is performed to get watermarked shots. Finally, the watermarked video is gotten after organizing these watermarked shots as their original timing order.

## Watermark Detection

The detection is performed on each PICF of the detection video. The detecting algorithm is:

$$w'(i) = \text{round}(\langle \mathbf{H}'_i, \mathbf{K}_i \rangle / d_i) \% 2, \quad (4.53)$$

where  $w'(i)$  is the extracted watermark.

The score is set to evaluate the detection result.

$$\frac{\sum_{i=1}^N (2w(i) - 1) \cdot (2w'(i) - 1)}{\sqrt{\sum_{i=1}^N (2w(i) - 1)^2 \cdot \sum_{i=1}^N (2w'(i) - 1)^2}} = \frac{1}{N} \sum_{i=1}^N (2w(i) - 1) \cdot (2w'(i) - 1). \quad (4.54)$$

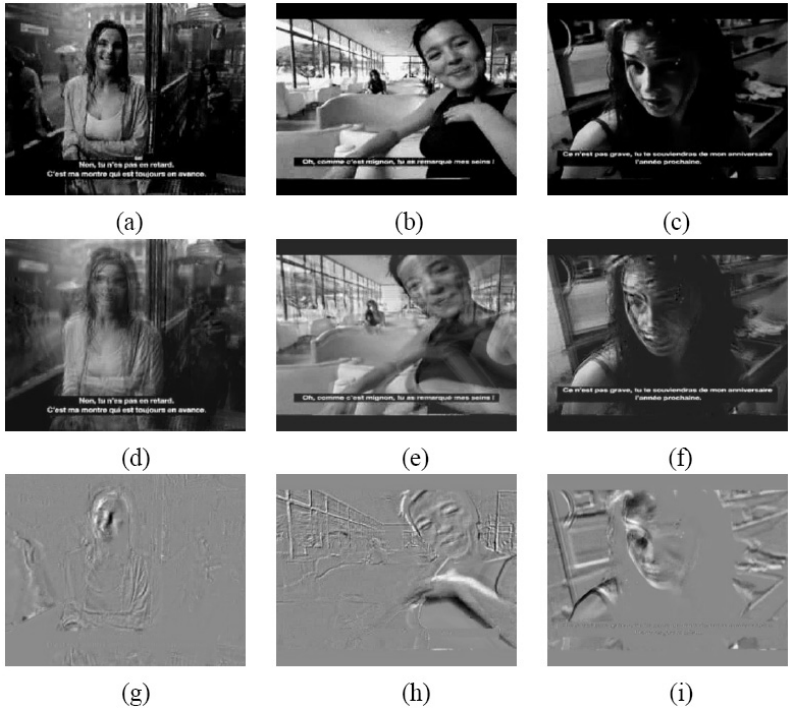
The embedding and detecting are performed on every obtained PICF. And the maximum detection score of each segment is selected as the detection score of this segment. If the score is over a threshold, the segment is considered to be watermarked. As long as there is one watermarked segment, the whole video is watermarked.

## Experiments

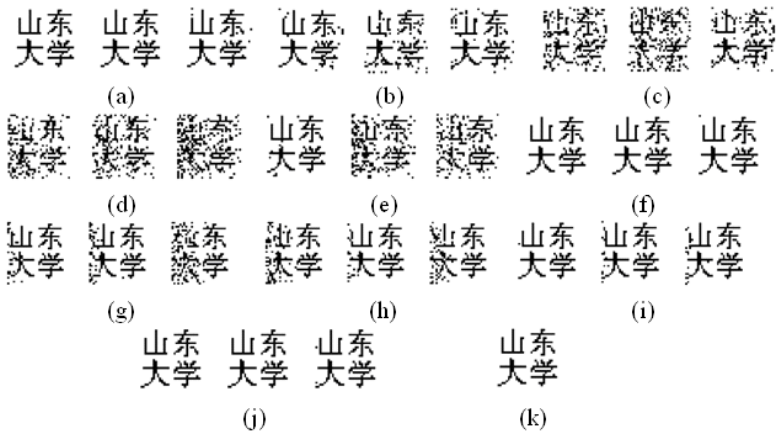
Fig. 4.35 shows the arbitrary frame and PICFs in each segment. Fig. 4.35(a)–(c) are the frames of original video, Fig. 4.35(d)–(f) are the background PICFs, and Fig. 4.35(g)–(i) are the arbitrary motion PICFs.

In the experiments, the PICFs of each segment are watermarked same and 10 attacks are performed after MPEG-2 encoding and decoding, including (a)  $3 \times 3$  median filter, (b) frame shrink with factor 2, (c) frame rotation with 0.5 degree, (d) salt noise with density 0.02, (e) Gaussian noise with mean 0 and variance 0.001, (f) intra-segment frame swapping, (g) frame dropping, (h) fps changing from 25 to 30 fps, (i) fps changing from 25 to 24 fps, and (j) intra-video collusion in each video segment. The detected watermarks of the above attacks are shown in Fig. 4.36(a)–(j) respectively and the original watermark is shown in Fig. 4.36(k). In Fig. 4.36, there are three detected watermarks for each attack for there are three segments in the video. From the left to the right, the detected watermarks are extracted from the first, second and third segment respectively. In (f) attack, the exchanged frames are in the same segment in order to keep the quality of the video. And in each segment, there are two frames exchanged every five frames. There is one dropped frame of every five frames in each segment in the frame dropping attack. There are two frame rate changing attacks (h), (i). In the former, the frame rate is changed from 25 to 30 fps and the number of frames increases to 85. And in the latter, the frame rate is changed from 25 to 24 fps and the number of frames decreases to 69.

From Fig. 4.36 and Table 4.5, it is obvious that the proposed scheme is robust to most of the common frame-based attacks, such as median filtering, resizing, salt noising, Gaussian noising, and also video-based attacks including MPEG-2 compression, frame swapping, frame dropping, fps changing, intra-video collusion. And it is robust to spatial desynchronization, e.g. rotation, to some extent.



**Fig. 4.35.** The frames and PICFs in each segment. (a) original frame of 1st segment; (b) original frame of 2nd segment; (c) original frame of 3rd segment; (d) background PICF of 1st segment; (e) background PICF of 2nd segment; (f) background PICF of 3rd segment; (g) motion PICF of 1st segment; (h) motion PICF of 2nd segment; (i) motion PICF of 3rd segment.



**Fig. 4.36.** The watermark extracted from each segment after attacks.

**Table 4.5.** NC detected from each segment with embedding in PICFs after various attacks.

|                | Attack (a) | Attack (b) | Attack (c) | Attack (d) | Attack (e) |
|----------------|------------|------------|------------|------------|------------|
| First segment  | 1          | 0.9609     | 0.7051     | 0.7227     | 0.9824     |
| Second segment | 1          | 0.8516     | 0.6094     | 0.7207     | 0.7070     |
| Third segment  | 0.9863     | 0.9199     | 0.8281     | 0.5762     | 0.7598     |
|                | Attack (f) | Attack (g) | Attack (h) | Attack (i) | Attack (j) |
| First segment  | 1          | 0.9375     | 0.8398     | 0.9981     | 1          |
| Second segment | 1          | 0.9063     | 0.9668     | 0.9727     | 1          |
| Third segment  | 0.9981     | 0.7520     | 0.8848     | 0.9688     | 0.9961     |

At present, the temporal wavelet is usually used to extract video feature, wavelet feature frames (WFFs), to represent the motion and the background feature of video. Here, a comparison experiment is performed. In this experiment, the video is segmented into shots, and the WFFs of each shot are extracted just like what is done in [56], and the watermark is embedded into the WFFs by SWE scheme, and then the watermarked video is obtained. All the above attacks are performed on this watermarked video, and the detection results are listed in Table 4.6.

From the comparison with the results in Table 4.5 and Table 4.6, PICFs have better stability than WFFs under various attacks on video watermark. It is this property of PICFs that makes the watermark more robust. Compared with WFFs, PICFs are more suitable for watermarking.

**Table 4.6.** NC detected from each segment with embedding in WFFs after various attacks.

|                | Attack (a) | Attack (b) | Attack (c) | Attack (d) | Attack (e) |
|----------------|------------|------------|------------|------------|------------|
| First segment  | 0.9922     | 0.5391     | -0.6250    | -0.6250    | -0.6250    |
| Second segment | 0.9863     | 0.5469     | -0.6152    | -0.6152    | -0.6152    |
| Third segment  | 0.9728     | 0.6230     | -0.6211    | -0.6211    | -0.6211    |
|                | Attack (f) | Attack (g) | Attack (h) | Attack (i) | Attack (j) |
| First segment  | 0.5215     | 0.8538     | 0.8082     | 0.8203     | 0.8871     |
| Second segment | 0.5527     | 0.8770     | 0.7402     | 0.7246     | 0.8920     |
| Third segment  | 0.0977     | 0.8355     | 0.7832     | 0.9707     | 0.9051     |

## References

1. Jutten, C., Herault, J.: Blind separation of sources, Part I: An adaptive algorithm based on neuromimetic architecture. *Signal Processing* 24, 1–10 (1991)
2. Tong, L., Inouye, Y., Liu, R.: Waveform preserving blind estimation of multiple independent sources. *IEEE Trans. on Signal Processing* 41, 2461–2470 (1993)
3. Comon, P.: Independent component analysis — A new Concept? *Signal Processing* 36, 287–314 (1994)
4. Hyvrinen, A., Oja, E.: A fast fixed-point algorithm for independent component analysis. *Neural Computation* 9, 1483–1492 (1997)
5. Cardoso, J.F.: Blind signal separation: Statistical principles. *Proceedings of the IEEE* 9, 2009–2025 (1998)
6. Amari, S.I., Cichocki, A.: Adaptive blind signal processing-neural network approaches. *Proceedings of the IEEE* 86, 2026–2048 (1998)
7. Hyvarinen, A.: Survey on independent component analysis. *Neural Computing Surveys* 2, 94–128 (1999)
8. Girolami, M.: *Self-Organising Neural Networks — Independent Component Analysis and Blind Source Separation*. Springer (1999)
9. Jutten, C.: Source separation: from dusk till dawn. In: *Proc. 2nd Int. Workshop on Independent Component Analysis and Blind Source Separation*, pp. 15–26 (2000)
10. Lee, T.W.: *Independent Component Analysis — Theory and Applications* (1998)

11. Lee, T.W., Girolami, M., Bell, A.J., et al.: A unifying information-theoretic framework for independent component analysis. *Computers and Mathematics with Applications* 31, 1–12 (2000)
12. He, Z., Liu, J., Yang, L., et al.: An ICA and EC based approach for blind equalization and channel parameter estimation. *Science in China, Series E* 43, 1–8 (2000)
13. He, Z., Yang, L., Liu, J., et al.: Blind source separation using cluster-based multivariate density estimation algorithm. *IEEE Trans. Signal Processing* 48, 575–579 (2000)
14. Hurri, J., Hyvarinen, A., Karhunen, J., et al.: Image feature extraction using independent component analysis. In: *Proc. IEEE Nordic Signal Processing Symposium* (1996)
15. Hyvarinen, A., Karhunen, J., Oja, E.: *Independent Component Analysis*. John Wiley, New York (2001)
16. Novey, M., Adall, T.: ICA by maximization of non-gaussianity using complex functions. In: *Proc. IEEE Workshop on Machine Learning for Signal Processing*, pp. 21–26 (2005)
17. Pearlmutter, B.A., Parra, L.C.: Maximum likelihood blind source separation: A context-sensitive generalization of ICA. In: *Advances in Neural Information Processing Systems* 9, pp. 613–619 (1997)
18. Eriksson, J., Karvanen, J., Koivunen, V.: Source distribution adaptive maximum likelihood estimation of ICA model. In: *Proc. 2nd Int'l. Workshop on Independent Component Analysis and Blind Signal Separation*, pp. 227–232 (2000)
19. Erdogmus, D., Hild, K.E., Rao, Y.N., et al.: Minimax mutual information approach for independent component analysis. *Neural Computation* 16, 1235–1252 (2004)
20. Ungureanu, M., Bigan, C., Strungaru, R., et al.: Independent component analysis applied in biomedical signal processing. *Measurement Science Review* 4, 1–8 (2004)
21. Lee, J.H., Jung, H.Y., Lee, T.W., et al.: Speech feature extraction using independent component analysis. In: *Proc. IEEE Int'l. Conf. Acoustics, Speech, and Signal Processing*, vol. 3, pp. 1631–1634 (2000)
22. Kwak, N., Choi, C.-H., Choi, J.-Y.: Feature Extraction Using ICA. In: *Dorffner, G., Bischof, H., Hornik, K. (eds.) ICANN 2001. LNCS*, vol. 2130, pp. 568–573. Springer, Heidelberg (2001)
23. Ozawa, S., Kotani, M.: A Study of Feature Extraction and Selection Using Independent Component Analysis. In: *Proc. 7th Int'l. Conf. Neural Information Processing*, vol. 1, pp. 369–374 (2000)
24. Chagnaa, A., Ock, C.Y., Lee, C.B., et al.: Feature extraction of concepts by independent component analysis. *International Journal of Information Processing Systems* 3, 33–37 (2007)
25. Lu, C.J., Lee, T.S., Chiu, C.C.: Financial time series forecasting using independent component analysis and support vector regression. *Decision Support Systems* 47, 115–125 (2009)
26. Cha, S.M., Chan, L.W.: Applying independent component analysis to factor model in finance. In: *Proc. Intelligent Data Engineering and Automated Learning*, pp. 538–544 (2000)
27. Oja, E., Kiviluoto, K., Malaroiu, S.: Independent component analysis for financial time series. In: *Proc. The Symp. of Adaptive Systems for Signal Processing, Communications, and Control*, pp. 111–116 (2000)
28. Wang, F., Li, H., Li, R.: Data mining with independent component analysis. In: *Proc. 6th World Congress on Intelligent Control and Automation*, pp. 6043–6047 (2006)
29. Bingham, E.: *Advances in independent component analysis with applications to data mining*. Doctoral Thesis (2003)
30. Liu, J., Sun, J.: ICA-based image/video analysis and applications in watermarking. *CAAI Trans. Intelligent System* 6(6), 495–506 (2011)

31. Liu, J., Hu, H., Sun, J., et al.: An ICA-based watermarking scheme resistant to copy attack. In: Proc. IEEE Int'l. Workshop VLSI Design and Video Technology, pp. 154–157 (2005)
32. Sun, J., Liu, J.: A novel digital watermark scheme based on image independent feature. In: Proc. 2003 IEEE Int'l. Conf. Robotics, Intelligent Systems and Signal Processing, pp. 1333–1338 (2003)
33. Sun, J., Liu, J., Zhang, X.: A novel watermark scheme based on image independent feature. *Journal of Circuit and System* 8, 53–56 (2003)
34. Zhang, Q., Sun, J., Liu, J., et al.: A novel ICA-Based Image Video Processing Method. In: Proc. Int'l. Symp. Independent Component Analysis and Blind Signal Separation, pp. 836–842 (2007)
35. Sun, J., Liu, J.: Data hiding with video independent components. *IEE Electronics Letters* 40, 858–859 (2004)
36. Hateren, J.H.V., Ruderman, D.L.: Independent component analysis of natural image sequences yields spatio-temporal filters similar to simple cells in primary visual cortex. In: Proc. the Royal Society of London B, pp. 2315–2320 (1998)
37. Gonzalez-Serrano, F.J., Molina-Bulla, H.Y., Murillo-Fuentes, J.J.: Independent component analysis applied to digital image watermarking. In: Proc. IEEE Int'l. Conf. Acoustics, Speech and Signal Processing, pp. 1997–2000 (2001)
38. Liu, J., Sun, J., Du, Z., et al.: Embodying information into images by an MMI-based independent component analysis algorithm. In: Proc. ICSP, pp. 1600–1603 (2002)
39. Yu, D., Sattar, F., Ma, K.K.: Watermark detection and extraction using independent component analysis method. *EURASIP Journal on Applied Signal Processing* 1, 92–104 (2002)
40. Bounkong, S., Toch, B., Saad, D., et al.: ICA for watermarking digital images. *Journal of Machine Learning Research* 4, 1471–1498 (2003)
41. Shen, M., Zhang, X., Sun, L., et al.: A method for digital image watermarking using ICA. In: Proc. 4th Int'l Symp. Independent Component Analysis and Blind Signal Separation, pp. 209–214 (2003)
42. Liu, J., Zhang, X., Sun, J., et al.: A digital watermarking scheme based on ICA detection. In: Proc. 4th Int'l. Symp. Independent Component Analysis and Blind Signal Separation, pp. 215–220 (2003)
43. Liu, J., Sun, J.: A new scheme of digital watermarking based on independent component analysis. *Acta Electronica Sinica* 32, 657–660 (2004)
44. Hu, H., Liu, J., Sun, J.: A novel ICA-based blind watermarking algorithm with resistance to copy attack. In: Proc. the 7th International Conference on Signal Processing, vol. 3, pp. 2346–2349 (2004)
45. Ling, J., Liu, J., Sun, J., et al.: Novel image watermarking scheme based on ICA. In: Proc. IEEE Int'l Conf. Neural Networks & Signal Processing, pp. 73–77 (2008)
46. Niu, X.-L., Liu, J., Sun, J.-D., Qiao, J.-P.: A Novel Watermarking Method with Image Signature. In: Wang, J., Yi, Z., Žurada, J.M., Lu, B.-L., Yin, H. (eds.) *ISNN 2006, Part III. LNCS*, vol. 3973, pp. 293–298. Springer, Heidelberg (2006)
47. Moulin, P., O'Sullivan, J.A.: Information-theoretic analysis of information hiding. *IEEE Trans. Information Theory* 49, 73–77 (2008)
48. Lu, C.S., Liao, M.H.Y., Kutter, M.: A new watermarking scheme resistant to denoising and copy attacks. In: Proc. 4th IEEE Workshop Multimedia Signal Processing, pp. 505–510 (2001)
49. Chen, B., Wornell, G.W.: Quantization index modulation: A class of provably good methods for digital watermarking and information embedding. *IEEE Trans. Information Theory* 47, 1423–1443 (2001)



50. Wong, P.H.W., Au, O.C., Yeung, Y.M.: A novel blind multiple watermarking technique for images. *IEEE Trans. Circuits and Systems for Video Technology* 13, 813–830 (2003)
51. Hong, I., Kim, I., Han, S.S.: A blind watermarking technique using wavelet transform. In: *Proc. IEEE Int'l. Symp. Industrial Electronics*, vol. 3, pp. 1946–1950 (2001)
52. Barr, J., Bradley, B., Hannigan, B.T.: Using digital watermarks with image signatures to mitigate the threat of the copy attack. In: *Proc. of 2003 IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 3, pp. 69–72 (2003)
53. Su, K., Kunder, D., Hatzinakos, D.: A novel approach to collusion-resistant video watermarking. In: *Proceedings of SPIE 4675, Security and Watermarking of Multimedia Content IV*, pp. 491–502 (2002)
54. Sun, Z., Liu, J., Sun, J., et al.: A motion location based video watermarking scheme using ICA to extract dynamic frames. *Neural Computing & Applications* 18, 507–514 (2009)
55. Li, Q., Cox, I.J.: Using perceptual models to improve fidelity and provide invariance to valumetric scaling for quantization index modulation watermarking. In: *Proc. IEEE Int'l. Conf. Acoustics, Speech and Signal Processing*, vol. 2, pp. 1–4 (2005)
56. Swanson, M.D., Zhu, B., Tewfik, A.H.: Multiresolution scene-based video watermarking using perceptual models. *IEEE Journal on Selected Areas in Communications* 16, 540–550 (1998)
57. Watson, A.B.: DCT quantization matrices visually optimized for individual images. In: *Proc. SPIE Human Vision, Visual Processing, and Digital Display*, vol. 9, pp. 202–216 (1993)
58. Cox, I.J., Miller, M.L., Bloom, J.A.: *Digital Watermarking*. Morgan Kaufmann (2001)
59. Sun, J., Liu, J.: A novel blind video watermarking scheme based on independent dynamic component. *Multidimensional Systems and Signal Processing* 17, 59–74 (2006)
60. Sun, J., Liu, J.: A blind video watermarking scheme based on ICA and shot segmentation. *Science in China Series F* 49, 302–312 (2006)
61. Koprinska, I., Carrato, S.: Temporal video segmentation: A survey. *Signal Processing: Image Communication* 16, 477–500 (2001)

# Content Based Invariant Image Watermarking with High Capacity

Leida Li<sup>1</sup>, Jianying Zhang<sup>2</sup>, and Baolong Guo<sup>3</sup>

<sup>1</sup> School of Information and Electrical Engineering,  
China University of Mining and Technology,  
Xuzhou, China  
reader1104@hotmail.com

<sup>2</sup> School of Information and Electrical Engineering,  
China University of Mining and Technology,  
Xuzhou, China  
zjycumt@126.com

<sup>3</sup> Institute of Intelligent Control and Image Engineering (ICIE),  
School of Mechano-Electronic Engineering,  
Xidian University,  
Xi'an, China  
blguo@xidian.edu.cn

**Summary.** Recently, many content based image watermarking schemes have been addressed to resist geometric attacks, such as rotation, scaling and translation (RST). The underlying idea of this kind of method is to align the geometric transforms before watermark embedding and extraction. These schemes are usually preceded by watermark synchronization, an operation to determine some local regions for watermark embedding and extraction. Due to the small size of the regions, the capacities of these schemes are usually limited, typically tens of bits. In this chapter, we propose two methods that can embed high capacity watermark images, while maintaining robustness to the general geometric attacks. Compared to the existing schemes, the proposed high capacity watermarking methods embed the watermark in a content based manner. The efficiencies of the proposed schemes are demonstrated by experiments.

## 5.1 Introduction

After more than ten years' development, great achievements have been achieved in the digital watermarking community, such as intelligent watermarking [1], reversible watermarking [2] and multipurpose watermarking [3]. However, the real-world applications are still limited. The reasons are complex, among which geometric attack is one of the bottlenecks. In practical applications, geometric attacks, such as image rotation, scaling and translation (RST), are common and easy to implement. However, these operations can defeat most of the existing watermarking

schemes, because the synchronization between watermark embedding and detection is destroyed. In other words, the detector cannot detect the watermark signal from the places where it is originally embedded. In order to resist this kind of attacks, many schemes have been proposed these years, and they can be classified into the following categories [4, 5].

1. Inverse transform based method. This kind of method first rectifies the geometric error before watermark extraction. Extensive search and template based methods belong to this category [6, 7].
2. Invariant domain based embedding. From this point of view, image normalization [8, 9], Fourier-Mellin [10, 11, 12] and invariant image moments [13]–[17] are commonly employed for watermark embedding. The idea of image normalization and Fourier-Mellin based embedding is to obtain a standard image before watermark embedding and extraction. However, the embedding process will inevitably introduce interpolation error, which is mainly due to the inverse normalization and inverse log-polar mapping. In the invariant moment based methods, the watermark is usually encoded into the magnitudes of the moments.
3. Feature based embedding. Feature extraction methods have been extensively used in pattern recognition problems. The local features extracted from an image are often invariant to the general affine transforms, thus are consistent with the requirement of watermarking.

Feature based watermarking can be regarded as the second generation watermarking [18], because the embedded watermark signal is usually associated with the image contents (interest points, edges etc.). This kind of method has the following advantages:

1. Good invisibility. The watermark is embedded into the local regions instead of the entire image, so the image quality is better.
2. Enhanced robustness.

These schemes can resist to both geometric attacks and traditional signal processing attacks. The local image features are invariant to a variety of affine distortions, including RST attacks, viewpoint change, aspect change etc. Embedding the watermark by referring to these features guarantee the invariance of the watermark. Recently, many content based watermarking schemes have been reported using feature points [19]–[25]. Among these schemes, two kinds of watermark signals are employed. The first kind is a binary watermark sequence, typically tens of bits. In [21], a 16-bit long binary watermark is embedded into the Harris corner based local regions in spatial domain. In [24], the 16-bit long watermark is embedded in the Fourier domain. An improved method is addressed in [25], where the watermark capacity is 32 bit. The second kind of watermark is a random binary pattern. Fig. 5.1 shows the watermark patterns used in some of the existing schemes.

It is seen from Fig. 5.1 that the watermark image is in a random pattern, without physical meaning. This kind of watermark is usually embedded adaptively to the local region. Namely, the watermark image is rendered according to the shape and size of the local image region. Besides, they are usually embedded additively in

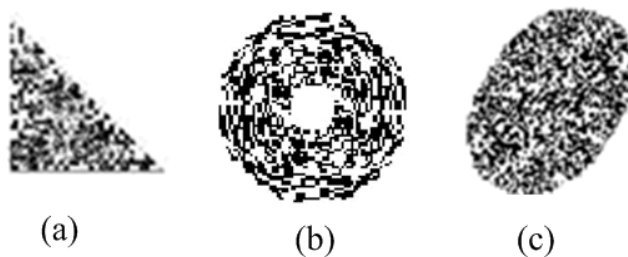


Fig. 5.1. The watermark images used in [19], [20] and [23].

spatial domain. In the decoder, correlation based watermark extraction is commonly adopted. Due to the additive embedding scheme in spatial domain, the robustness of this kind of method is not satisfactory.

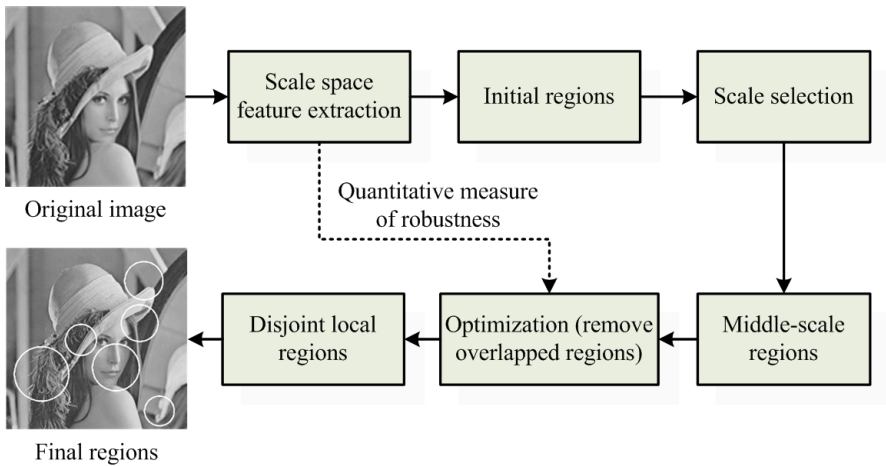
It is challenging to embed a high capacity watermark signal into the feature based local regions, because the size of the regions are small. In this chapter, we mainly investigate the principles on embedding binary watermark images into the feature based local regions.

## 5.2 Content Based Watermark Synchronization

Feature based watermarking schemes can resist geometric attacks, because the geometric error can be aligned by watermark synchronization. In this section, we first review the current feature based synchronization methods quickly, and then we propose an improved framework for multi-scale feature based watermark synchronization.

Most of the existing schemes employ feature point, also called interest point, as the underlying invariant feature. The popular feature point extraction methods include, but not limited to, Mexican-Hat Wavelet based feature point [24], Harris corner [26], Harris-Laplace [27] and scale-invariant feature transform (SIFT) [28]. In [24], the interest points are extracted by Mexican hat wavelet scale interaction, and some local regions are generated by setting a fixed radius. In order to obtain disjoint regions, a feature point has a higher priority for watermark embedding if it has more neighboring feature points inside its disk. Wang et al. improves this method using multi-scale Harris-Laplace feature points [25]. In [19], the Harris feature points are first extracted, and a Delaunay tessellation is then performed on the feature points, producing many triangular regions. Then the random binary watermark image is embedded into each of the regions. In [21], the concept of locally most stable feature point (LMSP) is proposed to achieve watermark synchronization. Compared to the existing schemes, the LMSP based synchronization performs better. In [20], the SIFT features are employed to determine the places for both watermark embedding and extraction. The binary watermark pattern is then embedded in spatial domain additively.

The current feature point based watermark synchronization schemes can be grouped into two categories. One kind employs traditional interest point detection methods, including Mexican wavelet based features and Harris corners. The other kind utilizes the scale space features for watermark synchronization, including Harris-Laplace and SIFT. Traditional feature extraction methods detect the feature point at a specific image resolution, so the features are not invariant to image scaling. By comparison, scale space feature points are detected inherently with a scale parameter, namely the characteristic scale. The local regions are determined by referring to the characteristic scale, so the local regions are scale invariant. From this point of view, scale space feature based synchronization is preferable for efficient watermark embedding. Another requirement in watermark synchronization is that the local regions should be disjoint, so that the embedded watermark do not affect each other. In order to achieve this goal, the feature with better robustness should be used. Based on the above findings, we propose an improved watermark synchronization framework based on scale space features. Fig. 5.2 shows the diagram of watermark synchronization.



**Fig. 5.2.** Space space feature based watermark synchronization.

The main difference between the proposed scheme and the existing schemes is that an optimization operation is added. The goal of this step is to measure the robustness of the feature points (namely the local regions), and select the region with better robustness to generate the final disjoint regions. This process is crucial for the performance of watermark synchronization. We claim that a quantitative measure of robustness (invariance) is desired to select the features for watermark synchronization. In implementation, this can be done using the responses of the feature extraction method. Generally, feature points with bigger responses tend to have better invariance. In the following of this chapter, the high capacity watermark embedding schemes are based on the improved framework of watermark synchronization.

### 5.3 High Capacity Watermark Embedding

In this section, we propose two high capacity watermark embedding schemes using scale space feature based synchronization. In particular, we investigate how to embed meaningful binary watermark images into the local regions.

#### 5.3.1 Content Based Embedding

As the size of the local region is small, traditional watermarking schemes are not suitable for high capacity watermark embedding. We claim that to achieve high capacity watermark embedding, the watermark encoding scheme should be conducted in a content based manner. The authors propose a high capacity embedding scheme in [29]. The diagram of the proposed scheme is shown in Fig. 5.3.

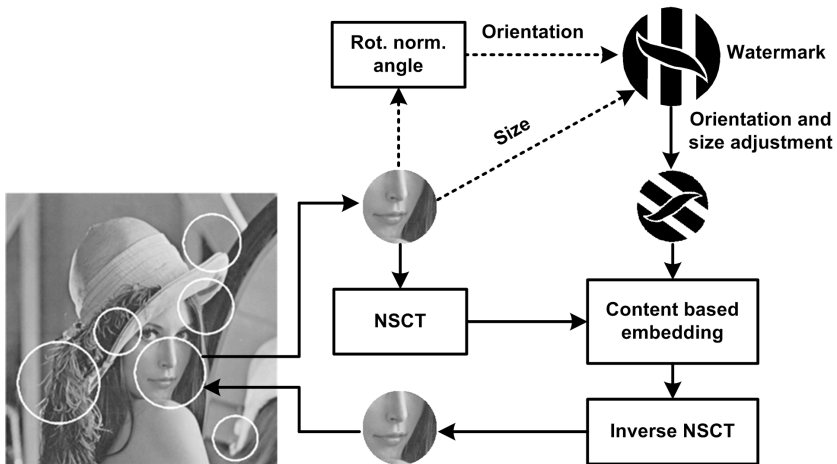


Fig. 5.3. Diagram of the proposed scheme.

For a digital image, local regions are first extracted. Then the rotation normalization angle is computed for each region. Before embedding, the watermark image is adjusted with respect to the orientation and size. After processing, the size of the watermark image is same to that of the local region. Besides, the orientation of the watermark image is adaptive to the local region. Fig. 5.4 shows some of the local region and the adjusted watermark images. The watermark images are processed to avoid processing of the local regions. The reason is that if the regions are rendered for watermarking, they have to be inversely processed to obtain the watermarked image, which will inevitably introduce interpolation error. By processing the watermark image instead of the local regions, both the image quality and watermark robustness can be improved.

Fig. 5.5 shows the watermark encoding process. For a region, it is first decomposed using the Non-subsampled Contourlet Transform (NSCT). Then the low



Fig. 5.4. Local region and the processed watermark.

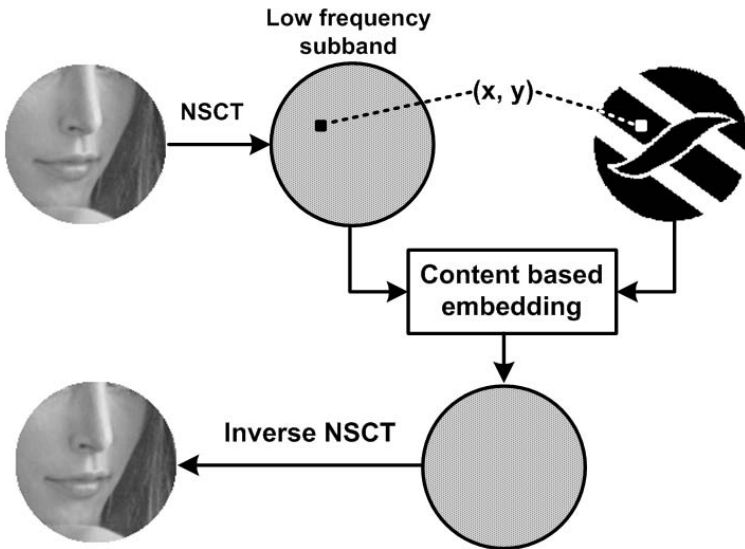


Fig. 5.5. Watermark encoding process.

frequency subband coefficients are used to embed the watermark. As the size of the watermark is same size to that of the region, a watermark bit is encoded by modifying one NSCT coefficient that has the same coordinate.

During extraction, a region is first transformed into NSCT domain and the low frequency subband coefficients are selected. Then the watermark image can be extracted. It should be noted that the orientation of the extracted watermark is not same to the original watermark image. The irregularity of the orientation can be rectified by referring to the rotation normalization angle. Fig. 5.6 shows the process of watermark rectification.

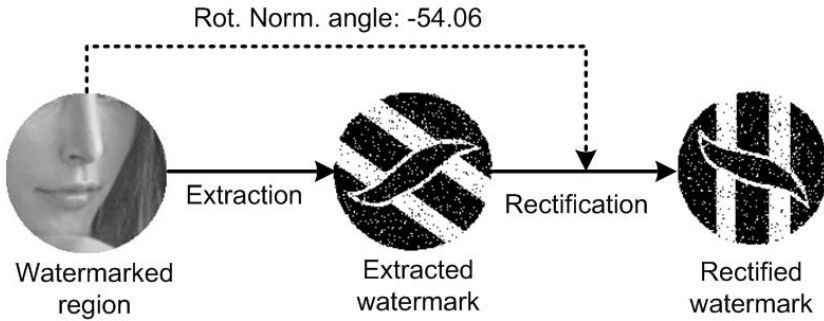


Fig. 5.6. Watermark rectification.

### 5.3.2 Partition Based Embedding

In section 5.3.1, a content based embedding scheme is addressed. One possible deficiency of the scheme is that a watermark bit is encoded by a single coefficient, so watermark robustness is not very satisfactory. In this section, we propose a partition based high capacity watermark embedding scheme. Unlike the existing schemes, the local region used is square. The center of the square is the feature point, and the size of the square region is

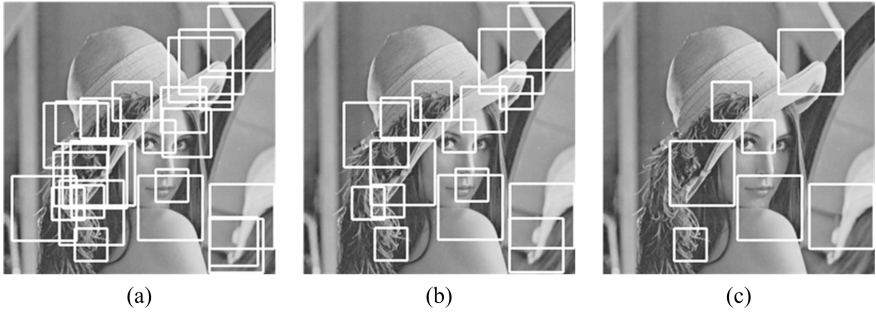
$$2\tau\sigma \times 2\tau\sigma \quad (5.1)$$

where  $\tau$  is a magnification factor, and  $\sigma$  is the characteristic scale. In implementation, the 100 most robust features are first extracted. Then the features too close to image borders are discarded. In order to enhance robustness, features with small or large characteristic scales are also removed, because they have lower repeatability. In order to embed the watermark, these regions should be disjoint. In this chapter, this is done by reducing the less robust regions while reserving the robust ones. For each feature point with characteristic scale  $\sigma$ , a neighboring circular area is determined, whose radius is  $\tau\sigma$ . Note that in a circular area, other features may also exist in this area. The response value of the central feature is compared with those of other features in this region. If the response of the central feature point achieves local maximum, it is regarded as a locally stablest feature. This operation is done repeatedly for all features. Then the locally stablest features can be obtained. To obtain the disjoint regions, a further optimization procedure is conducted as follows. The region with the best robustness (biggest response) is first selected, and the regions that overlap with it is discarded. This operation is done repeatedly until all regions are processed, producing the final disjoint regions. An example of the local square regions generated on standard image Lena is shown in Fig. 5.7.

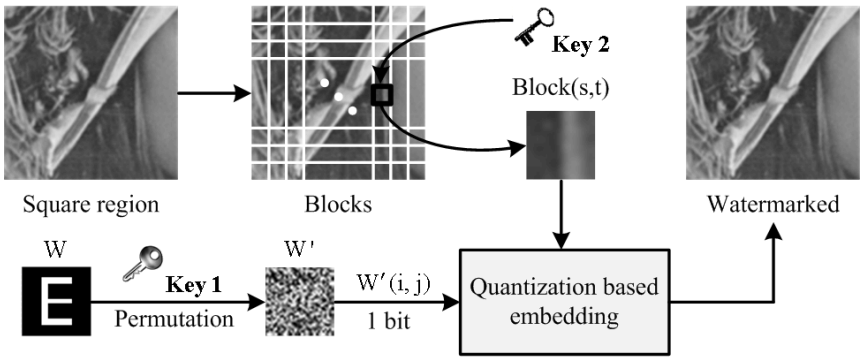
The proposed method is based on partition, namely the local region is divided into blocks and the watermark bits are embedded into each of the blocks. Fig. 5.8 illustrates how to embed the watermark into a local region.

For a square region, it is first divided into blocks with the same size. Furthermore, the number of blocks is equal to the size of the watermark image, which is denoted





**Fig. 5.7.** Generation of local regions. (a) Middle scale regions, (b) Locally stablest regions, (c) Final regions.



**Fig. 5.8.** Diagram of the partition based embedding.

by  $N_w \times N_w$  here. Then each block is employed to encode one watermark bit. As the size of the region is  $2\tau\sigma \times 2\tau\sigma$ , the size of the block is

$$N_{blk} = \frac{2\tau\sigma}{N_w}. \tag{5.2}$$

In order to enhance the security of the proposed scheme, two secret keys are adopted during the watermarking process. The first key, *Key1*, is used to permute the watermark image before embedding. The second key, *Key2*, is used to randomly select a block to encode a watermark bit. In this letter, the original watermark image is denoted by  $W$ , and the permuted one is denoted by  $W'$ . For a permuted watermark bit  $W'(i, j)$ ,  $i, j = 1, 2, \dots, N_w$ , a block  $Blk(s, t)$ ,  $s, t = 1, 2, \dots, N_w$ , is first selected randomly using *Key2*. Then  $W'(i, j)$  is embedded into  $Blk(s, t)$  by modifying all the pixels within it.

Fig. 5.9 illustrates the process of encoding a single watermark bit. If  $W'(i, j) = 0$ , all the pixels in  $Blk(s, t)$  are quantized to have even values. If  $W'(i, j) = 1$ , then the pixels are quantized to have odd values. In other words, the block is modified to have the same odd or even property as the encoded watermark bit.

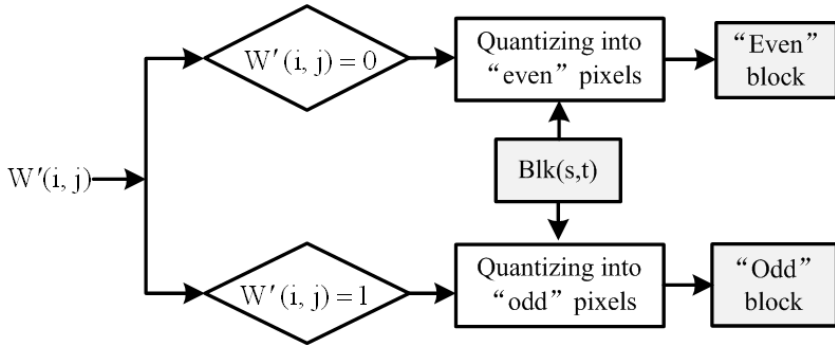


Fig. 5.9. The process of encoding one watermark bit.

Watermark extraction is the inverse process of watermark embedding. For a distorted image, the local square regions are first generated. Then the watermark are extracted in a block based manner. The square region is first divided into blocks with the same number as the size of the watermark image, i.e.  $N_W \times N_W$ . As the size of the square region has changed to be  $2\tau\sigma' \times 2\tau\sigma'$ , with  $\sigma'$  the characteristic scale, the size of the block during watermark extraction is

$$N'_{blk} = \frac{2\tau\sigma'}{N_W}. \quad (5.3)$$

It should be noted that  $\sigma'$  may be different to  $\sigma$ , because the image may have been rescaled. Therefore, the size of the block in watermark extraction may be different to that of watermark embedding. In implementation, a block is first selected using the same secret key  $Key2$ , as embedding. Then a watermark bit can be decoded from each block by judging the odd even property of the block.

## 5.4 Simulation Results

In this section, the performances of the proposed watermarking schemes are estimated by simulations. In experiments, gray scale standard images with size  $512 \times 512$  are employed as the original images, and some of them are shown in Fig. 5.10.

The performances of the proposed schemes are estimated from two aspects, namely watermark invisibility and watermark robustness. In the invisibility test, the commonly used peak signal to noise ratio (PSNR) is used to test the quality of the watermarked images.

### 5.4.1 Watermark Invisibility

Fig. 5.11 and Fig. 5.12 show the effect of watermark embedding of the two schemes. The original images, watermarked images are given together with the magnified residual images. For the content based embedding, the PSNR values of Lena and

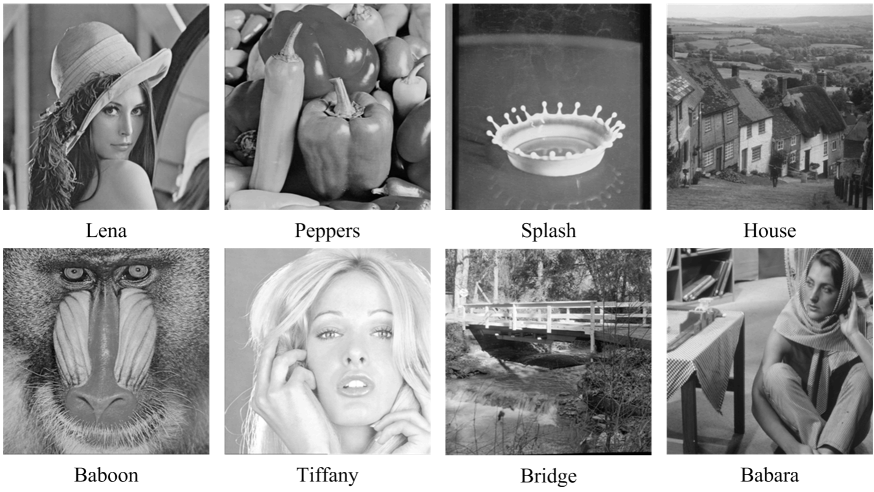


Fig. 5.10. Test image used in experiments.

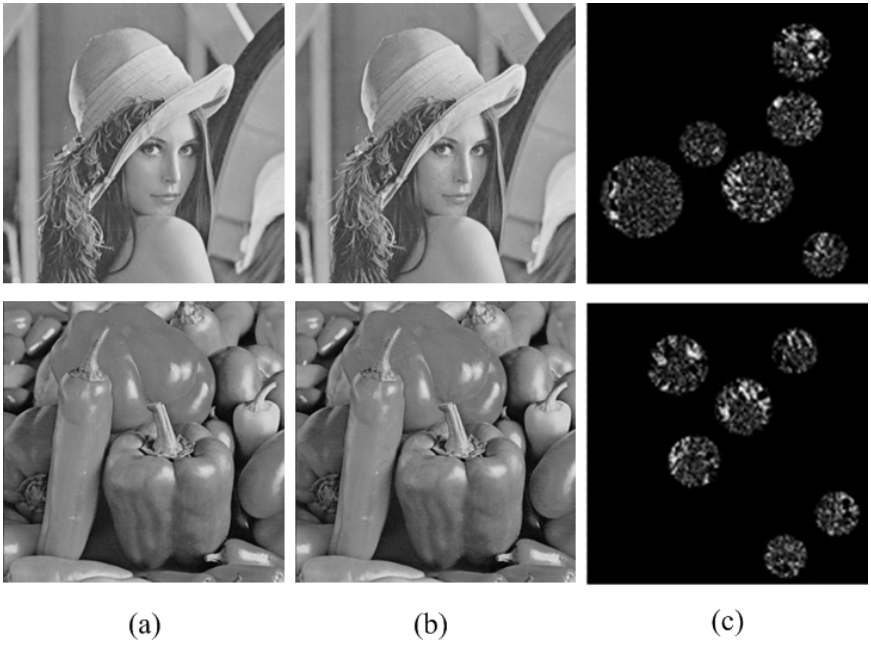
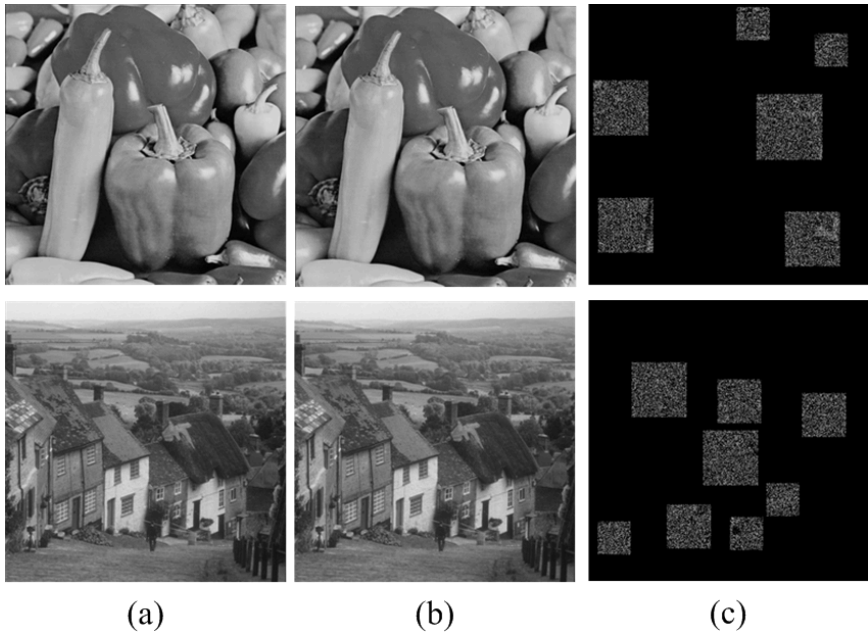


Fig. 5.11. Watermark invisibility of content based embedding. (a) original image, (b) watermarked image, (c) residual image.



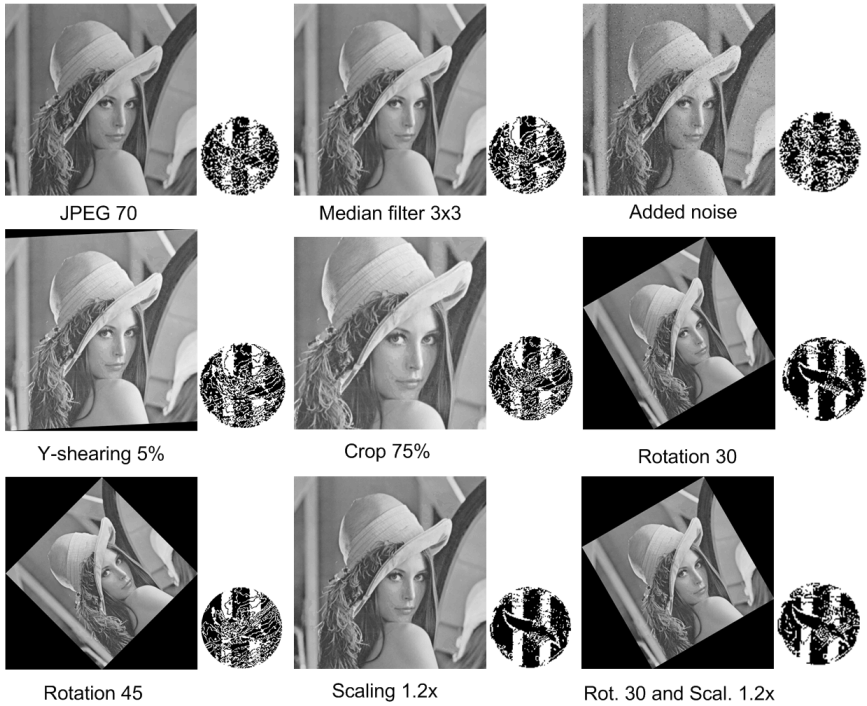
**Fig. 5.12.** Watermark invisibility of partition based embedding. (a) original image, (b) watermarked image, (c) residual image.

Peppers are 41.189 dB and 42.464 dB respectively. For the partition based embedding, the PSNR values of Peppers and House are 48.28dB and 49.49 dB, respectively. In watermarking community, the empirical PSNR value for a good watermarking algorithm should be higher than 40 dB. From this point of view, the PSNR values are satisfactory. Fig. 5.11 and Fig. 5.12 also demonstrate this point, where the embedded watermark is invisible to the naked eyes.

#### 5.4.2 Watermark Robustness

In order to estimate the performances of watermark extraction, both geometric attacks and traditional signal processing attacks are tested on the watermarked images. Then the watermark images are extracted from the distorted images. Geometric attacks consists of rotation, scaling, crop et al., while signal processing attacks consists of noise, filtering and image compression. The simulation results are shown in Fig. 5.13 and Fig. 5.14.

It is observed from the figures that the proposed watermarking schemes can successfully extract the watermark images, regardless of geometric attacks and signal processing attacks. The extracted watermark images are all recognizable. It is also known from Figs. 5.13 and 5.14 that the orientation of the extracted watermark is adaptive to the distorted image. By comparison, the second method can extract the watermark more accurately. The reason is that the pixels in a block is used to encode

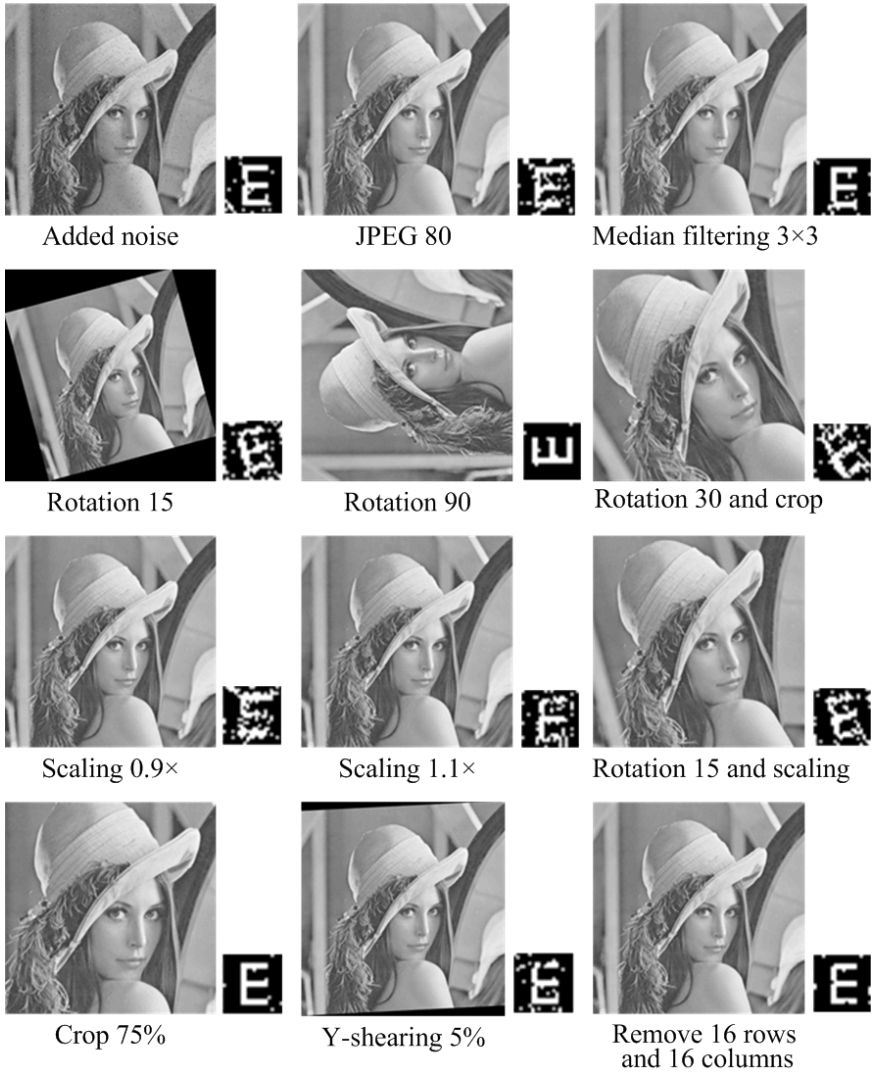


**Fig. 5.13.** The distorted images and the detected watermark images.

a signal watermark bit in the second method, while in the first method a watermark bit is encoded by a single NSCT coefficient. Of course, better robustness is achieved at the cost of reducing watermark capacity. Although both methods employ binary images as the watermark signal, the capacity of the second method is lower than that of the first method.

## 5.5 Conclusion

In this chapter, we introduce two content based watermarking schemes that can embed high capacity watermarking schemes into the feature point based local regions. In order to achieve high capacity information hiding, the watermark should be encoded in a content based manner. Experimental results show that the proposed schemes can achieve good image quality, and the watermark images can be extracted under both common signal processing attacks and geometric attacks. The robustness of the proposed schemes should be further improved for real word applications.



**Fig. 5.14.** The distorted images and the detected watermark images.

*Acknowledgement.* This work is supported by National Natural Science Foundation of China under grant No. 60802077, China Postdoctoral Science Foundation under grant No. 20100471415, and Youth Science Foundation of China University of Mining and Technology under grant No. 2009A022.

## References

1. Al-Qaheri, H., Mustafi, A., Banerjee, S.: Digital watermarking using ant colony optimization in Fractional Fourier Domain. *Journal of Information Hiding and Multimedia Signal Processing* 1, 179–189 (2010)
2. Yang, C., Lin, C., Hu, W.: Reversible data hiding by adaptive IWT-coefficient adjustment. *Journal of Information Hiding and Multimedia Signal Processing* 2, 24–32 (2011)
3. Lin, C., Yang, C.: Multipurpose watermarking based on blind vector quantization (BVQ). *Journal of Information Hiding and Multimedia Signal Processing* 2, 239–249 (2011)
4. Licks, V., Jordan, R.: Geometric attacks on image watermarking systems. *IEEE Multimedia* 12, 68–78 (2005)
5. Zheng, D., Liu, Y., Zhao, J., El-Saddik, A.: A survey of RST invariant image watermarking algorithms. *ACM Computing Surveys* 39, 1–89 (2007)
6. Lichtenauer, J., Setyawan, I., Kalker, T., Lagendijk, R.: Exhaustive geometrical search and the false positive watermark detection probability. In: *Proc. SPIE Conf. on Security and Watermarking of Multimedia Contents*, pp. 203–214 (2003)
7. Pereira, S., Pun, T.: Robust template matching for affine resistant image watermarks. *IEEE Trans. Image Processing* 9, 1123–1129 (2000)
8. Alghoniemy, M., Tewfik, A.: Geometric invariance in image watermarking. *IEEE Trans. Image Processing* 13, 145–153 (2004)
9. Dong, P., Brankov, J.G., Galatsanos, N.P., Yang, Y.Y., Davoine, F.: Digital watermarking robust to geometric distortions. *IEEE Trans. Image Processing* 14, 2140–2150 (2005)
10. Lin, C., Wu, M., Bloom, J., Cox, I., Miller, M., Lui, Y.: Rotation, scale, and translation resilient watermarking of images. *IEEE Trans. Image Processing* 10, 767–782 (2001)
11. O’Ruanaidh, J., Pun, T.: Rotation, scale and translation invariant spread spectrum digital image watermarking. *Signal Processing* 63, 303–317 (1998)
12. Zheng, D., Zhao, J., ElSaddik, A.: RST-invariant digital image watermarking based on log-polar mapping and phase correlation. *IEEE Trans. Circuits and Systems for Video Technology* 13, 753–765 (2003)
13. Kim, H., Lee, H.: Invariant image watermark using Zernike moments. *IEEE Trans. Circuits and Systems for Video Technology* 13, 766–775 (2003)
14. Li, L., Guo, B., Shao, K.: Geometrically robust image watermarking using scaleinvariant feature transform and Zernike moments. *Chinese Optics Letters* 5, 332–335 (2007)
15. Xin, Y., Liao, S., Pawlak, M.: Circularly orthogonal moments for geometrically robust image watermarking. *Pattern Recognition* 40, 3740–3752 (2007)
16. Zhang, L., Qian, G.B., Xiao, W.W., Ji, Z.: Geometric invariant blind image watermarking by invariant Tchebichef moments. *Optics Express* 15, 2251–2261 (2007)
17. Zhang, L., Xiao, W.W., Ji, Z.: Local affine transform invariant image watermarking by Krawtchouk moment invariants. *IET Information Security* 1, 97–105 (2007)
18. Kutter, M., Bhattacharjee, S., Ebrahimi, T.: Towards second generation watermarking schemes. In: *Proc. Int’l. Conf. Image Processing*, vol. 1, pp. 320–323 (1999)
19. Bas, P., Chassery, J., Macq, B.: Geometrically invariant watermarking using feature points. *IEEE Trans. Image Processing* 11, 1014–1028 (2002)

20. Lee, H., Kim, H., Lee, H.: Robust image watermarking using local invariant features. *Optical Engineering* 45, 037002-1–037002-11 (2006)
21. Li, L., Guo, B.: Localized image watermarking in spatial domain resistant to geometric attacks. *International Journal of Electronics and Communications* 63, 123–131 (2009)
22. Qi, X., Qi, J.: A robust content-based digital image watermarking scheme. *Signal Processing* 87, 1264–1280 (2007)
23. Seo, J., Yoo, C.: Image watermarking based on invariant regions of scale-space representation. *IEEE Trans. Signal Processing* 54, 1537–1549 (2006)
24. Tang, C., Hang, H.: A feature-based robust digital image watermarking scheme. *IEEE Trans. Signal Processing* 51, 950–959 (2003)
25. Wang, X., Wu, J., Niu, P.: A new digital image watermarking algorithm resilient to desynchronization attacks. *IEEE Trans. Information Forensics and Security* 2, 655–663 (2007)
26. Harris, C., Stephens, M.: A combined corner and edge detector. In: *Proc. 4th Alvey Vision Conference*, pp. 147–151 (1988)
27. Mikolajczyk, K., Schmid, C.: Scale and affine invariant interest point detectors. *International Journal of Computer Vision* 60, 63–86 (2004)
28. Lowe, D.: Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60, 91–110 (2004)
29. Li, L., Yuan, X., Lu, Z., Pan, J.: Rotation invariant watermark embedding based on scale-adapted characteristic regions. *Information Sciences* 180, 2875–2888 (2010)



---


## Single Bitmap Block Truncation Coding of Color Images Using Cat Swarm Optimization

Shi-Yu Cui<sup>1</sup>, Zhi-Hui Wang<sup>1</sup>, Pei-Wei Tsai<sup>3</sup>, Chin-Chen Chang<sup>4</sup>, and Shuai Yue<sup>1</sup>

<sup>1</sup> Department of Software,  
Dalian University of Technology,  
DaLian, China  
cuishiyu0523@gmail.com, wangzhihui1017@gmail.com,  
peaceful1207@gmail.com

<sup>2</sup> Department of Electronic Engineering,  
National Kaohsiung University of Applied Sciences,  
Kaohsiung, Taiwan  
pwtsai@bit.kuas.edu.tw

<sup>3</sup> Department of Information Engineering and Computer Science,  
Feng Chia University,  
Taichung, Taiwan  
alan3c@gmail.com

**Summary.** This chapter proposes a novel color image compression technique based on Block Truncation Coding (BTC) using Cat Swarm Optimization (CSO). The original BTC method compresses a block of a grayscale image into a bitmap and a pair of quantization numbers. For a color image, which is encoded using R, G, and B channels, the BTC method enhances the compression rate using a common bitmap instead of three naturally formed bitmaps. However, it is difficult to find an efficient common bitmap due to its complexity. The CSO algorithm is generated by modeling the behaviors of cats, which show better performance in achieving the goal of finding best solution than other optimization algorithms. In this chapter, we adopt Cat Swarm Optimization (CSO) to search for a near optimal solution for this problem. The experimental results indicate that the proposed method could improve the decompressed BTC color image quality by obtaining a near optimal solution of the common bit map. 

### 6.1 Introduction

Simple and fast, block truncation coding (BTC) [1, 2] is a well-known image compression technique. It divides the image into many fixed blocks for processing and returns two mean pixel values and a bitmap for one block as the results. The pixels of each block are separated into high and low levels by a threshold  $x$ , which is the

---

<sup>1</sup> Supported by the Fundamental Research Funds for the Central Universities.

average of all pixels in the block. If a pixel value is larger than  $x$ , it is added to high level group, and the corresponding element in the bitmap is valued as “high”; otherwise, it belongs to low level group, and the corresponding element in bit map is valued as “low”. Lastly, it calculates the  $x_L$  and  $x_H$ , which are the mean values of all pixels belonging to the low or high level group, respectively.

The two mean values and one bitmap are the representation for one block of an image. For instance, the bit map shown in Fig. 6.1(b) is calculated from the block shown in Fig. 6.1(a). The  $x$  value is calculated as:

$$x = \frac{(138+135+125+107+122+127+125+111+80+95+109+103+77+72+72+78)}{16} = 104.75.$$

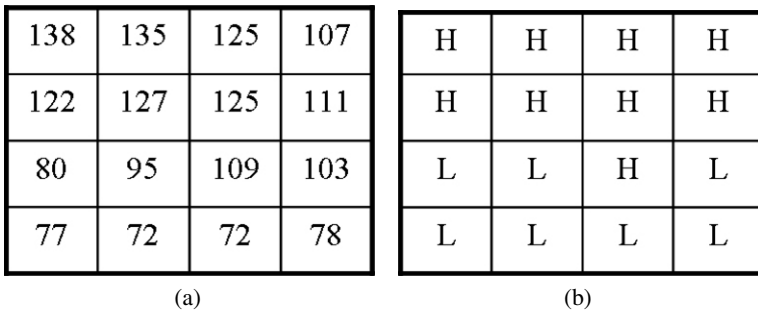
The mean value for the low level group  $x_L$  is computed as:

$$x_L = \frac{(80 + 95 + 103 + 77 + 72 + 72 + 78)}{7} = 82.4.$$

The mean value for high level group  $x_H$  is listed as:

$$x_H = \frac{(138 + 135 + 125 + 107 + 122 + 127 + 125 + 111 + 109)}{9} = 122.1.$$

The BTC method can also apply to color image compression, because each channel in the color image corresponds to a grayscale image. However, finding a suitable common bitmap to represent three different bitmaps generated from different channels has become an important problem.



**Fig. 6.1.** (a) A block of a gray level image. (b) The bitmap calculated from Fig. 6.1(a).

In past decades, many image compression techniques based on BTC technique have been proposed. To retain the image details of the content, Wu et al. [3] proposed an improved BTC scheme for color images. However, this approach is time-consuming. Kurita et al. [4] proposed another BTC method using Principal Component Analysis (PCA) theory that requires small computational load and little memory. Tai et al. [5] also proposed a neural network based BTC scheme for

color image compression. To solve the problem of finding an optimal bitmap and obtaining a high compression rate, Tai et al. [6] proposed a method using a genetic algorithm. Chang and Wu [7] presented an improved scheme for Tai et al.'s genetic method [6], which limits the mean square error and enhances the compression rate with less computational complexity.

In order to obtain a better common bitmap for BTC based color image compression than related schemes, we propose a novel image compression technique that manipulates cat swarm optimization strategy. We treat each channel of a color image as a gray level image and combine the three bitmaps that converted from three channels to generate a common bitmap as the initial bitmap of selection process. Lastly, we find the optimal common bitmap using the cat swarm strategy in limited iterations. According to the experimental results, our proposed scheme is superior to related methods in terms of visual quality.

The rest of this chapter is organized as follows. Section 6.2 introduces the cat swarm optimization (CSO) algorithm, which is used for finding the optimal common bitmap. The proposed scheme, namely CSO-BTC, is presented in Section 6.3. Section 6.4 presents the experimental results of the proposed method, including the comparison with related schemes. Finally, conclusions are provided in Section 6.5.

## 6.2 Related Works

In this chapter, the cat swarm optimization algorithm is adopted for selecting a near optimal common bitmap from three different bitmaps generated from three channels R, G, and B. The CSO algorithm is introduced in this section.

Cat swarm optimization (CSO) was first proposed by Chu and Tsai in 2007 [8]. As with many main ideas of computational intelligence, which portray behavior as formulas, CSO is generated by observing the behaviors of cats. Cats have their own behavioral traits, which are classified into two distinctive features: strong curiosity for moving objects and natural hunting skills. Cats spend most of their time either observing their surroundings or hunting some object until it is captured. According to this principle, these two behaviors are modeled for CSO: seeking mode and tracing mode. The purpose of the seeking mode, in which cats rest but are alert, is to find a better position to move; in the sub-mode of tracing, the cats chase targets. For modeling convenience, each cat has its own attributes: a flag to sign a cat is in seeking mode or tracing mode, its position consists of  $N$  dimensions; the velocity in each dimension, a fitness value (FS) which is used to be evaluated through the fitness function. The detailed procedure of CSO consists of the following steps [8]:

**Step 1:** Create  $K$  cats.

**Step 2:** Sprinkle the cats into the  $N$ -dimensional solution space and give random values to the cats. Ensure that the values are smaller than the maximum velocity. Then, choose a number of cats and set them into tracing mode according to a mixture ratio ( $MR$ ). Set others into seeking mode, where  $MR$  is a given ratio indicating the joining of seeking mode with tracing mode.

**Step 3:** Evaluate the  $FS$  of each cat by putting the current position of each cat into the fitness function. Keep the position of best cat ( $x_{best}$ ), which represents the best solution so far in memory.

**Step 4:** Move the cats according to their flags. If  $cat_n$  is signed in the seeking mode, apply it to the seeking mode; otherwise, apply it to the tracing mode. The details of the seeking mode and tracing mode processes are presented in Steps 5 and 6, respectively.

**Step 5:** First, make  $j$  copies of the present position of  $cat_n$ , where  $j$  equals the seeking memory pool ( $SMP$ ). If the present position is one of the candidates, retain the current position as a candidate and set  $j = (SMP - 1)$ . For each copy, randomly plus or minus the  $SRD$  percents of the present values and replace the old ones according to  $CDC$ , where  $SRD$  is the seeking range of the selected dimension and  $CDC$  is the counts of dimension to change. Next, calculate  $FS$  for each present position. If all  $FS$ s are exactly equal, then set all selecting probability candidates to 1; otherwise, compute the selecting probability of each candidate point by Eq. (6.1), where  $FS_{best}$  is the best fitness value so far. If the purpose of the fitness function is to find the maximum solution, then  $FS_n = FS_{min}$ ; otherwise,  $FS_n = FS_{max}$ . Finally, choose a new candidate point from the candidate points to replace the old position of  $cat_n$  according to the selecting probability.

$$P_n = \frac{|FS_n - FS_{best}|}{FS_{max} - FS_{min}}, \quad \text{where } 0 < n < j. \quad (6.1)$$

**Step 6:** First, update the velocities for every dimension ( $v_{n,d}$ ) according to Eq. (6.2a), where  $x_{best,d}$  is the position of the cat having the best  $FS$ ,  $x_{n,d}$  is the current position of  $cat_n$ ,  $c_1$  is a constant, and  $r_1$  is a random value in the range of  $[0, 1]$ . If new velocities are out of the range of maximum velocity, set them to the limit. Then, update the position of  $cat_n$  according to Eq. (6.2b) and Eq. (6.2b).

$$v_{n,d} = v_{n,d} + r_1 \cdot c_1 \cdot (x_{best,d} - x_{n,d}), \quad d = 1, 2, \dots, M. \quad (6.2a)$$

$$x_{n,d} = x_{n,d} + v_{n,d}. \quad (6.2b)$$

**Step 7:** Pick some cats and apply them into tracing mode according to  $MR$  and set others into seeking mode.

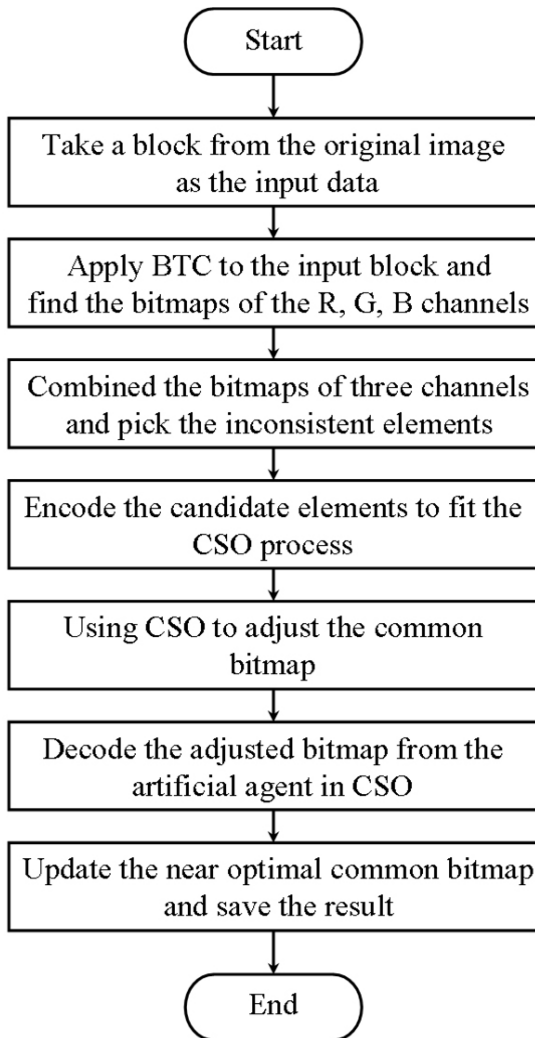
**Step 8:** Repeat Steps 3 through 7 until the termination condition is satisfied.

The experimental results of this method indicate that CSO provides better performance and performs faster than related methods to find the best solutions [8].

### 6.3 Proposed Scheme

This chapter proposes a novel color image compression method based on the cat swarm optimization algorithm, namely, CSO-BTC. The main idea is to create a near

optimal common bitmap as a representation of three single bitmaps using the CSO algorithm. As mentioned before, BTC is a simple and effective image compression method for grayscale images. It divides a gray level image into fixed blocks; for each block, it returns a bitmap and two mean values. To apply this method for color images, the BTC method should be used on three channels. Therefore, a bitmap and two mean values of two levels can be obtained for each channel by following the procedure of BTC. To further compress the three individual bitmaps, we use CSO to find a near optimal common bitmap to represent them. The flowchart of the proposed scheme involves seven steps as shown in Fig. 6.2. The detailed searching phase is described below.



**Fig. 6.2.** The flowchart of proposed scheme.

Let  $B_{best}$ ,  $B_R$ ,  $B_G$  and  $B_B$  denote the near best common bitmap and three bitmaps of the R, G, and B channels, respectively. All bitmaps are sized  $H \times W$ , where  $H$  and  $W$  stand for the height and width of each bitmap, respectively.  $B_{best}$ ,  $B_R$ ,  $B_G$  and  $B_B$  are defined as Eq. (6.3):

$$\begin{aligned} B_{best} &= \{e_1, e_2, \dots, e_{H \times W} \mid e_i = L \text{ or } H\}, \\ B_R &= \{r_1, r_2, \dots, r_{H \times W} \mid r_i = L \text{ or } H\}, \\ B_G &= \{g_1, g_2, \dots, g_{H \times W} \mid g_i = L \text{ or } H\}, \\ B_B &= \{b_1, b_2, \dots, b_{H \times W} \mid b_i = L \text{ or } H\}. \end{aligned} \quad (6.3)$$

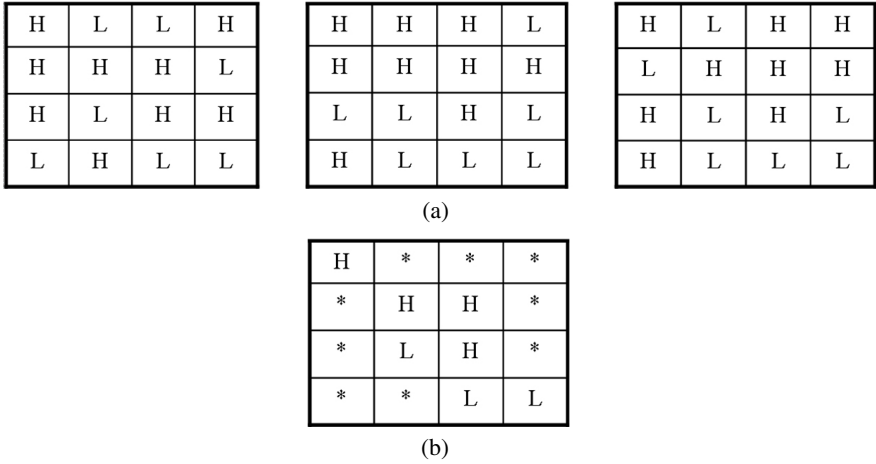
If an element appears in the same coordinate of  $B_R$ ,  $B_G$  and  $B_B$ , it also appears in the same coordinate of  $B_{best}$ . Thus, we can obtain some parts of  $B_{best}$  by adding the elements that appear in the same coordinates of the three bitmaps mentioned above. For instance, the bitmaps of three channel blocks are shown in Fig. 6.3(a). All of the first elements in  $B_R$ ,  $B_G$  and  $B_B$  exactly equal H, so the first element in  $B_{best}$  is determined to be H. In contrast, the second element of  $B_{best}$  is unknown, because the second elements in  $B_R$ ,  $B_G$  and  $B_B$  are L, H and L, respectively. Let \* denote each unknown element in  $B_{best}$ . The initial bitmap  $B_{best}$  is combined by three bitmaps as shown in Fig. 6.3(b). Then the initial value set of  $B_{best}$  is represented as Eq. (6.4):

$$e_i = \begin{cases} b_i & \text{if } r_i = g_i = b_i, \\ \text{unknow} & \text{otherwise.} \end{cases} \quad (6.4)$$

Then, we attempt to find the near optimal common bitmap using the CSO algorithm. The initial bitmap is obtained by combining the three bitmaps R, G, and B as described before. Let  $m$  denote the number of inconsistent elements of the initial bitmap; the unknown elements are composed of  $m$  dimensions. Instead of using a common encoding method in the selection process, we adopt a novel method of the initial input for the CSO algorithm. The details of the two encoding types are presented as follows.

Let 0 and 1 denote the values L and H, respectively; the initial value for each dimension of unknown elements is randomly generated as 0 or 1. In this case, the initial value set  $I$  is represented as  $I = \{s_i \mid s_i \in \{0, 1\}, \text{ for } i = 1, 2, \dots, m\}$ .

In the common encoding method, the initial data is the value sequence of unknown elements directly. For instance, assume that the initial value of the initial



**Fig. 6.3.** (a) The bitmaps of three channels of one image block. (b) The initial common bitmap block, where \* means the unknown elements.

common bitmap block shown in Fig. 6.3(b) is randomly generated as “HHLH-LLLHL”. Then the input data for CSO algorithm is “110100010” since we denote H and L as 1 and 0, respectively.

In the proposed encoding method, we express the solution space into a [0, 1] space no matter how many dimensions the original solution space has. Thus, the unknown elements are encoded into a float number in the range of [0, 1] as input data for the CSO algorithm. As a result, the dimension of the cat is fixed. Let  $I'$  denote a transformed float value expressed in the range of [0, 1]. Then,  $I'$  is converted from those k-bit binary data of  $I$  as Eq. (6.5). For instance, assume that the number of unknown elements equals 6 and the binary data is generated as “101101”. The initial value  $I'$  is computed as

$$\frac{(1 \times 25 + 0 \times 24 + 1 \times 23 + 1 \times 22 + 0 \times 21 + 1 \times 20)}{(26 - 1)} = 0.714286,$$

which is the input data of the CSO algorithm.

$$I' = \frac{\sum_{i=1}^m s_i \times 2^{m-i}}{(2^m - 1)} \quad \text{for } i = 1, 2, \dots, m. \tag{6.5}$$

The decoding process is the inverse process of encoding. If the near optimal result of the CSO algorithm is 0.3155, then the result is decoded as 19.8765, which equals approximately 20. The values of the unknown elements equal “010100”, which are the binary numbers converted from 20. Then, the coordinates of \* in the final bitmap are filled with L, H, L, H, L, and L, which correspond to the binary values 0, 1, 0, 1, 0, and 0.

Peak signal-to-noise ratio (PSNR) is generally used to judge the perceptual quality between the original image and the compression image. The PSNR value for the color image is calculated using Eq. (6.6), where MSE is the mean square error:

$$\text{PSNR(dB)} = 10 \log_{10} \left( \frac{255^2}{\text{MSE}} \right). \quad (6.6)$$

Since the smaller the MSE is, the better PSNR will be, the evaluate function of the selection process follows the principle of the equation defined as below:

$$\text{MSE} = \sum_{e_i=H} (x_i - x_H)^2 + \sum_{e_i=L} (x_i - x_L)^2, \quad (6.7)$$

where  $i = 1, 2, \dots, n$ ,  $x_i$  denotes the  $i^{\text{th}}$  pixel value of a block of channel R, G or B. Moreover,  $x_H$  and  $x_L$  are represented as below, where  $q$  is the number of the elements that equals H in a block, and  $p$  is the number of L's in a block.

$$x_H = \frac{\sum_{e_i \in H} x_i}{q}, \quad (6.8a)$$

$$x_L = \frac{\sum_{e_i \in L} x_i}{p}. \quad (6.8b)$$

With either encoding type, the processing steps in the CSO algorithm are nearly the same except that the number of dimensions is set as 1 in the proposed encoding process. Detailed procedures of the CSO algorithm are described as follows.

First, the cats are divided into two parts according to  $MR$  for two sub-modes in each iteration. Next, every cat in seeking mode makes  $j$  copies including itself according to  $SMP$ . For each copy, randomly plus or minus  $SRD$  percents the present values and replace the old ones according to  $CDC$ , where  $SRD$  is the seeking range of the selected dimension and  $CDC$  is the count of dimensions to change. Calculate  $FS$  and selecting probability of each copy by Eqs. (6.7) and (6.1), respectively, where  $FS_b = FS_{max}$ . The goal of the selecting process is to find the minimum solution, which means that the cat having the minimum MSE is nearest to the near optimal solution. Thus, we retain the cat with the maximum selecting probability. The values should be decoded into a sequence of binary data before computing  $FS$  when using the proposed encoding type. The decoding method is the same as described previously.

The cats in tracing mode simultaneously change their current values according to the velocities in every dimension. The steps described above are repeated until the maximum iteration has been reached. The near optimal common bitmap can be obtained by decoding the adjusted bitmap from the artificial agent in CSO.

According to the near optimal bitmap, the mean values of the R, G and B channels are acquired from  $B_{best}$ , which are denoted as  $L_R$ ,  $H_R$ ,  $L_G$ ,  $H_G$ ,  $L_B$  and  $H_B$ . The calculation functions are defined as follows, where  $P_{best}$  is the number of L's in  $B_{best}$ :



$$L_R = \frac{\left( \sum_{r_i \in B_R \text{ and } r_i=L} r_i \right)}{p_{best}}, \quad (6.9a)$$

$$H_R = \frac{\left( \sum_{r_i \in B_R \text{ and } r_i=H} r_i \right)}{\left( (W \times H) - p_{best} \right)}, \quad (6.9b)$$

$$L_G = \frac{\left( \sum_{g_i \in B_G \text{ and } g_i=L} g_i \right)}{p_{best}}, \quad (6.9c)$$

$$H_G = \frac{\left( \sum_{g_i \in B_G \text{ and } g_i=H} g_i \right)}{\left( (W \times H) - p_{best} \right)}, \quad (6.9d)$$

$$L_B = \frac{\left( \sum_{b_i \in B_B \text{ and } b_i=L} b_i \right)}{p_{best}}, \quad (6.9e)$$

$$H_B = \frac{\left( \sum_{b_i \in B_B \text{ and } b_i=H} b_i \right)}{\left( (W \times H) - p_{best} \right)}. \quad (6.9f)$$

## 6.4 Experimental Results

In this section, the proposed scheme's experimental results are presented. Four color images of size  $512 \times 512$  were used as shown in Fig. 6.4. The block sizes for each image were set as  $4 \times 4$  and  $8 \times 8$ , respectively. The experiments were run on an Intel Core2 2.4G CPU with the GCC compiler for C++ with a FreeBSD 7.1-Release operating system. The mean square error that evaluates the image quality is defined as follows:

$$\text{MSE} = \frac{\sum_{i=1}^n \sum_{j=1}^n \left[ (R_r - I_r)^2 + (R_g - I_g)^2 + (R_b - I_b)^2 \right]}{512 \times 512 \times 3}, \quad (6.10)$$

where the  $R_r, R_g, R_b$  and  $I_r, I_g, I_b$  represent the reconstructed R, G, and B channel values from the optimal common bitmap and the original R, G and B channel values, respectively. As a result, the peak signal-to-noise ratio (PSNR) can be computed by Eq. (6.6).



**Fig. 6.4.** Four color images of size  $512 \times 512$  for experiments.

In our experiments, we implement the proposed scheme on two encoding types as described in Section 6.3. Let types 1 and 2 denote the scheme using common and the proposed encoding methods, respectively. For CSO in types 1 and 2, we set the parameters as shown in Tables 6.1 and 6.2, respectively. For both encoding types, we apply 20 iterations to search for the near optimal solution with 16 cats in each iteration. For encoding type 1, the dimension is set to be  $m$ , and the solution constraint is  $[0, m]$ , which is the number of inconsistent elements of the initial bitmap. The value in every dimension is rounded off to the unit. The values of  $SMP$ ,  $CDC$ ,  $SRD$ ,  $Con_1$ , and  $MR$  are listed in Table 6.1. For encoding type 2, the solution constraint is  $[0, 1]$ , and the dimension is set to be 1. In seeking mode,  $SMP$  and  $CDC$  are set to be 3 and 0.02, respectively. In tracing mode, the  $con_1$  is set to be 2, and the maximum velocity is set to be 1. The ratio between seeking mode and tracing mode is set to be 8 : 2, which means that  $MR$  is 0.2. Figs. 6.5 and 6.6 show the visual quality performance when using different encoding methods.

**Table 6.1.** Parameters setting for CSO in encoding type 1.

| Parameter           | Value |
|---------------------|-------|
| Iteration number    | 20    |
| Population size     | 16    |
| Dimension           | $m$   |
| <i>SMP</i>          | 5     |
| <i>CDC</i>          | 0.8   |
| <i>SRD</i>          | 0.2   |
| $con_1$             | 2.0   |
| <i>MR</i>           | 0.2   |
| <i>Max.Velocity</i> | 1.0   |

**Table 6.2.** Parameters setting for CSO in encoding type 2.

| Parameter           | Value |
|---------------------|-------|
| Iteration number    | 20    |
| Population size     | 16    |
| Dimension           | 1     |
| <i>SMP</i>          | 3     |
| <i>CDC</i>          | 1     |
| <i>SRD</i>          | 0.02  |
| $con_1$             | 2.0   |
| <i>MR</i>           | 0.2   |
| <i>Max.Velocity</i> | 1.0   |

Tables 6.3 and 6.4 give the experimental results using different encoding methods, where the blocks for tested images are sized  $4 \times 4$  and  $8 \times 8$ , respectively. It seems that the original coding scheme performs better only when the block size is  $8 \times 8$  as in the last three images.

The reason that the MSE is smaller when the block size is  $4 \times 4$  in Table 6.3 is that the scale of distortion on the reconstructed pixels is smaller than the larger block size, which reduces the MSE. Compared with the common encoding scheme, the solution space of the proposed encoding scheme, which is encoded into a single value in the range of  $[0, 1]$ , is smaller. Thus, the scheme using the proposed encoding method performs better than the scheme using common encoding method when block size is  $4 \times 4$ . However, the proposed encoding method also causes CSO to be more sensitive to change, because the encoding space is not increased accordingly as shown in Table 6.4.



PSNR=32.550



PSNR=32.569



PSNR=31.673

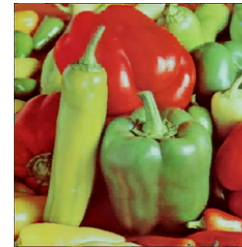


PSNR=32.866

(a)



PSNR=29.695



PSNR=27.688



PSNR=29.085



PSNR=30.034

(b)

**Fig. 6.5.** (a) Visual quality performance for CSO-BTC type 1 ( $4 \times 4$  block size). (b) Visual quality performance for CSO-BTC type 1 ( $8 \times 8$  block size).



PSNR=32.773



PSNR=33.018



PSNR=31.776



PSNR=33.170

(a)



PSNR=29.735



PSNR=27.245



PSNR=28.740



PSNR=29.655

(b)

**Fig. 6.6.** (a) Visual quality performance for CSO-BTC type 2 ( $4 \times 4$  block size). (b) Visual quality performance for CSO-BTC type 2 ( $8 \times 8$  block size).

**Table 6.3.** Image quality performance of two encoding methods with  $4 \times 4$  block size.

| Image<br>$512 \times 512$ | CSO-BTC Type 1<br>MSE | CSO-BTC Type 2<br>MSE |
|---------------------------|-----------------------|-----------------------|
| Watch                     | 36.150                | 34.338                |
| Pepper                    | 35.994                | 32.454                |
| Goldhill                  | 44.234                | 43.196                |
| Fruits                    | 33.612                | 31.337                |

**Table 6.4.** Image quality performance of two encoding methods with  $8 \times 8$  block size.

| Image<br>$512 \times 512$ | CSO-BTC Type 1<br>MSE | CSO-BTC Type 2<br>MSE |
|---------------------------|-----------------------|-----------------------|
| Watch                     | 69.753                | 69.113                |
| Pepper                    | 110.717               | 122.635               |
| Goldhill                  | 80.274                | 86.907                |
| Fruits                    | 64.521                | 70.405                |

We compare the proposed scheme with three other methods: W-plane BTC, GA-AMBTC, and GSBTC. The original BTC is also listed for comparison, because it provides the criteria of the original compressed results. Tables 6.5 and 6.6 give the experimental results for a variety of different methods, where the blocks for tested images are sized as  $4 \times 4$  and  $8 \times 8$ , respectively. The results obtained by GA-AMBTC and CSO-BTC are averaged by the outcomes of the 30 runs, each of which contains 20 iterations. Figs. 6.7 through 6.10 indicate the visual quality performance for test images using BTC, W-plane BTC, GA-AMBTC, and GSBTC, respectively.

Compared with the related schemes in Table 6.5, the proposed scheme can achieve a lower MSE value for each test image. These results show that the proposed scheme performs superior in terms of image quality than W-plane BTC, GA-AMBTC, and GS BTC with  $4 \times 4$  blocks. Table 6.6 reveals that the MSE values are better for the  $8 \times 8$  images “Watch” and “Goldhill” than in W-plane BTC, GA-AMBTC and GSBTC but worse for the  $8 \times 8$  image “Pepper” than in GA-AMBTC and GSBTS. For image “Fruits”, the MSE value in Table 6.6 is lower than in GSBTS but larger than in GA-AMBTC. This means that the proposed scheme provides better performance with smaller block size, because its solution space is controlled by the novel encoding method; however, this also causes the scheme to be more sensitive.



PSNR=32.763



PSNR=33.306



PSNR=32.204



PSNR=33.436

(a)



PSNR=29.819



PSNR=28.677



(b)

**Fig. 6.7.** (a) Visual quality performance for BTC method ( $4 \times 4$  block size). (b) Visual quality performance for BTC method ( $8 \times 8$  block size).



PSNR=32.622



PSNR=32.845



PSNR=31.512

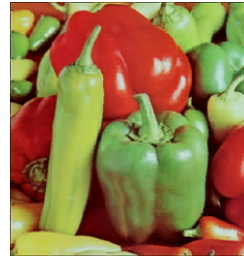


PSNR=32.902

(a)



PSNR=29.662



PSNR=28.234



PSNR=28.715



PSNR=29.821

(b)

**Fig. 6.8.** (a) Visual quality performance for W-plane BTC ( $4 \times 4$  block size). (b) Visual quality performance for W-plane BTC ( $8 \times 8$  block size).





PSNR=32.566



PSNR=32.559



PSNR=31.320



PSNR=32.693

(a)



PSNR=29.651



PSNR=27.504



PSNR=28.711



PSNR=29.673

(b)

**Fig. 6.9.** (a) Visual quality performance for GA-AMBTC ( $4 \times 4$  block size). (b) Visual quality performance for GA-AMBTC ( $8 \times 8$  block size).



PSNR=32.621



PSNR=32.440



PSNR=31.338

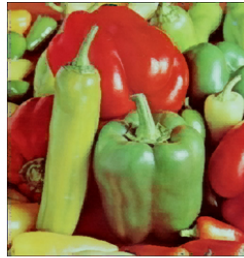


PSNR=32.667

(a)



PSNR=29.659



PSNR=27.737



PSNR=28.591



PSNR=29.647

(b)

**Fig. 6.10.** (a) Visual quality performance for GSBTC ( $4 \times 4$  block size). (b) Visual quality performance for GSBTC ( $8 \times 8$  block size).

**Table 6.5.** The MSE and PSNR values for a variety of methods for  $4 \times 4$  block size.

| Image            | BTC    | W-plane BTC | GA-AMBTC | GSBTC  | CSO-BTC | Type 2 |
|------------------|--------|-------------|----------|--------|---------|--------|
| $512 \times 512$ | MSE    | MSE         | MSE      | MSE    | MSE     | MSE    |
| Watch            | 34.416 | 35.550      | 36.013   | 35.558 | 34.338  |        |
| Pepper           | 30.370 | 33.771      | 36.074   | 37.075 | 32.454  |        |
| Goldhill         | 39.142 | 45.910      | 47.980   | 47.780 | 43.196  |        |
| Fruits           | 29.479 | 33.333      | 34.977   | 35.187 | 31.337  |        |

**Table 6.6.** The MSE and PSNR values for a variety of methods for  $8 \times 8$  block size.

| Image            | BTC    | W-plane BTC | GA-AMBTC | GSBTC   | CSO-BTC | Type 2 |
|------------------|--------|-------------|----------|---------|---------|--------|
| $512 \times 512$ | MSE    | MSE         | MSE      | MSE     | MSE     | MSE    |
| Watch            | 67.795 | 70.280      | 70.470   | 70.336  | 69.113  |        |
| Pepper           | 88.179 | 97.657      | 115.536  | 109.488 | 122.635 |        |
| Goldhill         | 76.902 | 87.405      | 87.501   | 89.941  | 86.907  |        |
| Fruits           | 60.191 | 67.767      | 70.114   | 70.527  | 70.405  |        |

## 6.5 Conclusions

In this chapter, we present a color image compression technique using the cat swarm optimization algorithm based on common bitmap block truncation coding. We consider the three channels of the color image as three grayscale images due to the procedure of BTC. To obtain an optimal common bitmap or a near optimal solution as a representation for the color image, the proposed scheme manipulates the cat swarm optimization strategy in the selection process for better image quality. We also adopt a novel encoding method of the initial input for the CSO algorithm to improve the experimental results for smaller block size. The experimental results comparison indicates that our proposed scheme performs superior in terms of visual quality of the compressed image than some related schemes.

## References

1. Delp, E.J., Mitchell, O.R.: Image compression using block truncation coding. *IEEE Trans. Communications* 27, 1335–1342 (1979)
2. Rao, Y.V.R., Eswama, C.: A new algorithm for BTC image bit plane coding. *IEEE Trans. Communications* 43, 2010–2011 (1995)
3. Wu, Y., Coll, D.C.: Single bit-map block truncation coding of color images. *IEEE Journal on Selected Areas in Communications* 10, 952–959 (1992)

4. Kurita, T., Otsu, N.: A method of block truncation coding of color image compression. *IEEE Trans. Communications* 41, 1270–1274 (1993)
5. Tai, S.C., Lin, Y.C., Lin, J.F.: Single bit-map block truncation coding of color images using a Hopfield neural network. *Information Sciences* 103, 211–228 (1997)
6. Tai, S.C., Chen, W.J., Cheng, P.J.: Genetic algorithm for single bitmap absolute moment block truncation coding of color images. *Optical Engineering* 37, 2483–2489 (1998)
7. Chang, C.C., Wu, M.N.: An algorithm for color image compression based on common bit map block. In: *Proc. Second Int'l. Workshop on Intelligent Multimedia Computing and Networking*, pp. 964–967 (2002)
8. Chu, S.C., Tsai, P.W.: Computational intelligence based on the behavior of cats. *International Journal of Innovative Computing, Information and Control* 3, 163–173 (2007)

---

# Application of Genetic-Based Wavelet Packet Watermarking for Copyright Protection

Hsiang-Cheh Huang<sup>1</sup> and Yueh-Hong Chen<sup>2</sup>

<sup>1</sup> National University of Kaohsiung,  
Kaohsiung 811, Taiwan, R.O.C.  
huang.hc@gmail.com

<https://sites.google.com/site/hch888dr/>

<sup>2</sup> Far East University,  
Tainan 744, Taiwan, R.O.C.  
yuehhong@gmail.com

**Summary.** Copyright protection is mostly required for lots of applications, and watermarking is a scheme for making this possible. In this chapter, we propose a genetic watermarking scheme based on the wavelet packet transform (WPT) for the applications of copyright protection and trusted communication. Wavelet packet transform can be regarded as a generalization of the discrete wavelet transform (DWT). For the WPT, a best wavelet basis in the sense of evaluation metric can be found within a large library of permissible bases. In addition, genetic algorithm (GA) is employed to select an appropriate basis from permissible bases of wavelet packet transform to meet the requirements of our watermarking algorithm. After the training with GA, the secret information corresponding to copyright owners can be embedded into original images, and the watermarked image is obtained. During transmission of watermarked images, intentional signal processing is expected, and our watermarking algorithm is considered to keep the integrity for the trusted communication. Experimental results demonstrate that the proposed method can increase the capability to resist intentional signal processing for the use of GA. Moreover, the watermarked image quality can be guaranteed based on the inherent characteristics of the watermarking algorithm. Better performances can also be observed over existing methods in literature, and our algorithm is suitable for the application of copyright protection and trusted communication.

## 7.1 Introduction

Multimedia contents can be easily accessed in our daily lives. For instance, people can capture images or video with digital cameras or mobile phone cameras at any time. Besides, with the development of computer industry and the widespread use of the Internet and wireless networks, a lot of digital media, including image, audio, and video, have been copied, stolen, and altered by anyone easily and unlimitedly. Because these digital media could be acquired quickly and easily over the Internet, how to provide the necessity for copyright protection and trusted communication has

become a more and more important task for both academic researches and industrial applications [1]. Under this scenario, digital watermarking has been regarded as an effective solution [2, 3].

Digital watermarking is a scheme to put the secret information into the multimedia contents. With the aim of copyright protection for watermarking, the secret information, relating to the copyright owners, would be hidden into the digital contents. After embedding the secret information with algorithms designed by researchers, the watermarked contents would be delivered to the receiver. During the delivery, some signal processing, called attacks, might apply to the watermarked contents, and this would make copyright protection unavailable. After reception, people are expected to extract the secret information from the marked contents after attacks. Owner's copyright can be retained once the extracted information can be recognizable, or the correlation between the embedded and extracted information lie above some confidence level.

From practical point of view, taking the characteristics of image contents for instance, by use of altering the pixel values in the spatial domain [4, 5], the codewords in the vector quantization (VQ) domain [6], and the coefficients in the frequency domain, including the discrete cosine transform (DCT) [7, 8] and the discrete wavelet transform (DWT) [9], we observe that these commonly employed techniques could lead watermarking a possible means for copyright protection and trusted communication in literature.

Because digital images are the most commonly acquired media on the web, in this chapter, we focus the multimedia contents on the digital images. Image watermarking is to embed the watermark (or the secret information) into images for copyright protection and trusted communication. The watermarked content needs to be as resemble as its original counterpart as possible. Also, secret information should be extracted at the decoder, which should lie above the threshold of the confidence level to retain the capability for copyright protection. As discussed in literature, a good image watermarking scheme would be expected to have the following characteristics [2, 10]:

1. *High fidelity*: A watermarked image should be perceptually identical or very similar to its original counterpart. The Peak Signal-to-Noise Ratio (PSNR) is generally considered as the representation of fidelity between the original image  $X$ , and watermarked one,  $X'$ . For image size of  $M \times N$ , the PSNR can be calculated by

$$\text{PSNR} = 10 \times \log_{10} \left( \frac{255^2}{\frac{1}{M \times N} \sum_{i=1}^M \sum_{j=1}^N (X(i, j) - X'(i, j))^2} \right). \quad (7.1)$$

Larger PSNR values imply less distortions induced, hence the better outcomes can be reached.

2. *Good robustness*: The watermark can be successfully detected or extracted based on the application of algorithm, even though some intentional attacks, or image processing operations, had been applied on the watermarked image. People often use the bit correct rate (BCR), with the definition in Eq. (7.2), to assess

the robustness of the watermarking algorithm. The normalized cross-correlation (NC) between the embedded watermark and the extracted one may also be considered to assess the survivability for the extracted watermark. Let the embedded and extracted watermarks be  $W$  and  $W'$ , respectively, with length of  $W_N$ . With its ease of calculation, we choose the BCR values to measure the robustness of extracted watermarks. The BCR is

$$\text{BCR} = \left( 1 - \frac{1}{W_N} \sum_{i=1}^{W_N} [W(i) \oplus W'(i)] \right) \times 100\%, \quad (7.2)$$

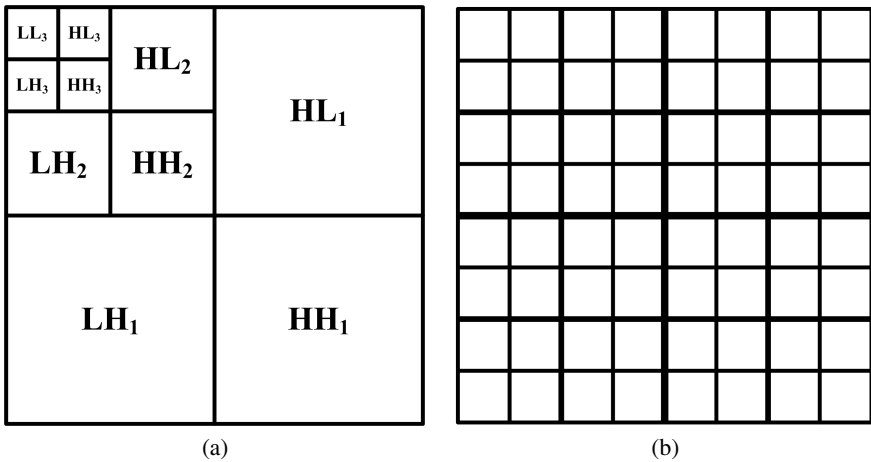
where  $\oplus$  denotes the exclusive-or (XOR) operation. The larger the BCR value, the better the result for copyright protection.

3. *Large data capacity*: The larger the data capacity is, the more secret information can be embedded into original images. On the other hand, if error control codes (ECC) is employed, enhanced performance of extracted information can be increased with the expense of the added redundancy. Larger capacity can reside the produced redundancy, and provide the flexibility for the selection of error control codes.

We can easily see that for obtaining high fidelity, as less alteration to the original image should be performed as possible. Intuitively, in the spatial domain, the secret information can be embedded to the least significant bits, while in the frequency domain, it should be hidden into higher frequency bands due to the energy distribution of frequency coefficients. On the contrary, the commonly employed attacks, JPEG for instance, tend to remove the data at higher frequency bands, causing the secret information vanished. From the first two requirements, there are conflicts between each other. Furthermore, comparing between the first and the third requirements, if more capacity is embedded, more alteration to the original content should be performed, leading to the degradation of output image quality. Following this induction, because conflicts between each other can be easily observed, some compromises among the requirements need to be reached. Therefore, designing an optimal watermarking scheme to meet all the requirements has been a difficult and interesting problem.

Embedding watermarks into images can be referred to as a constrained optimization problem by taking the three requirements mentioned above into consideration. Hence, genetic algorithm (GA) can possibly be used to solving this kind of problem [10]. A detail explanation of GA can be found in [11]. Different from typical watermarking schemes, conventional genetic watermarking looks for the optimal coefficients in frequency domain in the sense of a certain evaluation function to embed watermarks. Under the constraint of keeping these requirements acceptable, and to simplify the optimization problem to some extent, GA can be employed to optimize the robustness of watermarks, the fidelity of watermarked images, or both, under the assumption of a fixed number of capacity for embedding [7]. The maximized PSNR and BCR values corresponding to the optimized combinations of embedding bands can be obtained with GA.

A number of methods have been proposed to embed robust and invisible watermarks. Some operate directly in the spatial domain [5], others in the transform domain, such as Fourier [12], DCT [13], or wavelet domains [14]. Among all methods, wavelet packet transform (WPT) has attracted much attention [15, 16, 17, 18]. WPT can be regarded as a generalization of the discrete wavelet transform. In the usual dyadic wavelet decomposition, as depicted in Fig. 7.1(a), only the low-pass-filtered subband is recursively decomposed and it thus can be represented by a logarithmic tree structure. However, a wavelet packet decomposition, shown in Fig. 7.1(b), allows the process to be represented by any pruned subtree of the full tree topology. Therefore, this representation of the decomposition topology is isomorphic to all permissible subband topologies. The leaf nodes of each pruned subtree represent one permissible orthonormal basis [17]. Thus, a best wavelet basis in the sense of evaluation metric can be found within a large library of permissible bases in WPT [18, 19].



**Fig. 7.1.** Comparisons of frequency domain representations between (a) the three-level decomposition of DWT, and (b) the fully decomposed three-level WPT.

In [20], authors indicated that the watermarking scheme employing the zerotree of WPT provided the best performance in terms of Peak Signal-to-Noise Ratio in comparison with schemes employing DWT and DCT. On the contrary, the robustness against various types of attacks of the scheme employing WPT was somewhat decreased. Therefore, we propose a genetic watermarking scheme based on the wavelet packet transform in this chapter in order to obtain both the better image quality and the more resistance to intentional attacks. Better image quality can be expected with the characteristics of the proposed embedding algorithm in Sec. 7.2 and the more robustness for extracted watermark can be obtained with the training in GA. In the proposed method, GA is employed to select an appropriate basis from permissible bases of WPT to increase the robustness of the embedded watermarks.



Experimental results demonstrate that proposed method can increase the capability to resist image processing methods if an appropriate fitness function of GA is adopted.

## 7.2 Genetic Watermarking in Wavelet Packet Transform

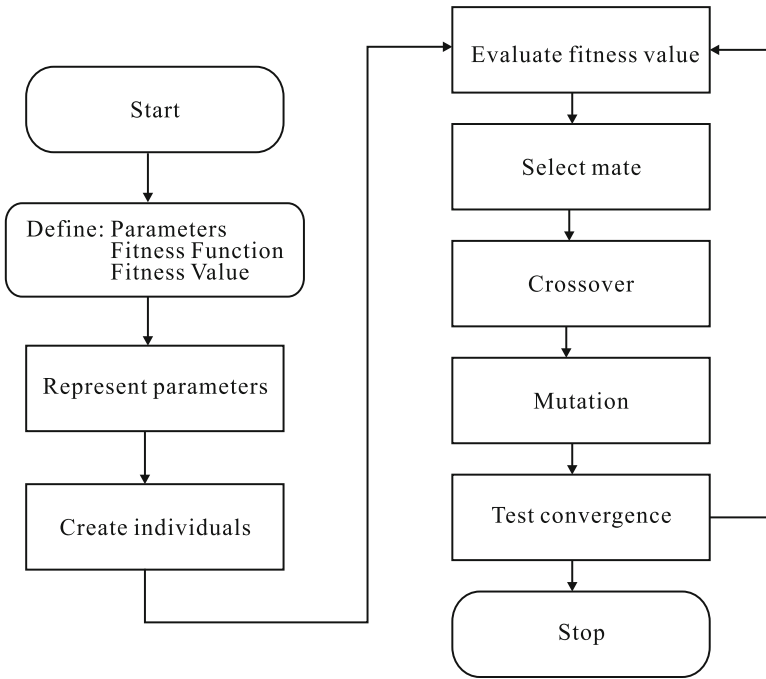
As people know, the goals for DWT and WPT are to get the minimal representation of data relative to particular fitness function. Wavelet transform is applied to low-pass results recursively. In Fig. 7.1(a), it represents the three-level decomposition of DWT. Only the low-pass bands are supposed to be further decomposed. On the contrary, for WPT, both the low-pass and high-pass bands can be selected for further decomposition. Figure 7.1(b) denotes the three-level decomposition of WPT when all the bands are fully decomposed. With our algorithm, we are going to hide our secret message into the selected bands in WPT.

Assuming that the watermark for embedding consists of 0's and 1's, and all bits in the watermark are embedded into an image with the same manner accordingly. To embed the watermark, the genetic algorithm is first used to select an appropriate basis of WPT, and then a number of coefficients are randomly chosen and modified. The random seed and the WPT decomposition tree for watermark embedding play the roles as secret keys. Genetic algorithm for basis selection is described in Sec. 7.2.1 and the method for embedding watermarks into images is introduced in Sec. 7.2.2, respectively. Application aspects and limitations with our algorithm are briefly addressed in Sec. 7.2.3.

### 7.2.1 Best Basis Selection with GA

Conventional search techniques are often incapable of optimizing non-linear functions with multiple variables. One scheme called the “genetic algorithm”, based on the concept of natural genetics, is a directed random search technique. The exceptional contribution of this method was developed by Holland [21] over the course of 1960s and 1970s, and finally popularized by Goldberg [11].

In genetic algorithms, the parameters are represented by an encoded binary string, called the “chromosome.” And the elements in the binary strings, or the “genes,” are systematically adjusted to minimize or maximize the fitness value. The fitness function generates its fitness value, which is composed of multiple variables to be optimized by GA. For every training iteration, a pre-determined number of individuals would correspondingly produce fitness values associated with the chromosomes. Figure 7.2 demonstrates the flow chart for a typical binary GA. It begins by defining the optimization parameters, the fitness function, and the fitness value, and it ends by testing for convergence. According to the applications for optimization, designers need to carefully define the necessary elements for training with GA. Next, we are able to evaluate the fitness function in addition to the terminating criteria with the natural selection, crossover, and mutation operations in a reasonable way. With the aid of Fig. 7.2, the core components are depicted as follows.



**Fig. 7.2.** The flow chat of genetic algorithms

1. *Mate selection.* A large portion of the low fitness individuals is discarded through this natural selection step. Of the  $N$  individuals in one iteration, only the top  $N_{\text{good}}$  individuals survive for mating, and the bottom  $N_{\text{bad}} = N - N_{\text{good}}$  ones are discarded to make room for the new offspring in the next iteration. Therefore, the selection rate is  $p_s = \frac{N_{\text{good}}}{N}$ .
2. *Crossover.* Crossover is the first way that GA explores a fitness surface. Two individuals are chosen from  $N_{\text{good}}$  individuals to produce two new offsprings. A crossover point is selected between the first and last chromosomes of the parents' individuals. Then, the fractions of each individual after the crossover point are exchanged, and two new offspring are produced.
3. *Mutation.* Mutation is the second way that GA explores a fitness surface. It introduces traits not in the original individuals, and keeps GA from converging too fast. The pre-determined mutation rate,  $p_m$ , should be low. It is accomplished by intentionally flipping the value of selected bits based on the mutation rate. Most mutations deteriorate the fitness of an individual, however, the occasional improvement of the fitness adds diversity and strengthens the individual.

Chromosomes of the genetic algorithm used in the proposed scheme consist of a binary string. Meanwhile, referring to Fig. 7.1(b), the length of the binary string,  $L$ , is equal to

$$\begin{aligned}
L &= 1 + 4 + 4^2 + \dots + 4^{D-1} \\
&= \frac{(1 - 4^D)}{(1 - 4)} \\
&= \frac{(4^D - 1)}{3},
\end{aligned} \tag{7.3}$$

where  $D$  is decomposition level of WPT. Bit ‘1’ in the string indicates that the corresponding subband should be decomposed further, and bit ‘0’ denotes the termination of subband decomposition. If the value at index  $i$  is equal to ‘1’, the indices of the resulting four subbands,  $i_m$ , can be derived by Eq. (7.4):

$$i_m = 4 \times i + m, \quad m \in \{1, 2, 3, 4\}. \tag{7.4}$$

An example of the chromosome and its corresponding WPT decomposition tree is shown in Fig. 7.3. Suppose that at most two levels are decomposed, shown in Fig. 7.3(a).  $D = 2$ , meaning that a chromosome is composed of  $\frac{4^2-1}{3} = 5$  bits. At the first level, if bit 0 in Fig. 7.3(b) is ‘1’, meaning that further decomposition is needed, and the values of remaining four bits are determined accordingly.

The fitness function in the proposed scheme is described as follows:

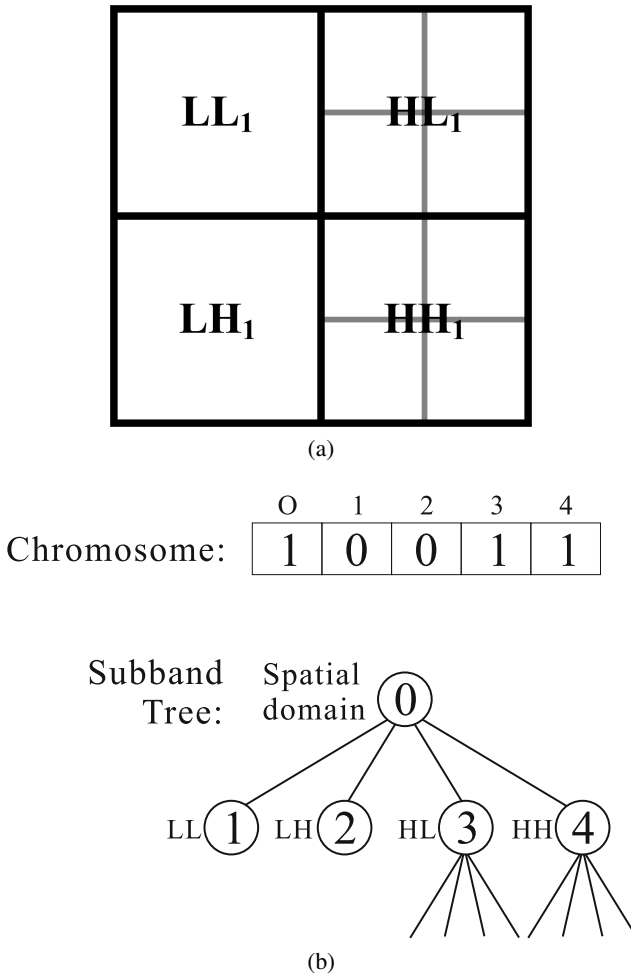
$$\text{fitness value} = \prod_{i=1}^K P_i, \tag{7.5}$$

where  $P_i$  denotes the percentage of the watermark bits that still survive after applying certain attacks, or the BCR in Eq. (7.2), and  $K$  means the number of attack method adopted for robustness evaluation. From Eq. (7.5), we can see that since  $P_i \in [0, 1]$ , the maximum of the possible fitness value would be 1.0. By doing so, all the expected attacks can be conquered because we choose the product of all extracted BCR values for optimization, and each value of  $P_i$  needs to be as large as possible. If the BCR values of extracted watermarks after experiencing some of the attacks get too low, such a kind of watermark embedding fails to exist with the fitness function in our training iterations. Thus, the goal for GA optimization is to search for the maximum of the fitness in Eq. (7.5). Because the watermarked image quality would lead to the optimized PSNR value by using the proposed algorithm, which will be discussed in Sec. 7.2.2, only the robustness of the watermarking algorithm is taken into consideration in Eq. (7.5).

## 7.2.2 Adaptive Watermarking in Wavelet Packet Transform

The method for embedding one binary value into an image is the extension to the method in [22]. When one bit of the watermark is embedded, a pre-specified number of coefficients are chosen randomly. These coefficients are then modified such that the first one, in the order of being chosen, is the largest if a ‘1’ is embedded. If a ‘0’ is embedded, the coefficients should be modified such that the first one is the smallest.

Due to its simplicity in implementation, authors are suggested to refer to [22] in watermark embedding and extraction for more details.



**Fig. 7.3.** An example for our algorithm. (a) The representation of decomposed subbands in WPT. (b) The binary representation of chromosome and its corresponding subband tree.

### 7.2.3 Limitations and Flexibilities with the Proposed Method

Our method aims at obtaining both the better image quality of watermarked image, and the enhanced result in watermarking robustness. For achieving the goals, we employ GA for selecting suitable bands with WPT. For training with GA, it takes some time to calculate the results in each training iteration, and hence our method may have limitations for real-time processing. This issue is addressed in Sec. 7.3 shortly. For the offline applications such as archival of multimedia with trusted communication, our method may work well.

The proposed method also offers flexibilities for users to obtain the optimized result. For instance, the levels of decomposition, the compositions of elements in

the fitness function, or the crossover and mutation rates in GA, are flexible to meet the users' needs.

### 7.3 Experimental Results

In this section, we present some experimental results to demonstrate the performance of the proposed method from three aspects:

1. the output image quality of the watermarked image,
2. the robustness of watermarking algorithm after selected attacks, and
3. the time consumption for completing the training procedures.

Two  $512 \times 512$ , 8 bit/pixel gray scale images, Lena and F16, are employed as the test images. A watermark, 1000-bit in length, is the randomly generated bitstream. The number of chromosomes is 200, and the maximal decomposition level is set to  $D = 8$ , meaning that the length of the chromosome is  $\frac{4^8-1}{3} = 21845$  bits. Additionally, selection and mutation rates are set to  $p_s = 0.5$  and  $p_m = 0.04$ , respectively, and the number of training iteration is set to 450. Three image processing operations, including JPEG compression with quality factor = 30, Gaussian filtering, and sharpening, are applied on the watermarked images to evaluate fitness value of the chromosomes. The mask for  $3 \times 3$  Gaussian filtering can be addressed with Eq. (7.6):

$$\begin{bmatrix} \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \\ \frac{2}{16} & \frac{4}{16} & \frac{2}{16} \\ \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \end{bmatrix}. \quad (7.6)$$

In comparison with [22], the watermark could only be embedded into subbands decomposed from *LH2*, *HL2* and *HH2* in Fig. 7.3. Note that parameters employed in the watermarking scheme here are adopted from those in [22].

With the best basis evolved by GA, the watermarked images are demonstrated in Fig. 7.4 for subjective evaluation. Objective quality, represented by PSNR, are 44.80 and 42.27 dB, respectively. The image quality of both images should be acceptable both subjectively and objectively. Next, robustness in this paper is represented by bit correct rate (BCR). These results of applying three image processing operations in Fig. 7.4, including JPEG with quality factor (QF) of 30, the  $3 \times 3$  Gaussian filtering, and image sharpening, are represented in Table 7.1 for Lena, and Table 7.2 for F16, respectively.

We can see that the BCR values are high enough to make copyright protection possible with WPT. Moreover, results with the proposed algorithm are better than those in [22], meaning that watermarking with WPT and GA has the potential to overcome conventional schemes. It is because a proper basis is selected by GA aiming at the three image processing methods. Thus, for specific image processing methods, the proposed method can generate a best WPT basis to increase the robustness of the watermark.

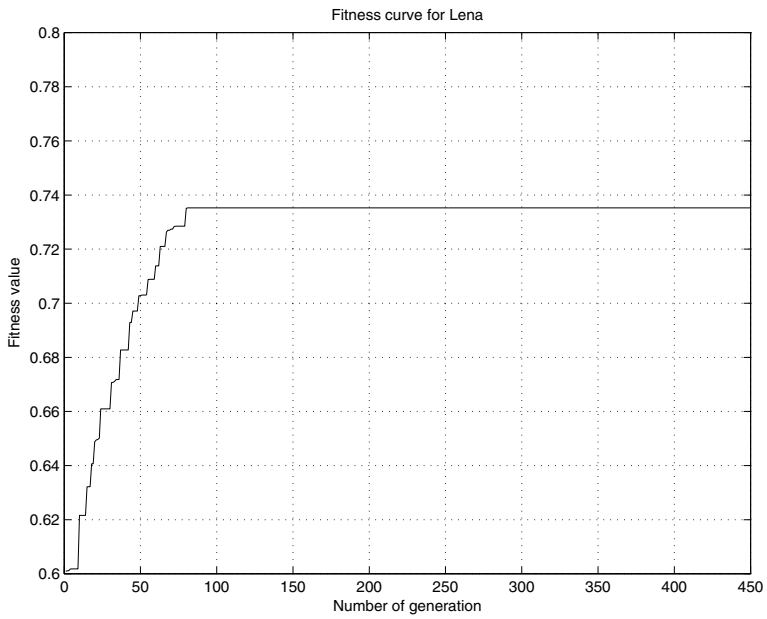


(a)

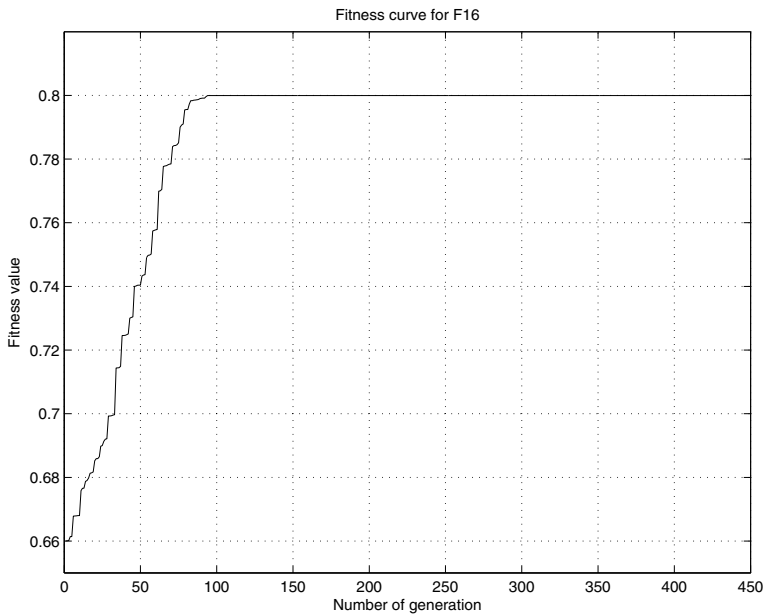


(b)

**Fig. 7.4.** The watermarked images with the proposed GA watermarking scheme. (a) Watermarked *Lena*, with the objective quality of 44.80 dB. (b) Watermarked *F16*, with the objective quality of 42.27 dB.



(a)



(b)

**Fig. 7.5.** The curve of fitness value vs. the number of generation. (a) With watermarked Lena. (b) With watermarked F16.

**Table 7.1.** The bit correct rates for extracted watermark after applying three different attacks on the watermarked Lena image.

| Attack method      | Results with proposed method | Results in [22] |
|--------------------|------------------------------|-----------------|
| JPEG (Q=30)        | 83.4%                        | 76.6%           |
| Gaussian Filtering | 88.0%                        | 83.1%           |
| Sharpening         | 99.8%                        | 99.7%           |

**Table 7.2.** The bit correct rates for extracted watermark after applying three different attacks on the watermarked F16 image.

| Attack method      | Results with proposed method | Results in [22] |
|--------------------|------------------------------|-----------------|
| JPEG (Q=30)        | 87.1%                        | 81.4%           |
| Gaussian Filtering | 90.0%                        | 84.8%           |
| Sharpening         | 99.8%                        | 99.8%           |

The fitness values in the GA training process can be evaluated in Fig. 7.5(a) for Lena, and Fig. 7.5(b) for F16, respectively. We observe that both curves are non-decreasing functions, with the maximum values of 0.73526 in Fig. 7.5(a), and 0.79997 in Fig. 7.5(b). Because the fitness value is the product of survival probabilities in Eq. (7.5), results in Fig. 7.5(a) and (b) seem reasonable. From the simulation results, we can observe the advantages with the proposed algorithm.

Finally, as described in Sec. 7.2, GA is mainly used to optimize non-linear functions with multiple variables. These optimization problems are usually difficult to solve directly. Therefore, comparing to other algorithms, GA often outputs promising solutions with more time consumption. From the viewpoints in theory or in practice, an analysis on time complexity of a complicated algorithm could help users to select proper algorithms for their applications. Since the fitness function used by our method are, however, composed of several image processing operations or attacks, these attacks adopted would have a significant impact on the performance of the watermark embedding process. Besides, the proposed method is performed in



**Table 7.3.** The maximum, average, and minimum duration among all the 450 iterations for GA training with our method for Lena and F16 images.

| Duration (in sec.) | Lena   | F16    |
|--------------------|--------|--------|
| Maximum            | 257.48 | 274.53 |
| Average            | 233.81 | 246.36 |
| Minimum            | 209.27 | 221.18 |

the wavelet packet domain; therefore, the average decomposition levels of wavelet packet trees generated with GA chromosomes would also have an effect on execution time. Thus, it is difficult to evaluate time complexity of the proposed method theoretically.

In this chapter, the execution time of all the 450 iterations of GA in our experiments has been measured and summarized to illustrate the performance of our method. Table 7.3 represents the maximum, the minimum, as well as the average durations of all iterations in our experiments. Though GA is time-consuming, the experimental results depicted in Table 7.3 are still effective because the main memory is utilized as a cache to store the original image and some intermediate results. With the statistics in Table 7.3, we can observe that for each training iteration in our watermarking algorithm, the time consumption for Lena and F16 lies between 209.27 to 274.53 seconds. These values would vary according to the different test images, the decomposition levels in WPT, and the compositions of the fitness function. As we pointed out in Sec. 7.2.3, time consumption with this order limits the application of our method for real-time processing. For the archival of multimedia with trusted communication, our method may have the potential for real applications.

## 7.4 Conclusion

Embedding watermarks into images can be referred to as a constrained optimization problem. Hence, genetic algorithm can be used to solving this problem. In this paper, we proposed a genetic watermarking scheme based on the wavelet packet transform. Genetic algorithm is used to select an appropriate basis from permissible bases of wavelet packet transform to increase the robustness of the embedded watermarks. Robustness can be optimized by GA, and fidelity can be reached with the inherent characteristics of the proposed algorithm. Experimental results demonstrate that the proposed method can increase the capability for resisting some image processing

attacks if an appropriate fitness function of GA was adopted. Therefore, proposed method in this paper is suitable for the application of trusted communication. In the future, efficient approaches to finding the best basis for watermark embedding would be another topic for further studies.

## References

1. Moschetta, E., Antunes, R.S., Barcellos, M.P.: Flexible and secure service discovery in ubiquitous computing. *Journal of Network and Computer Applications* 33, 128–140 (2010)
2. Petitcolas, F.A.P., Anderson, R.J., Kuhn, M.G.: Information hiding — A Survey. *Proceedings of the IEEE* 87, 1062–1078 (1999)
3. Huang, H.C., Fang, W.C.: Techniques and applications of intelligent multimedia data hiding. *Telecommunication Systems* 44, 241–251 (2010)
4. Nikolaidis, N., Pitas, I.: Copyright protection of images using robust digital signatures. In: *IEEE Int'l. Conf. on Acoustics, Speech, and Signal Processing*, pp. 2168–2171 (1996)
5. Oztan, B., Sharma, G.: Continuous phase-modulated halftones. *IEEE Trans. Image Processing* 18, 2718–2734 (2009)
6. Pan, J.S., Hsin, Y.C., Huang, H.C., Huang, K.C.: Robust image watermarking based on multiple description vector quantisation. *Electronics Letters* 40, 1409–1410 (2004)
7. Shieh, C.S., Huang, H.C., Wang, F.H., Pan, J.S.: An embedding algorithm for multiple watermarks. *Journal of Information Science and Engineering* 19, 381–395 (2003)
8. Huang, H.C., Pan, J.S., Huang, Y.H., Wang, F.H., Huang, K.C.: Progressive watermarking techniques using genetic algorithms. *Circuits, Systems, and Signal Processing* 26, 671–687 (2007)
9. Chu, S.C., Huang, H.C., Shi, Y., Wu, S.Y., Shieh, C.S.: Genetic watermarking for zerotree-based applications. *Circuits, Systems, and Signal Processing* 27, 171–182 (2008)
10. Huang, H.C., Chu, C.M., Pan, J.S.: The optimized copyright protection system with genetic watermarking. *Soft Computing* 13, 333–343 (2009)
11. Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, Boston (1989)
12. Ó Ruanaidh, J.J.K., Dowling W.J., Boland F.M.: Phase watermarking of digital images. In: *Proc. Int'l Conf. on Image Processing*, 239–242 (1996)
13. Huang, H.C., Fang, W.C.: Metadata-based image watermarking for copyright protection. *Simulation Modelling Practice and Theory* 18, 435–445 (2010)
14. Rahman, S.M.M., Ahmad, M.O., Swamy, M.N.S.: A new statistical detector for dwt-based additive image watermarking using the Gauss-Hermite expansion. *IEEE Trans. Image Processing* 18, 1782–1796 (2009)
15. Coifman, R.R., Wickerhauser, M.V.: Entropy-based algorithms for best basis selection. *IEEE Trans. Information Theory* 38, 713–718 (1992)
16. Lévy Véhel, J., Manoury, A.: Wavelet packet based digital watermarking. In: *Proc. Int'l. Conf. Pattern Recognition*, pp. 413–416 (2000)
17. Li, Z.N., Drew, M.S.: *Fundamentals of Multimedia*. Prentice-Hall, Upper Saddle River (2003)
18. Weickert, T., Benjaminsen, C., Kiencke, U.: Analytic wavelet packets combining the dual-tree approach with wavelet packets for signal analysis and filtering. *IEEE Trans. Signal Processing* 57, 493–502 (2009)

19. Dietl, W., Uhl, A.: Watermark Security via Secret Wavelet Packet Subband Structures. In: Lioy, A., Mazzocchi, D. (eds.) CMS 2003. LNCS, vol. 2828, pp. 214–225. Springer, Heidelberg (2003)
20. Schwindt, S., Amornraksa, T.: Performance comparison of zerotrees based digital watermarking. In: Proc. IEEE Int'l Conf. on Industrial Technology, pp. 78–81 (2002)
21. Holland, J.H.: Adaptation in Natural and Artificial Systems. The MIT Press, Cambridge (1992)
22. Chen, Y.H., Su, J.M., Huang, H.C., Fu, H.C., Pao, H.T.: Adaptive watermarking using relationships between wavelet coefficients. In: Proc. IEEE Int'l Symp. on Circuits and Systems, pp. 4979–4982 (2005)

## Lossless Text Steganography in Compression Coding

Chin-Feng Lee<sup>1</sup> and Hsing-Ling Chen<sup>2</sup>

<sup>1</sup> Department of Information Management,  
Chaoyang University of Technology,  
Taichung 41349,  
Taiwan, R.O.C.  
lcf@cyut.edu.tw

<sup>2</sup> Department of Applied Foreign Languages,  
Chaoyang University of Technology,  
Taichung 41349, Taiwan, R.O.C.  
hsinglingchen@gmail.com

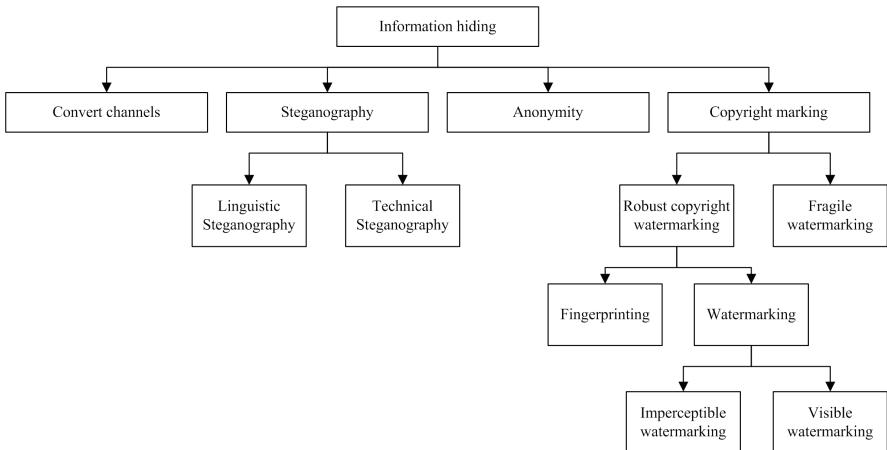
**Summary.** This work presents a novel text steganography scheme in compression domain using a lossless compression coding which called variable Huffman coding. The secret message is embedded into compression codes. The goal of the proposed scheme is covert communication by means of text files, providing high embedding capacity, improving security of the embedded secret message and reducing transmission cost. Additionally, the original text files can be reconstructed without any distortion after the embedded secret message is extracted. In the proposed scheme, each leaf of variable Huffman tree can be used to convey a secret bit at least. According to the practical application, the embedding rate for each leaf is to be increased, i.e, the embedding capacity of the proposed scheme is scalability. The secret keys are employed to generate the stego-compression-code for each leaf of variable Huffman tree to protect the embedded message. The extracting embedded message will be meaningless without the secret keys being known. Furthermore, the size of stego-compression-code is less than the size of original file plus secret message in the proposed scheme so that the transmission cost can be reduced. Additionally, experimental results show that the performance of our method is indeed superior to Chang et al.'s scheme as well as Chen and Chang's scheme in terms of embedding capacity and transmission cost.

### 8.1 Background

The objective of information hiding provides covert communication avoiding unauthorized users attention, attack or tampered. A secret message can be concealed in a cover medium, forming a stego-medium. The secret message can then be covertly delivered successfully because the human eye cannot easily distinguish between the cover medium and stego-medium. Nowadays, digital media, such as videos, images, text files, and sounds, are commonly used as cover media to convey secret messages

in information-hiding methods. Many information-hiding schemes have been developed [1]–[15]. We most use a digital image as a cover medium to conceal a secret message because a digital image has a special property in that a pixel is similar to its neighboring pixels. Thus, secret data can be concealed easily in a digital image by slightly modifying pixel values. Once a secret message is embedded in a cover image, the cover image is called a stego-image. Secret data can then be conveyed for information sharing as stego-image distortion is insignificant. The peak-signal-to-noise ratio (PSNR) is typically utilized to assess the degree of distortion when a digital image is used as a cover medium. Distortion is generally indiscernible to the human eye when the PSNR exceeds 30 dB.

Today, a great number of information-hiding schemes are developed by different research communities. These schemes can be briefly classified into four main types, i.e. covert channels, steganography, anonymity, and copyright marking, as shown in Fig. 8.1 which can be referred to [16]. However, most information-hiding schemes recently focus on technical steganographic schemes [1, 5, 11, 15, 17] or watermarking schemes [2, 3, 18, 19, 20]. These two categories typically work for the media in the spatial domain or frequency domain. The goal of digital watermarking techniques is to protect copyright ownership. Consequently, their robustness, rather than embedding capacity, is the most important issue. Conversely, steganographic schemes are primarily used for sharing secret messages. Hence, the goals of steganography are high embedding capacity and low distortion. Numerous steganographic schemes which belong to technical steganography have been developed, but the linguistic steganography schemes are seldom discussed. Since a slight modification in text content can be easily discovered so that most efforts work on the field of technical steganography rather than linguistic steganography.



**Fig. 8.1.** A classification of information-hiding schemes [16].

Nevertheless, permanent distortion exists in original images regardless of whether watermarking or steganographic schemes are utilized, even when embedded secret data have been extracted. Applications such as those for military, medical, and law enforcement uses cannot accept even minor image distortion. For example, important secret information, such as the coordinates of an enemy's location can be hidden in a military map for covert communication. However, a commander may make an incorrect decision when a military map is reconstructed with distortion after extracting embedded secret information on the receiver side as the correct coordinates of the enemy's location cannot be clearly positioned on a military map. Over the past decade, reversible data-hiding techniques have garnered considerable attention. A reversible data-hiding technique can extract a secret message embedded in a stego-image, and the cover image can be reconstructed without distortion. Two important evaluation criteria for reversible data-hiding techniques are embedding capacity and image quality degradation. Many early schemes [21]–[23] had very limited embedding capacity. This is because the amount of additional information must be recorded to restore an original cover image. However, this additional information reduces total embedding capacity.

Current reversible data-hiding techniques can be classified as the difference expansion (DE)-based scheme proposed by Tian [24] and histogram-based scheme developed by Ni et al. [25], which improve embedding capacity or reduce image distortion. The properties of both schemes are as follows. Tian [24] developed an excellent reversible data-hiding scheme based on the difference expansion (DE) technique. A secret bit is conveyed after the difference between two neighboring pixels is expanded twice. The embedding capacity of the scheme by Tian is scalable according to a predetermined threshold. As the threshold value increases, embedding capacity increases. In other words, image quality increases when the threshold value decreases. The maximum embedding capacity of the scheme by Tian is 0.5 bit per pixel (bpp). Nevertheless, Tian's scheme still encounters the problem of needing a massive amount of additional information (location map) to identify which pixel suffers from overflow / underflow. Additionally, a highly efficient compression algorithm is required to compress this extra information to create more space for conveying more secret bits. The scheme by Ni et al. [25] is another remarkable reversible scheme in the reversible data-hiding field based on the histogram technique. Their scheme differs from that by Tian. A pair of peak and zero points on a histogram are first obtained. The peak point of the histogram is the most frequently occurring pixel value and the zero point of the histogram is the pixel value with zero or minimal occurrences in the image. The secret message is then conveyed in the peak point after shifting pixel values located between the peak point and zero point. In their scheme, pixel values are shifted by 1 at most. Although Ni et al.'s scheme achieves high image quality, embedding capacity is insufficient because embedding capacity is limited by the number of peak points. Most current reversible schemes [26]–[36] are extensions of the schemes by Tian or Ni et al. that improve embedding capacity or reduce image degradation.

To reduce the bandwidth requirement and improve transmission speed, digital media are typically compressed before being transmitted via the Internet. Generally,

the most common data compression methods are lossy compression and lossless compression. Although these two data compression methods can reduce the size of digital media, lossy compression permanently distorts digital media. Conversely, lossless compression does not generate permanent distortion; namely, a receiver can restore the original media without distortion. For a digital image, a vector quantization (VQ) method [37, 38, 39] is usually used for compression to reduce transmission cost. Before VQ encoding, the Linde-Buzo-Gray (LBG) algorithm [40] is applied to train a codebook. A codebook is generally derived from many typical images to obtain representative blocks. During the encoding stage, the original image is first divided into many non-overlapping blocks. Each block is then assigned an index value, which is the minimal Euclidean distance between the input block and each codeword in the codebook, replacing whole pixels in the input block. Finally, the index table, instead of the original image, is transmitted to the receiver. During the decoding stage, the receiver must have the same codebook as the sender to reconstruct the digital image. Notably, VQ compression is an effective compression method for digital images; however, serious distortion is generated in a reconstructed image; this is called the block effect problem. Thus, the side match VQ (SMVQ) method [41]–[44] has been used to solve the block effect problem. Recent data compression methods compress digital media and conceal secret messages. Many data-hiding schemes [45]–[48] in the compression domain have been developed. A secret message is conveyed using compression codes. Take the VQ-based and SMVQ-based schemes as examples; the codebook is typically sorted, divided into many subcodebooks or a few indication bits are added in front of each index value to assist in concealing secret data. Additionally, some VQ-based data-hiding schemes [49]–[53] have reversibility, such that a receiver can extract a secret message, reconstruct a digital image, and restore the index table without distortion.

## 8.2 Motivation

To date, numerous data-hiding schemes, including reversible and irreversible schemes, for digital images have been generated; however, data-hiding schemes for digital text files [54]–[61] are seldom discussed. The principal reason is that text files are not easily modified. A slight modification in text file content can be detected easily. Most current text-hiding methods [55, 56, 58] alter the spacing of lines or words, add many invisible characters, such as tables, blanks, and ends of rows, or rearrange the order of words in a sentence, such as changing the position of adverbs, to conceal a secret message. Nevertheless, file size increases when a huge number of invisible characters are embedded. Thus, some text-hiding methods [57, 61] first convert text files into digital images. The secret message is then embedded by slightly modifying the spacing of letters, words, or lines. However, text file content cannot be further modified once transformed into a digital image. Liu and Tsai [62] developed a text-hiding method by degenerating document content. Text segments are degenerated by inferior writing, and are then revised using change tracking in Microsoft Word documents while the secret message is conveyed. The secret

message can be extracted by a receiver using change tracking information stored in a stego-document. Although the secret message can be concealed in Microsoft Word documents using the scheme by Liu and Tsai, the large amount of editing de-generation rules must be stored. Additionally, embedding capacity of Liu and Tsai's scheme is very low. Chang et al. [63] generated a novel data-hiding scheme for text files using lossless compression rather than modifying text file content directly. A locally adaptive data compression (LADC) method [64] was adopted to compress a text file and further conceal a secret message using a word list based on a self-organizing heuristic strategy. Since LADC is a lossless compression approach, the scheme by Chang et al. can extract an embedded secret message and restore the original text file without distortion. In their scheme, each symbol can be used to convey one secret bit. Furthermore, the primary merit of the scheme by Chang et al. is that the number of stego-compression codes is less than the number of compression codes plus the secret message. That is, transmission cost is reduced. Chen and Chang [65] then proposed another text-hiding method based on the scheme by Shim et al. [66]. In the scheme by Shim et al., a secret message is concealed in compression codes, which are generated by a lossless compression method called the Lempel-Ziv-Welch (LZW) method. Since few symbols can be utilized to convey a secret message and only one symbol at most can be used to convey a secret bit, the embedding capacity of their scheme is very low. Chen and Chang devised a high-capacity data-hiding LZW (HCDH-LZW) approach that also conceals secret data in LZW compression codes by reducing symbol length. The embedding capacity of the scheme by Chen and Chang is superior to that of the scheme by Shim et al. The merit of their scheme is that the number of symbols that can be used to convey a secret message is increased. Furthermore, the embedding capacity of some symbols can exceed one bit.

To meet the requirements for covert communication and reduced transmission cost, data-hiding schemes in the compression domain have attracted much attention. However, a digital image is often used as a cover medium to convey secret data in most data-hiding methods. Thus, this work proposes a novel lossless text steganographic approach by compression coding. Variable Huffman coding, derived from a well-known lossless data-compression scheme (Huffman coding), is utilized to generate lossless compression codes. Secret data can be concealed in these codes. The proposed scheme has the following advantages: high embedding capacity; scalable embedding capacity; the embedded message is protected by secret keys; reversibility; and, reduced transmission cost. Additionally, comparison with the schemes by Chang et al. [63] and Chen and Chang [65] demonstrate that the proposed scheme is superior in embedding capacity and compression rate. Actually, the medium in the proposed method is not restricted to text files; that is, the proposed method can use other digital media such as images, videos, and audio.

The remainder of this paper is organized as follows. Related work is described in Section 8.3. Detailed descriptions of embedding and extracting procedures are given in Section 8.4. Section 8.5 compares the performance of the proposed scheme with that of other text-hiding schemes. Finally, conclusions are given in Section 8.6.



### 8.3 Related Work

The Huffman coding method is first introduced in Section 8.3.1. Section 8.3.2 describes the scheme by Chang et al. [63]. Finally, the scheme by Chen and Chang [65] is characterized in Section 8.3.3.

#### 8.3.1 Huffman Coding

Huffman coding, which was developed by Huffman in 1952 [67], has been widely used for compression of digital media such as text files, images, audio, and video. For instance, Huffman coding is applied to compress the result of a quantization stage in JPEG quantization stage [68]. Digital media, depending on the characteristics of the digital media being compressed, can be saved the file size of 20% – 90%. The unique binary string, called a codeword, is then assigned to each symbol. The length of each codeword in Huffman coding, which depends on the frequency at which a symbol occurs, is variable. As codeword length increases, the number of times a symbol occurs in a digital medium decreases, and vice versa. Obviously, Huffman coding uses the frequency at which symbols occur to generate an optimal method for substituting the original symbol for digital media using codewords. Huffman encoding and Huffman decoding are described as follows.

#### Huffman Encoding

Let source symbols  $C = \{c_1, c_2, c_3, \dots, c_u\}$  with frequencies  $F = \{f_1, f_2, f_3, \dots, f_u\}$ , where  $u$  is the number of symbols in input digital medium. The process for building a Huffman tree, which is a bottom-up process, is as follows.

Input: A digital medium.

Output: A Huffman tree.

- Step 1: Statistically analyze an input digital medium to obtain the corresponding symbol set  $C = \{c_1, c_2, c_3, \dots, c_u\}$  and the associated frequency set  $F = \{f_1, f_2, f_3, \dots, f_u\}$ .
- Step 2: Sort elements  $c_i$  in  $C$  in increasing order according to their occurrence frequencies  $f_i$  in  $F$ .
- Step 3: Remove the two least frequent elements from  $C$  and make each be a leaf. Next, create a new node as the parent of the two leaves with a frequency computed by summing the frequencies of its two child nodes. Two binary bits, '0' and '1', are then assigned to the left edge and right edge of the new node, respectively.
- Step 4: Insert the new created node and its frequency into set  $C$  and set  $F$ , respectively.
- Step 5: Repeat Steps 2-4 until the root node is obtained. Finally, the Huffman tree is complete.

The following example describes how to build a Huffman tree. Figure 8.2 shows these processes.

### Example 1

Consider source symbols  $C=\{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f}\}$  with frequencies  $F=\{10, 32, 18, 59, 45, 80\}$ , respectively. The symbols in  $C$  have been sorted in increasing order according to their frequencies in  $F$  (Fig. 8.2(a)). Since symbols  $\mathbf{a}$  and  $\mathbf{b}$  are the two elements in  $C$  with the lowest occurrence frequencies, first merge  $\mathbf{a}$  and  $\mathbf{b}$  into a new node (28) (Fig. 8.2(b)). Next, two binary bits, ‘0’ and ‘1’, are assigned to the left edge and right edge of the new node (28), respectively. Next, add node (28) and its frequency into set  $C$  and set  $F$ , respectively, and resort elements in  $C$  based on their frequencies in  $F$ . Repeat this process (Fig. 8.2(c)–(e)) until the last merged node (244) is obtained. Figure 8.2(f) shows the resulting Huffman tree.

After constructing the Huffman tree, each symbol  $c_i$  is assigned a unique binary string (codeword) by traversing the Huffman tree from the root node to each leaf node. These codewords are then substituted for the original symbols to deliver a receiver. Table 8.1 lists the final codeword for each symbol. Thus, symbol  $\mathbf{a}$  is replaced by “1000”, and symbol  $\mathbf{b}$  is replaced by “101”, and so on. The final size of the compressed code is

$$10 \times 4 + 32 \times 3 + 18 \times 4 + 59 \times 2 + 45 \times 2 + 80 \times 2 = 576 \text{ bits.}$$

Clearly, the original file size is

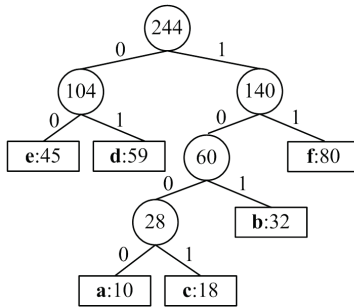
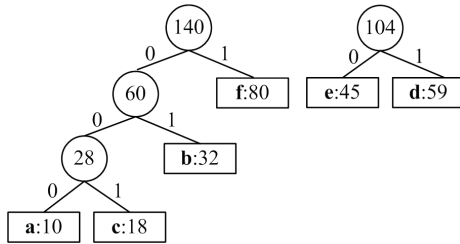
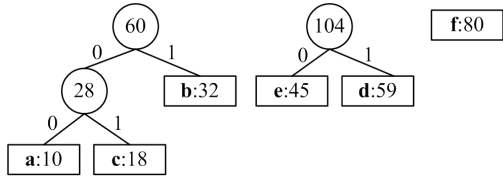
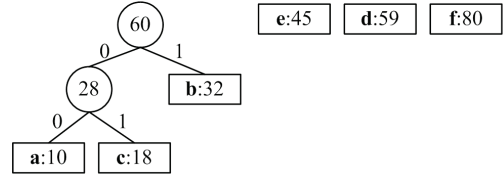
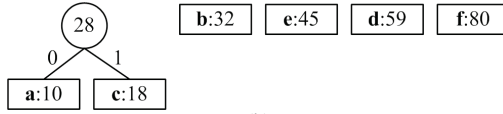
$$(10 + 32 + 18 + 59 + 45 + 80) \times 8 = 1,952 \text{ bits}$$

Accordingly, approximate 71% space can be saved.

On the receiver side, the original digital medium can be reconstructed without distortion according to the received compressed code and Huffman table, both of which are sent by a sender. Since Huffman coding is easy and has good compression efficiency, Huffman coding remains a popular lossless compression method.

### 8.3.2 The Scheme by Chang et al.

Chang et al. [63] developed a novel data-hiding scheme for text files using lossless compression method (LDAC). A secret message is concealed in compression codes by using a word list based on a self-organizing heuristic strategy. In their scheme, each symbol in a cover file can be used to convey one secret bit. Moreover, the embedded secret message can be extracted and the original text file can be reconstructed without distortion. Before introducing the embedding procedure, some notation must first be defined. Whole characters in the cover file are denoted as  $C = \{c_1, c_2, \dots, c_u\}$ , where  $u$  is the number of characters. The secret message,  $S = \{s_1, s_2, \dots, s_u\}$ , is a bitstream. Notations  $TL_0$  and  $TL_1$  are two character lists for character encoding; and  $PE(c_i)$  represents position encoding corresponding to character  $c_i$ . A switcher, \$, is inserted into  $TL_0$  or  $TL_1$  when a secret bit, 1, follows a sequence of secret bits 0s, or secret bit 0 follows a sequence of secret bits 1s.



**Fig. 8.2.** Building a Huffman tree. (a) Source symbols and their frequencies; (b)-(e) intermediate stages; (f) the final Huffman tree.

**Table 8.1.** The Huffman table.

| Symbol   | Frequency | Codeword |
|----------|-----------|----------|
| <b>a</b> | 10        | 1000     |
| <b>b</b> | 32        | 101      |
| <b>c</b> | 18        | 1001     |
| <b>d</b> | 59        | 01       |
| <b>e</b> | 45        | 00       |
| <b>f</b> | 80        | 11       |

Indicator  $I \in \{0, 1\}$  is used to identify whether a character  $c_i$  appears for the first time. The embedding procedure is described as follows.

Input: A cover file and secret message  $S$ .

Output: Stego-compression codes.

Procedure Encoding ( $c, TL_i$ )

Begin

    If  $c$  exists in  $TL_j$ , then

        Set  $I = 1$ , and then return  $I||PE(c)$

    Otherwise

        Add  $c$  into  $TL_i$ , set  $I = 0$ , and then return  $I||c$

End

Procedure Embedding ( )

Step 1: Empty the character list,  $TL_j$ , where  $j \in \{0, 1\}$  and initially  $j = 0$ . Namely,  $TL_0 = \emptyset$  and  $TL_1 = \emptyset$ .

Step 2: Obtain the first character,  $c_1$ , from a cover file and the first secret bit,  $s_1$ , from the secret message. Next, call Encoding ( $c_1, TL_0$ ). The remaining characters,  $c_i$ , and secret bits,  $s_i$ , are processed in Step 3, where  $i = 2, 3, \dots, u$ .

Step 3:

    If  $s_i = s_{i-1}$ , then

        Call Encoding ( $c_i, TL_j$ )

    Otherwise, if  $s_i \neq s_{i-1}$ , then

        Call Encoding ( $\$, TL_j$ )

        Switch to another character list  $TL_j$  by  $j = (j + 1) \bmod 2$

        Call Encoding ( $c_i, TL_j$ )

Step 4: Repeat Step 3 until all secret bits are embedded. Finally, generate the stego-compression code.

To extract the embedded secret message, a receiver must obtain the stego-compression code and know the first secret bit,  $s_1$ ; otherwise, the embedded secret message cannot be extracted and the cover file cannot be reconstructed.

### 8.3.3 The Scheme by Shim et al.

Shim et al. [66] developed a data-hiding approach using a lossless compression method (LZW). The secret message is concealed in compression codes by reducing symbol lengths. Before the embedding procedure, a threshold,  $TH$ , must have an assigned value. When the length of a symbol exceeds  $TH$ , the symbol is used to convey one secret bit in the output LZW code; otherwise, only the LZW code of the symbol is output. In their approach, a secret bit is conveyed by modifying the parity of symbol length to match that of the secret bit by shrinking the last character of a symbol. Since only few symbols can meet this condition for conveying a secret message and only one symbol at most can be used to convey a secret bit, embedding capacity is too low. To overcome this flaw, Chen and Chang [65] developed a method that is an extension of the scheme developed by Shim et al. In the scheme by Shim et al., a secret message is also conveyed by reducing symbol lengths. However, the scheme by Chen and Chang increases the capacity for conveying secret bits for each symbol and increases the number of symbols; thus, high embedding capacity is achieved. Notations and the embedding procedure are described as follows. Whole characters in the cover file are denoted as  $C = \{c_1, c_2, \dots, c_u\}$ , where  $u$  is the number of characters. The secret message,  $S$ , is a bitstream, where  $s_j \in \{0, 1\}$ .

Input: A cover file and secret message.

Output: Stego-compression codes.

Step 1: Read first two characters,  $c_1$  and  $c_2$ , from the cover file. Next, assign an initial symbol  $\alpha = c_1 || c_2$ , where symbol ' $||$ ' indicates that  $c_1$  concatenates  $c_2$ . Assign  $n = 2$ , where  $n$  is the length of the new created symbol  $\alpha$ .

Step 2:

If symbol  $\alpha$  occurs in the dictionary, then

assign the previous symbol  $\alpha_p$ , such that  $\alpha_p = \alpha$ . Next, read next character  $c$  and reassign  $\alpha = \alpha || c$ . Next, increase  $n$  by one and go to Step 2.

Otherwise,

estimate embedding capacity,  $m$ , which is computed as  $m = \log_2(n - 1)$ .

Step 3: Read the next  $m$  binary secret bits and transform the  $m$ -bit string into its corresponding decimal value, denoted as  $m_d$ . Next, shrink the last  $m_d$  characters of symbol  $\alpha$ , i.e.,  $c_1 c_2 \dots c_{m_d}$ . Set a new symbol  $\alpha_n$  such that  $\alpha = \alpha_n || c_1 c_2 \dots c_{m_d}$ . Then, set the previous symbol  $\alpha_p$  such that  $\alpha_n = \alpha_p || c_s$ , where  $c_s$  is the last character in  $\alpha_n$ . Return the shrunk character  $c_1 c_2 \dots c_{m_d}$  back to the cover file. The  $n$  is decreased by  $m$ .

Step 4: Output the dictionary index of previous symbol  $\alpha_p$  as the LZW code and then add the new symbol,  $\alpha_n$ , to the dictionary. Next, assign  $\alpha = c_s$  and  $n = 1$ , then go to Step 2 until all  $c_i$  and  $s_j$  are encoded and embedded, respectively.

In the technique by Chen and Chang, embedding capacity of a symbol is logarithm of the length of the previous symbol. For instance, embedding capacity is 1 when the

length of the previous symbol is 2-3. If the length of the previous symbol is 4-7, then embedding capacity is 2. Clearly, the embedding capacity of each symbol in their scheme varies. Furthermore, a large number of new symbols is created, such that an increased number of symbols can be used to convey secret bits. This is why the scheme by Chen and Chang achieves a higher embedding capacity than the scheme by Shim et al.

Although the schemes by Chang et al. [63] and Chen and Chang [65] can reduce transmission cost and convey the secret message, the embedding capacity of both schemes is limited. To embedding capacity, this work proposed a novel lossless text steganographic scheme using a variable Huffman coding. The proposed scheme has several merits such totally reversibility for text content, scalability in embedding capacity, secret protection by secret keys, and high compress rate. The more details of the proposed scheme are described in Section 8.4.

## 8.4 Proposed Scheme

Unlike common steganographic methods that use digital images as cover media, the goal of the proposed scheme is covert communication using text files, providing high embedding capacity, improving security of embedded secret messages, and reducing transmission cost. A variable Huffman coding derived from the well-known Huffman coding scheme is utilized to generate lossless compression codes. Secret data are concealed in these codes in the proposed scheme. A detailed description of the embedding procedure is given in Section 8.4.1. Section 8.4.2 illustrates the extraction and recovery procedure.

### 8.4.1 Embedding Procedure

For convenience, some notations are first defined. Symbols in the cover file are denoted as  $C = \{c_1, c_2, c_3, \dots, c_u\}$  with frequencies  $F = \{f_1, f_2, f_3, \dots, f_u\}$ , where  $u$  is the number of symbols. The secret message,  $S$ , is a bitstream, where  $s_j \in \{0, 1\}$ . Notation  $n$  indicates that each leaf node in a variable Huffman tree can be used to convey the maximum number of secret bits. Notably,  $n$  is also a secret key in the proposed scheme. The secret key,  $K$ , is regarded as the seed of a pseudo-random-number function used to generate a binary sequence  $R$ . In the proposed scheme, a variable Huffman tree is first constructed before embedding secret data. The process of building a variable Huffman tree is summarized as follows; Figure 8.3 shows the process flowchart.

Input: A cover file,  $n$ , and  $K$ .

Output: A variable Huffman Tree  $T$ .

Step 1: Statistically analyze the input cover file to obtain the corresponding symbol set  $C = \{c_1, c_2, c_3, \dots, c_u\}$  and the associated frequency set  $F = \{f_1, f_2, f_3, \dots, f_u\}$ .

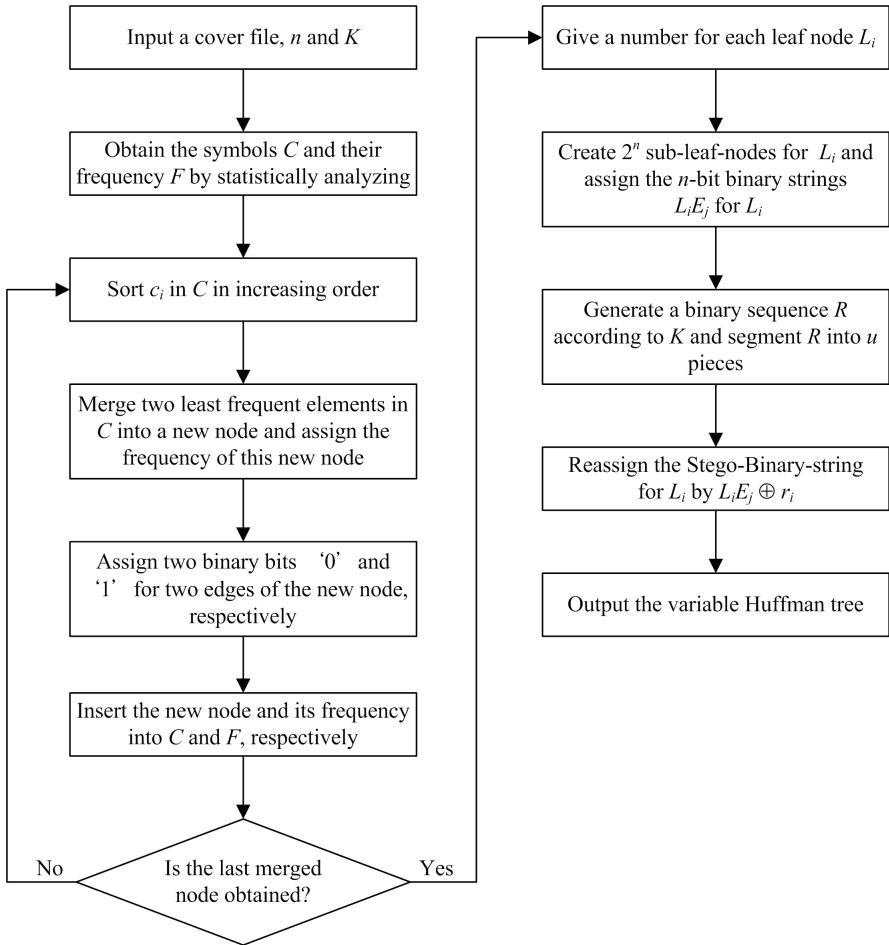
Step 2: Sort the elements  $c_i$  in  $C$  in increasing order based on their frequencies  $f_i$  in  $F$ .

- Step 3 : Remove the two least frequent elements from  $C$  and make each be a leaf. Next, create a new node as the parent of these two leaves and assign the frequency value, which is computed by summing the frequencies of two least frequent elements, to this parent node. Next, two binary bits, '0' and '1', are assigned to the left edge and right edge of the new node, respectively.
- Step 4: Insert the created new node and its frequency into set  $C$  and set  $F$ , respectively.
- Step 5: Repeat Steps 2-4 until the root is obtained. The Huffman tree is then complete.
- Step 6: Give a number  $nL_i$  for each leaf node  $L_i$  in the Huffman tree from left to right sequentially, i.e.,  $nL_0, nL_1, \dots, nL_{u-1}$ , where  $u$  is the number of leaf nodes in the Huffman tree.
- Step 7: Create  $2^n$  children for each leaf node  $L_i$  on the Huffman tree. Next, the  $2^n$  edges of  $L_i$  are sequentially assigned an  $n$ -bit unique binary string. Each unique binary string of  $L_i$  is denoted as  $L_iE_j$ , where  $j = 0, 1, \dots, 2^n - 1$ . The values of these unique binary strings fall within the range  $[0, n - 1]$  in a decimal notational system.
- Step 8: Generate a binary sequence  $R$  by the pseudo-random-number function according to the secret key,  $K$ , and then segment  $R$  into  $u$  pieces, each of which has  $n$  bits. Namely,  $R = \{r_0, r_1, r_2, \dots, r_{u-1}\}$  and  $r_i$  are comprised of an  $n$ -bit string.
- Step 9: Reassign the stego-binary-string  $L'_iE'_j$  for  $L_i$  by  $L'_iE'_j = \{L_0E_0 \oplus r_0, L_0E_1 \oplus r_0, L_0E_2 \oplus r_0, \dots, L_0E_{2^n-1} \oplus r_0, L_1E_0 \oplus r_1, L_1E_1 \oplus r_1, L_1E_2 \oplus r_1, \dots, L_1E_{2^n-1} \oplus r_1, \dots, L_{u-1}E_0 \oplus r_{u-1}, L_{u-1}E_1 \oplus r_{u-1}, L_{u-1}E_2 \oplus r_{u-1}, \dots, L_{u-1}E_{2^n-1} \oplus r_{u-1}\}$ . Finally, the variable Huffman tree,  $T$ , is obtained.

The following example describes the process of building a variable Huffman tree.

### Example 2

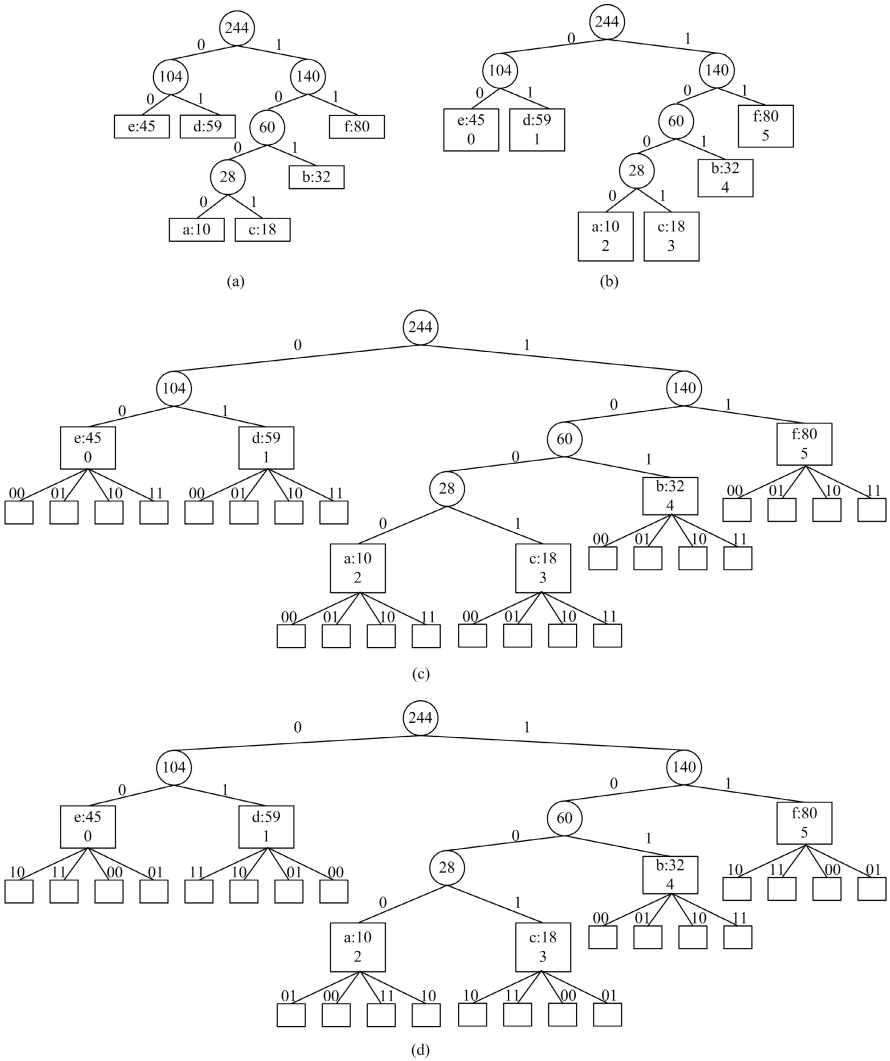
Let  $n = 2$  and  $K = 10$ . We assume the source symbols of a cover file are as same as those in Example 1— $C = \{ \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f} \}$  with frequencies  $F = \{ 10, 32, 18, 59, 45, 80 \}$ , respectively. Thus, the Huffman tree is generated as shown in Fig. 8.4(a). Next, each leaf node  $L_i$  of the Huffman tree is sequentially numbered from left to right in the Huffman tree, such that  $nL_0 = 0, nL_1 = 1, \dots, nL_5 = 5$  (Fig. 8.4(b)). Figure 8.4(c) shows that four children for each leaf node are created because  $n$  has an assigned value of 2. Additionally, each edge  $L_iE_j$  of  $L_i$  is the numbered 2-bit unique string, i.e., "00", "01", "10", and "11". According to the input secret key  $K$ , the binary sequence  $R = "101101100010"$  is generated by the pseudo-random-number function and  $R$  is then divided into six pieces, each of which is a 2-bit string, i.e.,  $R = \{ "10", "11", "01", "10", "00", "10" \}$  such that  $r_0 = "10", r_1 = "10", \dots$ , and  $r_5 = "10"$ . Generate the stego-binary-string  $L'_iE'_j$  for  $L_i$  by  $L_iE_j \oplus r_i$ . Namely,  $L'_iE'_j = \{ "00" \oplus "10", "01" \oplus "10", "10" \oplus "10", "11" \oplus "10", "00" \oplus "11", "01" \oplus "11", "10" \oplus "11", "11" \oplus "11", \dots, "00" \oplus "10", "01" \oplus "10", "10" \oplus "10", "11" \oplus "10" \}$ . Finally, the variable Huffman tree is completed as displayed in Fig. 8.4(d).



**Fig. 8.3.** Flowchart for constructing a variable Huffman tree.

Table 8.2 lists the stego-codeword of each symbol  $c_i$  after traversing the obtained variable Huffman tree. The 2-bit secret message is to be conveyed (fifth column in Table 8.2). Finally, the codeword of each symbol along with the embedded secret message form a stego-codeword (sixth column in Table 8.2) substituted for the original symbol and sent to a receiver. Clearly, cover file size can be reduced to decrease transmission cost and the secret message can be conveyed to achieve covert communication. Further, the cover file cannot be recovered and the embedded secret message is extremely difficult to extract when  $n$ ,  $K$ , and the pseudo-random-number function are not provided.





**Fig. 8.4.** Building a variable Huffman tree.

Example 3, which is extended from Example 2, elucidates the embedding procedure in the proposed scheme.

**Example 3**

We assume the secret message,  $S = "10111010"$ , and the first four symbols in the cover file are **"ebaf"**. First, the secret message is divided into four pieces—"00", "11", "10", and "10". Next, the first symbol, **e**, is substituted as "0000" because the codeword of **e** is "00" and the secret bits are "10" (Table 8.2). The stego-codeword "10111" is then replaced by the second symbol, **b**, as the embedded secret bit is

**Table 8.2.** The variable Huffman table.

| No. | Symbol   | $r_i$ | Codeword | Embedded secret bit | Stego-codeword |
|-----|----------|-------|----------|---------------------|----------------|
| 0   | <b>e</b> | 10    | 00       | 00                  | 0010           |
|     |          |       |          | 01                  | 0011           |
|     |          |       |          | 10                  | 0000           |
|     |          |       |          | 11                  | 0001           |
| 1   | <b>d</b> | 11    | 01       | 00                  | 0111           |
|     |          |       |          | 01                  | 0110           |
|     |          |       |          | 10                  | 0101           |
|     |          |       |          | 11                  | 0100           |
| 2   | <b>a</b> | 01    | 1000     | 00                  | 100001         |
|     |          |       |          | 01                  | 100000         |
|     |          |       |          | 10                  | 100011         |
|     |          |       |          | 11                  | 100010         |
| 3   | <b>c</b> | 00    | 1001     | 00                  | 100110         |
|     |          |       |          | 01                  | 100111         |
|     |          |       |          | 10                  | 100100         |
|     |          |       |          | 11                  | 100101         |
| 4   | <b>b</b> | 10    | 101      | 00                  | 10100          |
|     |          |       |          | 01                  | 10101          |
|     |          |       |          | 10                  | 10110          |
|     |          |       |          | 11                  | 10111          |
| 5   | <b>f</b> | 10    | 11       | 00                  | 1110           |
|     |          |       |          | 01                  | 1111           |
|     |          |       |          | 10                  | 1100           |
|     |          |       |          | 11                  | 1101           |

“11”. In the same process, symbols **a** and **f** are replaced by “100011” and “1100”, respectively. Finally, the stego-compression code “0000101111000111100” is replaced the original letters “**ebaf**” and transmitted to a receiver. Hence, file size is reduced by approximately 41%, computed from  $\left(\frac{(4 \times 8 - 19)}{(4 \times 8)}\right)$ .

#### 8.4.2 Extraction and Recovery Procedure

Once the receiver receives symbols and the frequencies of the cover file, the Huffman tree can be reconstructed. However, the receiver must also know the  $n$  and  $K$  values the sender used; otherwise, embedded secret data cannot be extracted and the cover file cannot be restored from the stego-compression code. The extraction and recovery procedure in the proposed scheme are described as follows; Figure 8.5 shows the flowchart of the extraction and recovery procedure.

**Input:** Symbols and their frequencies in the cover file, the stego-compression code,  $n$ , and  $K$ .

**Output:** The embedded secret message,  $S$ , and the cover file.

**Step 1:** Construct the Huffman tree according to input symbols and their frequencies.

- Step 2: Assign a number  $nL_i$  for each leaf node  $L_i$  in the Huffman tree from left to right sequentially.
- Step 3: Generate a binary sequence  $R$  according to the secret key,  $K$ , using the pseudo-random-number function and then segment  $R$  into  $u$  pieces, each of which has  $n$  bits, i.e.,  $R = \{r_0, r_1, r_2, \dots, r_{u-1}\}$ .
- Step 4: Traverse the Huffman tree when a character  $cc_k = 0$  is read from stego-compression code  $CC$ , then go left-node whereas go right-node.
- Step 5: Read the next character, i.e.,  $cc_k = cc_{k+1}$  and  $k = k + 1$ , and repeat Step 4 until a leaf node  $L_i$  is reached. Thus, the symbol can be restored. Next, the stego-secret-bits  $s'_j$  can be obtained by  $s'_j = cc_{k+1}||cc_{k+2}||\dots||cc_{k+n}$ , where ‘||’ is denoted as two characters that are concatenated. The true secret bits  $s_j$  can then be obtained by  $s_j = s'_j \oplus r_{nL_i}$ . Reassign  $cc_k$  and  $k$  as  $cc_k = cc_{k+n+1}$ ,  $k = k + n + 1$ , respectively.
- Step 6: Repeat Steps 4-5 until the stego-compression code is empty. Finally, the embedded secret message can be extracted and the cover file can be recovered without distortion.

A relatively more detailed example describing the extraction and recovery procedure is given as follows.

#### Example 4

A receiver obtains a stego-compression code as “0000101111000111100”, and source symbols  $C = \{ \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f} \}$  with frequencies  $F = \{10, 32, 18, 59, 45, 80\}$ . We assume the receiver knows the secret keys  $n = 2$  and  $K = 10$  and has the same pseudo-random-number function as the sender. The processes for extracting an embedded secret message and restoring original symbols from the stego-compression code are as follows. The corresponding Huffman tree is generated based on the content of  $C$  and  $F$  (Fig. 8.4(a)). Next, each leaf node  $L_i$  in the Huffman tree is numbered (Fig. 8.4(b)). According to the value of  $K$ , binary sequence  $R = “101101100010”$  is generated by the pseudo-random-number function. Since six symbols exist ( $u = 6$ ) and  $n = 2$  in this example, sequence  $R$  is further divided into six pieces, each of which is a 2-bit string, i.e.,  $R = \{“10”, “11”, “01”, “10”, “00”, “10”\}$ . One then traverses the Huffman tree from the root to leaves according to stego-compression code  $CC$ . Thus, the first original symbol,  $\mathbf{e}$ , can be restored from  $CC$  because the first 2-bit string of  $CC$  (“00”) equals the codeword of symbol  $\mathbf{e}$ . Next, the stego-secret-bits  $s'_1 = “00”$  is obtained by the third bit ‘0’ concatenate and the fourth bit 0 because  $n = 2$ . Finally, the true secret bits  $s_1$  can be obtained using  $s_1 = “00” \oplus r_0 = “00” \oplus “10” = “10”$ . Using the same process, the other original symbols can be restored as “**baf**” and the embedded secret bits can be extracted as “111010”. Finally, all original symbols, “**ebaf**”, and the secret message, “10111010”, can be reconstructed from the stego-compression code.

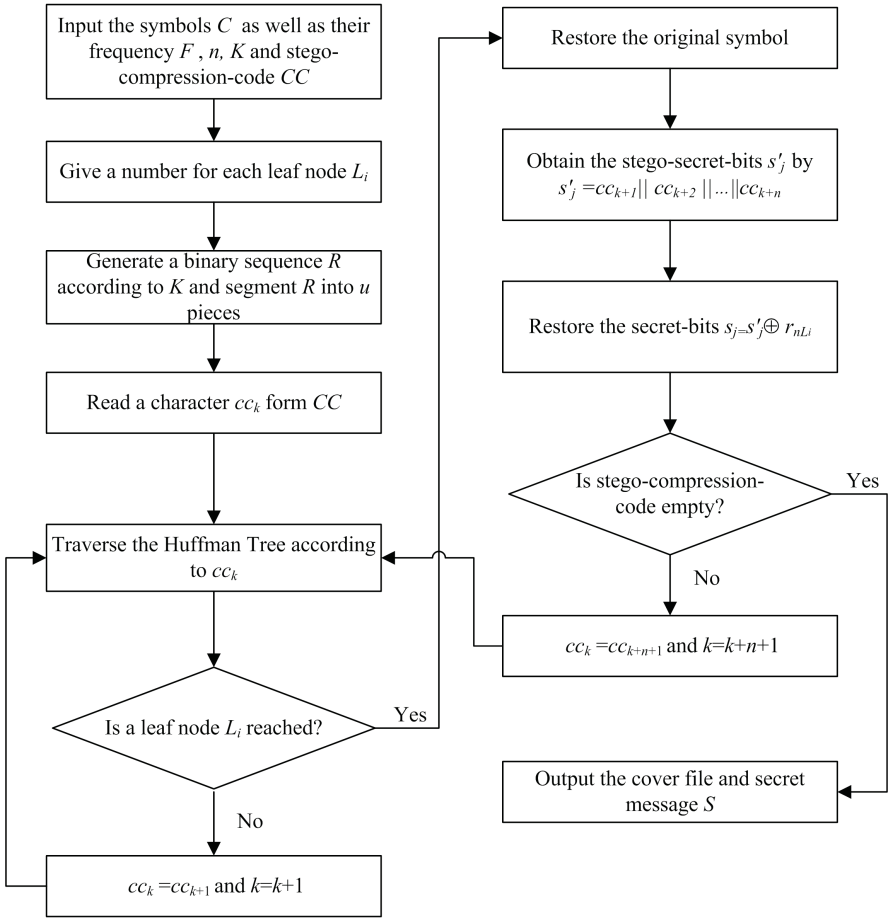


Fig. 8.5. Flowchart of the extraction and recovery procedure.

### 8.5 Experimental Results and Discussion

Experimental results characterize the performance of the proposed scheme in terms of embedding capacity and transmission cost. This work utilized Delphi language running on an AMD Athlon 64 Dual Core 5000+ computer with 2GB of memory. Five news items from the CNN News site on the Internet, and five program segments, each of which was written in C language, are used as cover files in the experiment. Figures 8.5(a) and 8.5(b) present one news and one program segment, respectively. Secret message,  $S$ , was generated randomly using the random function in Delphi language. The compressed rate, calculated as follows, was used to evaluate the transmission cost. As the compression rate increases, transmission cost decreases.

$$\text{Compressed rate} = \frac{(\text{cover file size} + \text{secret message} - \text{stego-compression code size})}{(\text{cover file size} + \text{secret message})}$$

President Bush used a meeting with Mexican and Canadian leaders Monday to hammer Democrats who oppose a free trade deal between the U.S. and Colombia, saying that blocking the deal is bad for American workers and bad for our security. Bush met with Mexican President Felipe Calderon and Canadian Prime Minister Stephen Harper at the fourth North American Leaders Summit, a two day meeting in New Orleans, Louisiana, that started Monday.

Mexico and Canada are not directly affected by the Colombia trade agreement, which Bush sent to Congress this month. But Calderon and Harper are expected to argue that free trade throughout the hemisphere is good for all three North American countries.

Democrats in the House of Representatives, led by Speaker Nancy Pelosi, have moved to change rules so they would not be forced to vote on the deal with Colombia.

Unfortunately, we had a setback in a very important free trade agreement with Colombia, Bush told a New Orleans civic group after meeting with Calderon and Harper. The deal is dead unless she changes her mind and that's bad for American workers and it's bad for our security.

Bush said the current trade program with Colombia is unfair because virtually all Colombian products enter the United States duty free, but American exports face steep tariffs upon entering the South American nation. The agreement, if passed, would eliminate those tariffs and other barriers to trade.

The president, along with Calderon and Harper, also used the meeting as an opportunity to defend the North American Free Trade Agreement, which Democratic presidential hopefuls Hillary Clinton and Barack Obama have criticized on the campaign trail. Both senators said they will work to amend the deal if elected president.

Bush said that since NAFTA was established years ago, trade among the three nations has tripled and their economies have grown.

(a)

```
#include <vc1.h>
#include <math.h>
#include <algorithm>
#include "Unit1.h"
#include <iostream.h>
#include <stdio.h>

#pragma package(smart_init)
#pragma resource "*.dfm"
#pragma hdrstop
TForm1 *Form1;
int range = 512*512;
int R[512][512];
int G[512][512];
int B[512][512];

int equ_R[512*512];
int equ_G[512*512];
int equ_B[512*512];

typedef struct RGB{
    int R;
    int G;
    int B;
}RGB;

RGB output[512*512];
```

(b)

Fig. 8.6. Content in the two cover files-(a) a news item, and (b) a program segment.

To demonstrate the performance of proposed method, the proposed scheme is compared with two recently developed compression-based data-hiding schemes [63, 65]. Table 8.3 and Table 8.4 show comparison results. Table 8.3 shows the comparison results when text file content is news; Table 8.4 lists comparison results when text file content is a program segment. The secret key,  $n$ , of the proposed scheme is assigned a value of 1 at this time. Table 8.3 and Table 8.4 show the “minimum” embedding capacity of the proposed scheme while the “maximum” embedding capacities of the schemes by Chang et al. [63] and Chen and Chang [65] are also displayed in Table 8.3 and Table 8.4. Clearly, the embedding capacity of the approach by Chang et al. and the proposed scheme are better than that of the scheme by Chen and Chang. Each symbol in the scheme by Chang et al. and the proposed scheme can be used to convey one secret bit. That is, the embedding capacity equals the number of symbols in the cover file. Nevertheless, only a few symbols can be used to convey secret bits in the method by Chen and Chang. For example, 11,054 secret bits can be conveyed by the scheme by Chang et al. and the proposed scheme (third column (news (1)) in Table 8.3). Conversely, only 4,726 secret bits can be conveyed using the method by Chen and Chang. The embedding capacity of the approach by Chang et al. and the proposed scheme is approximately three times that of the scheme by Chen and Chang. For the compressed rate of the three schemes, the proposed scheme outperforms the other two. For example, the cover file size along with secret message size can be reduced by about 39% using the proposed scheme (third column (program (1)) in Table 8.4). In contrast, the schemes by Chang et al. and Chen and Chang only reduce the file size by roughly 22% and 13%, respectively. Since the proposed scheme achieves a high compressed rate, transmission cost can be markedly reduced. Experimental results show that performance in terms of embedding capacity and transmission cost of the proposed scheme is superior to that of the other two schemes.

Another significant benefit of the proposed scheme is that embedding capacity is scalable. In practical applications, the embedding rate of each symbol can be increased. The embedding capacity of the proposed method is dominated by the parameter,  $n$ , which is also regarded as a secret key. The embedding rate of each symbol of a cover file equals the  $n$  value. The sixth column in Table 8.5 shows the number of embedded secret message that can be conveyed by ten test files. Take news (1) for example, the embedding capacity can be increased from 11,054 to 44,216 bits when the assigned value of  $n$  is in the range of one to four. Although the size of the sego-compressed code is to be increased when a large  $n$  is utilized, the compressed rate remains acceptable. For example, news (1) is used to conceal 44,216 secret bits when  $n = 4$  (eighth column in Table 8.5). Although the obtained the sego-compressed code is increased to 11,511 bytes, the compressed rate is maintained at about 30%. The embedding capacity of the proposed scheme is four times that of the scheme by Chang et al. and the embedding rate of the proposed method is 2.2 times that of the scheme by Chang et al. (Table 8.3). This work also compares the performance of the proposed scheme with that of the approach by Chen and Chang. Although the compressed rate of the proposed method (30.57 %) is less than that of the scheme by Chen and Chang (38.56%), embedding capacity of the

**Table 8.3.** Comparison results for the proposed scheme and the other schemes with news

| Methods                      | Size                         | Files  |       |        |        |       |
|------------------------------|------------------------------|--------|-------|--------|--------|-------|
|                              |                              | N (1)  | N (2) | N (3)  | N (4)  | N (5) |
|                              |                              | 11,054 | 8,876 | 12,864 | 18,167 | 9,017 |
| Chang et al.'s scheme [63]   | Compressed Code (Bytes)      | 10,815 | 8,674 | 12,717 | 18,064 | 8,225 |
|                              | Secret Message (Bits)        | 11,054 | 8,876 | 12,864 | 18,167 | 9,017 |
|                              | Sego-compressed Code (Bytes) | 10,715 | 8,945 | 13,174 | 18,535 | 9,099 |
|                              | Compressed rate(%)           | 13.84  | 10.42 | 8.97   | 9.31   | 10.30 |
| Chen and Chang's scheme [65] | Compressed Code (Bytes)      | 5,308  | 5,080 | 8,812  | 10,780 | 5,896 |
|                              | Secret Message (Bits)        | 4,726  | 3,641 | 4,489  | 7,147  | 3,464 |
|                              | Sego-compressed Code (Bytes) | 7,120  | 6,554 | 10,541 | 13,659 | 7,198 |
|                              | Compressed rate (%)          | 38.56  | 29.76 | 21.48  | 28.34  | 23.83 |
| Proposed scheme              | Compressed Code (Bytes)      | 5,984  | 5493  | 8864   | 10917  | 5367  |
|                              | Secret Message (Bits)        | 11,054 | 8,876 | 12,864 | 18,167 | 9,017 |
|                              | Sego-compressed Code (Bytes) | 7,366  | 6,602 | 10,472 | 13,188 | 6,494 |
|                              | Compressed rate (%)          | 40.77  | 33.88 | 27.64  | 35.48  | 35.99 |

\* N denotes news.

**Table 8.4.** Comparison results for the proposed scheme and the other schemes with program segments

| Methods                      | Size                         | Files  |        |       |        |        |
|------------------------------|------------------------------|--------|--------|-------|--------|--------|
|                              |                              | P (1)  | P (2)  | P (3) | P (4)  | P (5)  |
|                              |                              | 12,685 | 15,267 | 9,402 | 12,028 | 16,288 |
| Chang et al.'s scheme [63]   | Compressed Code (Bytes)      | 11,075 | 13,332 | 8,214 | 10,496 | 14,252 |
|                              | Secret Message (Bits)        | 12,685 | 15,267 | 9,402 | 12,028 | 16,288 |
|                              | Sego-compressed Code (Bytes) | 11,149 | 13,247 | 8,295 | 10,560 | 14,724 |
|                              | Compressed rate(%)           | 21.87  | 22.87  | 21.58 | 21.96  | 19.65  |
| Chen and Chang's scheme [65] | Compressed Code (Bytes)      | 9,718  | 11,698 | 7,359 | 9,380  | 12,820 |
|                              | Secret Message (Bits)        | 4,333  | 5,209  | 3,064 | 4,010  | 5,281  |
|                              | Sego-compressed Code (Bytes) | 11,467 | 13,802 | 8,706 | 10,992 | 14,961 |
|                              | Compressed rate (%)          | 13.30  | 13.29  | 11.03 | 12.27  | 11.72  |
| Proposed scheme              | Compressed Code (Bytes)      | 7,162  | 8,591  | 5,307 | 6,841  | 9,260  |
|                              | Secret Message (Bits)        | 12,685 | 15,267 | 9,402 | 12,028 | 16,288 |
|                              | Sego-compressed Code (Bytes) | 8,748  | 10,500 | 6,482 | 8,345  | 11,296 |
|                              | Compressed rate (%)          | 38.70  | 38.87  | 38.72 | 38.33  | 38.36  |

\* P denotes program segments.

proposed method is 6.2 times greater than that of their method when news (1) is used as the cover file. Clearly, the proposed scheme has a high compressed rate when a small  $n$  is adopted, and a high embedding capacity can be achieved when a large  $n$  is employed. Thus, embedding capacity is scalable in practical applications of the proposed method.

The proposed scheme has another advantage that the embedded secret data can be protected using the two secret keys,  $n$  and  $K$ , and the pseudo-random-number

**Table 8.5.** Embedding capacity of the proposed scheme with various values for  $n$ 

| Type             | Size              |                         |        |                       |                              |                     |       |
|------------------|-------------------|-------------------------|--------|-----------------------|------------------------------|---------------------|-------|
|                  | Text file (Bytes) | Compressed Code (Bytes) | $n$    | Secret Message (Bits) | Sego-compressed Code (Bytes) | Compressed rate (%) |       |
| News             | (1)               | 11,054                  | 5,984  | 1                     | 11,054                       | 7,366               | 40.77 |
|                  |                   |                         |        | 2                     | 22,108                       | 8,748               | 36.69 |
|                  |                   |                         |        | 3                     | 33,162                       | 10,130              | 33.35 |
|                  |                   |                         |        | 4                     | 44,216                       | 11,511              | 30.57 |
|                  | (2)               | 8,876                   | 5,493  | 1                     | 8,876                        | 6,602               | 33.88 |
|                  |                   |                         |        | 2                     | 17,752                       | 7,712               | 30.49 |
|                  |                   |                         |        | 3                     | 26,628                       | 18,821              | 27.72 |
|                  |                   |                         |        | 4                     | 35,504                       | 9,931               | 25.41 |
|                  | (3)               | 12,864                  | 8,864  | 1                     | 12,864                       | 10,472              | 27.64 |
|                  |                   |                         |        | 2                     | 25,728                       | 12,080              | 24.88 |
|                  |                   |                         |        | 3                     | 38,592                       | 13,688              | 22.62 |
|                  |                   |                         |        | 4                     | 51,456                       | 15,296              | 20.73 |
|                  | (4)               | 18,167                  | 10,917 | 1                     | 18,167                       | 13,188              | 35.48 |
|                  |                   |                         |        | 2                     | 36,334                       | 15,458              | 31.93 |
|                  |                   |                         |        | 3                     | 54,501                       | 17,729              | 29.03 |
|                  |                   |                         |        | 4                     | 72,668                       | 20,000              | 26.61 |
|                  | (5)               | 9,017                   | 5,367  | 1                     | 9,017                        | 6,494               | 35.99 |
|                  |                   |                         |        | 2                     | 18,034                       | 7,621               | 32.39 |
|                  |                   |                         |        | 3                     | 27,051                       | 8,748               | 29.44 |
|                  |                   |                         |        | 4                     | 36,068                       | 9,875               | 26.99 |
| Program Segments | (1)               | 12,685                  | 7,162  | 1                     | 12,685                       | 8,748               | 38.70 |
|                  |                   |                         |        | 2                     | 25,370                       | 10,333              | 34.83 |
|                  |                   |                         |        | 3                     | 38,055                       | 11,919              | 31.66 |
|                  |                   |                         |        | 4                     | 50,740                       | 13,505              | 29.03 |
|                  | (2)               | 15,267                  | 8,591  | 1                     | 15,267                       | 10,500              | 38.87 |
|                  |                   |                         |        | 2                     | 30,534                       | 12,408              | 34.98 |
|                  |                   |                         |        | 3                     | 45,801                       | 14,316              | 31.80 |
|                  |                   |                         |        | 4                     | 61,068                       | 16,225              | 29.15 |
|                  | (3)               | 9,402                   | 5,307  | 1                     | 9,402                        | 6,482               | 38.72 |
|                  |                   |                         |        | 2                     | 18,804                       | 7,658               | 44.84 |
|                  |                   |                         |        | 3                     | 28,206                       | 8,833               | 31.68 |
|                  |                   |                         |        | 4                     | 37,608                       | 10,008              | 29.04 |
|                  | (4)               | 12,028                  | 6,841  | 1                     | 12,028                       | 8,345               | 38.33 |
|                  |                   |                         |        | 2                     | 24,056                       | 9,848               | 34.50 |
|                  |                   |                         |        | 3                     | 36,084                       | 11,352              | 31.36 |
|                  |                   |                         |        | 4                     | 48,112                       | 12,855              | 28.75 |
|                  | (5)               | 16,288                  | 9,260  | 1                     | 16,288                       | 11,296              | 38.36 |
|                  |                   |                         |        | 2                     | 32,576                       | 13,332              | 34.52 |
|                  |                   |                         |        | 3                     | 48,864                       | 15,368              | 31.38 |
|                  |                   |                         |        | 4                     | 65,152                       | 17,404              | 28.77 |



function. Possible permutations of the label of each leaf  $L_i$  associated sub-leaf-nodes are  $2^n!$  according the input random binary sequence piece  $r_i$ , which is provided by the pseudo-random-number function with  $K$ . We assume a variable Huffman tree has  $m$  leaves. Thus, all possible permutations of all leaves in the variable Huffman tree are  $(2^n!)^m$ . For example, we assume  $n = 1$ ,  $K = 10$ , and 64 unrepeatable symbols exist in the test file (news (1)). Thus, all possible permutations of labels of all leaves are  $2^{64}$ . As the value of  $n$  increases, the number of possible permutations of labels of all leaves increases. Unauthorized users will therefore have extreme difficulty in guessing a secret message when they lack the correct secret keys,  $n$  and  $K$ , and the pseudo-random-number function. Furthermore, a cover file cannot be reconstructed from the obtained stego-compression code.

## 8.6 Conclusions

This work proposed a novel linguistic steganographic scheme in the compression texts using variable Huffman coding. Experimental results demonstrate that the proposed method has high embedding capacity, reduces transmission cost, improves the security of an embedded secret message, and reconstructs the original cover medium without distortion. Furthermore, the embedding capacity of the proposed scheme is scalable in practical applications. Additionally, experimental results show that the performance of the proposed method is superior to that of the schemes by Chang et al. and Chen and Chang in terms of embedding capacity and compression rate.

## References

1. Chan, C.K., Cheng, L.M.: Hiding data in images by simple LSB substitution. *Pattern Recognition* 37, 469–474 (2004)
2. Chang, C.C., Lin, P.Y.: Adaptive watermark mechanism for rightful ownership protection. *Journal of Systems and Software* 81, 1118–1129 (2008)
3. Chang, C.C., Lin, P.Y., Yeh, J.S.: Preserving robustness and removability for digital watermarks using subsampling and difference correlation. *Information Sciences* 179, 2283–2293 (2009)
4. Joo, S., Suh, Y., Shin, J., Kikuchi, H., Cho, S.J.: A new robust watermark embedding into wavelet DC components. *ETRI Journal* 24, 401–404 (2002)
5. Jung, K.H., Yoo, K.Y.: Data hiding method using image interpolation. *Computer Standards and Interfaces* 31, 465–470 (2009)
6. Lee, C.F., Chen, H.L.: A novel data hiding scheme based on modulus function. *Journal of Systems and Software* 83, 832–843 (2010)
7. Lee, C.F., Yang, T.C.: A blind associative watermarking technique using self-embedding. In: *Proc. Int'l. Conf. Intelligent Systems Design and Applications*, vol. 3, pp. 409–413 (2008)
8. Lee, C.F., Yang, T.C.: Semi-blind associative watermarking. In: *Proc. 14th Conf. Artificial Intelligence and Applications* (2009)
9. Lee, I.S., Tsai, W.H.: Data hiding in grayscale images by dynamic programming based on a human visual model. *Pattern Recognition* 42, 1604–1611 (2009)

10. Lee, Y., Kim, H., Park, Y.: A new data hiding scheme for binary image authentication with small image distortion. *Information Sciences* 179(22), 3866–3884 (2009)
11. Tsai, Y.Y., Wang, C.M.: A novel data hiding scheme for color images using a BSP tree. *Journal of Systems and Software* 80(3), 429–437 (2007)
12. Wang, C.M., Wu, N.I., Tsai, C.S., Hwang, M.S.: A high quality steganographic method with pixel-value differencing and modulus function. *Journal of Systems and Software* 81, 150–158 (2008)
13. Wang, R.Z., Lin, C.F., Lin, J.C.: Image hiding by optimal LSB substitution and genetic algorithm. *Pattern Recognition* 34(3), 671–683 (2001)
14. Wu, H.C., Wu, N.I., Tsai, C.S., Hwang, M.S.: Image steganographic scheme based on pixel-value differencing and LSB replacement methods. *Vision, Image and Signal Processing* 152(5), 611–615 (2005)
15. Zhang, X., Wang, S.: Efficient steganographic embedding by exploiting modification direction. *IEEE Communications Letters* 10(11), 781–783 (2006)
16. Petitcolas, F.A.P., Anderson, R.J., Kuhn, M.G.: Information hiding — A survey. *Proceedings of the IEEE* 87(7), 1062–1078 (1999)
17. Mielikainen, J.: LSB matching revisited. *IEEE Signal Processing Letters* 13, 285–287 (2006)
18. Bhatnagar, G., Raman, B.: A new robust reference watermarking scheme based on DWT-SVD. *Computer Standards and Interfaces* 31(5), 1002–1013 (2009)
19. Mohammad, A.A., Alhaj, A., Shaltaf, S.: An improved SVD-based watermarking scheme for protecting rightful ownership. *Signal Processing* 88(9), 2158–2180 (2008)
20. Ni, R., Ruan, Q., Cheng, H.D.: Secure semi-blind watermarking based on iteration mapping and image features. *Pattern Recognition* 38(3), 357–368 (2005)
21. Celik, M.U., Sharma, G., Tekalp, A.M., Saber, E.: Lossless generalized-LSB data embedding. *IEEE Trans. Image Processing* 14(2), 253–266 (2005)
22. De Vleeschouwer, C., Delaigle, J.F., Macq, B.: Circular interpretation of bijective transformations in lossless watermarking for media asset management. *IEEE Trans. Multimedia* 5(1), 97–105 (2003)
23. Fridrich, J., Goljan, M., Du, R.: Lossless data embedding—new paradigm in digital watermarking. *EURASIP Journal on Applied Signal Processing* 2002(2), 185–196 (2002)
24. Tian, J.: Reversible data embedding using a difference expansion. *IEEE Trans. Circuits and Systems for Video Technology* 13(8), 890–896 (2003)
25. Ni, Z., Shi, Y.Q., Ansari, N., Su, W.: Reversible data hiding. *IEEE Trans. Circuits and Systems for Video Technology* 16(3), 354–362 (2006)
26. Alattar, A.M.: Reversible watermark using the difference expansion of a generalized integer transform. *IEEE Trans. Image Processing* 13(8), 1147–1156 (2004)
27. Hong, W., Chen, T.S., Shiu, C.W.: Reversible data hiding for high quality images using modification of prediction errors. *Journal of Systems and Software* 82(11), 1833–1842 (2009)
28. Hyoung, J.K., Sachnev, V., Shi, Y.Q., Jeho, N., Choo, H.G.: A novel difference expansion transform for reversible data embedding. *IEEE Trans. Information Forensics and Security* 3(3), 456–465 (2008)
29. Kamstra, L., Heijmans, H.J.A.M.: Reversible data embedding into images using wavelet techniques and sorting. *IEEE Trans. Image Processing* 14(12), 2082–2090 (2005)
30. Kim, H.J., Sachnev, V., Shi, Y.Q., Nam, J., Choo, H.G.: A novel difference expansion transform for reversible data embedding. *IEEE Trans. Information Forensics and Security* 3(3), 456–465 (2008)
31. Kim, K.S., Lee, M.J., Lee, H.Y., Lee, H.K.: Reversible data hiding exploiting spatial correlation between sub-sampled images. *Pattern Recognition* 42(11), 3083–3096 (2009)

32. Lin, C.C., Tai, W.L., Chang, C.C.: Multilevel reversible data hiding based on histogram modification of difference images. *Pattern Recognition* 41(12), 3582–3591 (2008)
33. Lou, D.C., Hu, M.C., Liu, J.L.: Multiple layer data hiding scheme for medical images. *Computer Standards and Interfaces* 31(2), 329–335 (2009)
34. Tai, W.L., Yeh, C.M., Chang, C.C.: Reversible data hiding based on histogram modification of pixel differences. *IEEE Trans. Circuits and Systems for Video Technology* 19(6), 906–910 (2009)
35. Thodi, D.M., Rodriguez, J.J.: Expansion embedding techniques for reversible watermarking. *IEEE Trans. Image Processing* 16(3), 721–730 (2007)
36. Tsai, P., Hu, Y.C., Yeh, H.L.: Reversible image hiding scheme using predictive coding and histogram shifting. *Signal Processing* 89(6), 1129–1143 (2009)
37. De Natale, F.G.B., Perra, C., Vernazza, G.: DCT information recovery of erroneous image blocks by a neural predictor. *IEEE Journal on Selected Areas in Communications* 18(6), 1111–1121 (2000)
38. Du, W.C., Hsu, W.J.: Adaptive data hiding based on VQ compressed images. *IEE Proceedings — Vision, Image and Signal Processing* 150, 233–238 (2003)
39. Nasrabadi, N.M., King, R.A.: Image coding using vector quantization: A review. *IEEE Trans. Communications* 36, 957–971 (1988)
40. Linde, Y., Buzo, A., Gray, R.M.: An algorithm for vector quantization design. *IEEE Trans. Communications* 28, 84–95 (1980)
41. Chang, C.C., Wu, W.C.: A steganographic method for hiding secret data using side match vector quantization. *IEICE Trans. Information and Systems* E88-D, 2159–2167 (2005)
42. Kim, T.: Side match and overlap match vector quantizers for images. *IEEE Trans. Image Processing* 1, 170–185 (1992)
43. Lin, S.D., Shie, S.C.: Side-match finite-state vector quantization with adaptive block classification for image compression. *IEICE Trans. Information Systems* E83-D, 1671–1678 (2000)
44. Tsai, J.C., Hsieh, C.H., Hsu, T.C.: A new dynamic finite-state vector quantization algorithm for image compression. *IEEE Trans. Image Processing* 9, 1825–1836 (2000)
45. Chen, C.C., Chang, C.C.: High capacity SMVQ-based hiding scheme using adaptive index. *Signal Processing* 90, 2141–2149 (2010)
46. Hu, Y.C.: High-capacity image hiding scheme based on vector quantization. *Pattern Recognition* 39, 1715–1724 (2006)
47. Lee, C.F., Chen, H.L., Lai, S.H.: An adaptive data hiding scheme with high embedding capacity and visual image quality based on SMVQ prediction through classification codebooks. *Image and Vision Computing* 28, 1293–1302 (2010)
48. Shie, S.C., Lin, S.F., Fang, C.M.: Adaptive data hiding based on SMVQ prediction. *IEICE Trans. Information and Systems* E89-D, 358–362 (2006)
49. Chang, C.C., Kieu, T.D., Chou, Y.C.: Reversible information hiding for VQ indices based on locally adaptive coding. *Journal of Visual Communication and Image Representation* 20, 57–64 (2009)
50. Chang, C.C., Wu, W.C., Hu, Y.C.: Lossless recovery of a VQ index table with embedded secret data. *Journal of Visual Communication and Image Representation* 18, 207–216 (2007)
51. Yang, C.H., Lin, Y.C.: Fractal curves to improve the reversible data embedding for VQ-indexes based on locally adaptive coding. *Journal of Visual Communication and Image Representation* 21, 334–342 (2010)
52. Yang, C.H., Lin, Y.C.: Reversible data hiding of a VQ index table based on referred counts. *Journal of Visual Communication and Image Representation* 20, 399–407 (2009)

53. Yang, C.H., Wu, S.C., Huang, S.C., Lin, Y.K.: Huffman-code strategies to improve MFCVQ-based reversible data hiding for VQ indexes. *Journal of Systems and Software* 84, 388–396 (2011)
54. Atallah, M.J., Raskin, V., Crogan, M., Hempelmann, C., Kerschbaum, F., Mohamed, D., Naik, S.: Natural Language Watermarking: Design, Analysis, and a Proof-of-Concept Implementation. In: Moskowitz, I.S. (ed.) *IH 2001*. LNCS, vol. 2137, pp. 185–199. Springer, Heidelberg (2001)
55. Barilnik, S.S., Minin, I.V., Minin, O.V.: Adaptation of text steganographic algorithms for HTML. In: *Proc. Int'l Siberian Russian Workshop and Tutorial on Electron Devices and Materials*, pp. 225–228 (2007)
56. Huang, H., Zhong, S., Sun, X.: An algorithm of webpage information hiding based on attributes permutation. In: *Proc. Int'l Conf. Intelligent Information Hiding and Multimedia Signal Processing*, pp. 257–260 (2008)
57. Liu, J.A., Lu, H., Fang, D.Y., Gui, X.L.: A text digital watermarking for chinese word document. In: *Proc. Int'l Symp. Computer Science and Computational Technology*, vol. 2, pp. 217–220 (2008)
58. Memon, A.G., Khawaja, S., Shah, A.: Steganography: A new horizon for safe communication through XML. *Journal of Theoretical and Applied Information Technology* 4, 187–202 (2008)
59. Sun, X.M., Chen, H.W., Yang, L.H., Tang, Y.Y.: Mathematical representation of a chinese character and its applications. *International Journal of Pattern Recognition and Artificial Intelligence* 16, 735–747 (2002)
60. Wang, Z.H., Chang, C.C., Lin, C.C., Li, M.C.: A reversible information hiding scheme using left-right and up-down chinese character representation. *Journal of Systems and Software* 82, 1362–1369 (2009)
61. Yu, J., Wang, X., Li, J., Nan, X.: A fragile document watermarking technique based on wet paper code. In: *Proc. Int'l Conf. Intelligent Information Hiding and Multimedia Signal Processing*, pp. 25–28 (2008)
62. Liu, T.Y., Tsai, W.H.: A new steganographic method for data hiding in microsoft word documents by a change tracking technique. *IEEE Trans. Information Forensics and Security* 2, 24–30 (2007)
63. Chang, C.C., Lee, C.F., Chuang, L.Y.: Embedding secret binary message using locally adaptive data compression coding. *Journal of Computer Sciences and Engineering Systems* 3, 55–61 (2009)
64. Bentley, J.L., Sleator, D.D., Tarjan, R.E., Wei, V.K.: A locally adaptive data compression scheme. *Communications of the ACM* 29, 320–330 (1986)
65. Chen, C.C., Chang, C.C.: High-capacity reversible data-hiding for LZW Codes. In: *Proc. Int'l Conf. Computer Modeling and Simulation*, vol. 1, pp. 3–8 (2010)
66. Shim, H.J., Ahn, J., Jeon, B.: DH-LZW: Lossless data hiding in LZW compression. In: *Proc. Int'l Conf. Image Processing*, vol. 4, pp. 2195–2198 (2004)
67. Huffman, D.A.: A method for the construction of minimum redundancy codes. In: *Proceedings of the IRE*, vol. 40, pp. 1098–1101 (1952)
68. Chung, K.L.: Efficient Huffman decoding. *Information Processing Letters* 61, 97–99 (1997)

---

# A Fast and Low-Distortion Capacity Adaptive Synchronized Acoustic-to-Acoustic Steganography Scheme

Xuping Huang<sup>1</sup>, Yoshihiko Abe<sup>2</sup>, and Isao Echizen<sup>3</sup>

<sup>1</sup> School of Multidisciplinary Sciences,  
The Graduate University for Advanced Studies (SOKENDAI),  
2-1-2 Hitotsubashi, Chiyoda-ku,  
Tokyo, Japan 183-8512  
huang-xp@nii.ac.jp

<sup>2</sup> Software Information Science Faculty,  
Iwate Prefectural University,  
152-52 Sugo, Takizawa,  
Iwate, Japan 020-0193  
yoshi@iwate-pu.ac.jp

<sup>3</sup> National Institute of Informatics,  
2-1-2 Hitotsubashi, Chiyoda-ku,  
Tokyo, Japan 183-8512  
iechizen@nii.ac.jp

**Summary.** Data transmissions in public communications systems are not secure because of the chance of their being intercepted, and tampered with by eavesdroppers. The security of acoustic data is an important issue. Particularly, a real-time acoustic steganography is required in a public broadcasting or mobile communication system. This chapter proposes a new steganography scheme with capacity variability and synchronization for secure transmission of acoustic data. In this scheme, the sender records acoustic digital WAV data, converts it into adapted PCM properties, and divides data stream into size-fixed frames. Then it embeds synchronous secret data into fix-sized cover frames, interleaves and transmits the secret data to generate stego data. The receiver extracts the secret data in real-time. A short execution time ensures synchronization, which in turn ensures scalability of privacy protection in live broadcasts. The embedding capacity is chosen as a key depending on the imperceptibility of the scrambling bits of the cover data. A socket model is used to transmit the stego data. Objective (signal to noise ratio) measurements and subjective evaluations demonstrate the effectiveness of the scheme.

## 9.1 Introduction

Ubiquitous channels are now available for delivering multimedia information. However, data transmissions in public communications system are not secure because

of the chance of their being intercepted and/or tampered with by eavesdroppers. Steganography using different media content has been proposed as a way of enhancing information security. Steganography means embedding messages in such a way that only the sender and intended recipient can access the secret information. By imperceptibly scrambling bits of cover data and using a secret key, steganography embeds secret information in comprehensible materials in such a way that does not draw the suspicion of interceptors or eavesdroppers. The current work in this area covers embedding metadata (text or image) into other media (text, image, audio, video). Unlike watermarking, steganography cannot endure unintentional modifications or intentional attacks. Secret data, however, should be resistant to distortion and tampering. Since the secret data is hidden within the physical arrangement of the cover data, bit scrambling distorts the cover data. Steganography based on acoustic data takes advantage of human auditory system characteristics whereby distortion from the embedding process cannot be distinguished by the human ear.

The algorithm described in this chapter results in fewer artifacts of distortion and frees up more bit positions for embedding in comparison with previous approaches. In the case of acoustic data with a specific quantization in a certain time<sup>1</sup>, up to the 8<sup>th</sup> bit can be used for embedding without degrading the cover signal.

## 9.2 Problems with Conventional Methods

Empirical studies have been undertaken to embed metadata such as text and images in other media. The methods include least significant bit (LSB) embedding, discrete cosine transform (DCT) encoding, MP3Stego, spread spectrum, echo data hiding, etc. Several embedding approaches have been proposed [1]–[8]. Most of these studies focus on image, text, and audio, though only as the cover string, and their schemes have limitations as to

1. the type of metadata media,
2. synchronization requirements, and
3. data hiding capacity.

### 9.2.1 Difference between Hiding Secret Messages in Images and Acoustic Data

Hiding information in audio data is quite different from hiding it in image data because of the different physical properties of these data and certain technical issues. The signal processings for audio data and image data are however similar. Steganography used in electronic communication include steganographic coding inside of a transport layer, such as a file, or a protocol, such as TCP. Usually, files for Internet means are put into media types that are lossless, such as FLAC, WAV, png etc. Many

---

<sup>1</sup> For example, data sampled for 30 seconds with the following settings:

- (1) Cover: 32 kHz, 16 bit, 2 channels;
- (2) Embed: 16 kHz, 8 bit, 1 channel.

methods refer to hiding information in unused space, such as the “header file”, or noise. Image data is represented in pixels, whereas audio data is divided into sampling points. Compression and data bit scrambling cause distortion that shows up in different ways in images and audio. Steganography based on images takes advantage of the human visual system, while steganography based on audio takes advantage of the human auditory system (HAS), which cannot distinguish slight changes. For example, one can hide a message inside a in a mono signal if the brightness does not vary much from one pixel and the next, or hide one in frequencies beyond RGB (Red, Green, and Blue). Meanwhile, up to 8kB can be embedded in audio without degrading the signal to the point that the secret communication becomes apparent [2].

## 9.2.2 Conventional Methods of Acoustic Steganography and Their Problems

The principal challenge is to use steganography when acoustic media is the target object. Acoustic data, especially speech data, is widely used. Privacy protection for mobile phones is now an important issue, and the current methods use cryptography. Although cryptography prevents eavesdropping by rendering the message incomprehensible, steganography is a more sophisticated method that camouflages secret data. That is, the secret data and the cover data may be speech data created by the same speaker. This makes it difficult for attackers to distinguish the secret message from the cover data.

Furthermore, acoustic media can conceal subliminal communication channels in, for example, live broadcasting, mobile communications, and teleconferences. However, audio (secret)-to-audio (cover) steganography has seldom been implemented for this purpose. Real-time signal processing is required in emergencies. Furthermore, data security could be enhanced by recording the secret data in a synchronized process. Many methods use LSB embedding, echo hiding, spectrogram, etc. Regarding LSB, although a psychoacoustic model can perceptually weight the noise and replaces audio in such a way that cannot be detected by HAS, the embedding capacity cannot meet the requirements for data transmission. Echo hiding exploits the offset or delay between cover data and stego data. The secret information is added to the initial amplitude [9, 10]. Added redundancy and limited frequency range for embedding are limitations of echo hiding.

The previous studies can be divided into two main approaches: fragile and robust information hiding. The fragile approach requires a balance between capacity and data quality to take advantage of human’s perception, while the robust information hiding requires algorithmic enhancements to prevent it from being attacked. Some features of the related work and the problem statement are compared as follows:

As referred in Table 9.1, in Neil [1] and Kusatsu’s [16] work, robust information hiding was implemented by using images in Matsuoka [3] and Petitcolas’s [12] work, and WAV and mp3 data was used as cover data. As mentioned above, the previous methods do not meet the needs of real-time tele-communications to thwart eavesdroppers. Real-time processing of tele-communications and live broadcasts is required to ensure data integrity and security. However, if a complex algorithm is

**Table 9.1.** Related work and their problems.

|                 | Real-time | Audio-to-Audio | Robustness |
|-----------------|-----------|----------------|------------|
| Neil [11]       | N/A       | N/A            | YES        |
| Kusatsu [16]    | N/A       | N/A            | YES        |
| Matsuoka [3]    | YES       | N/A            | YES        |
| Petitcolas [12] | N/A       | YES            | YES        |

used to keep the system robust, for example, one using random PN as the key, then the time required for embedding and extraction makes real-time communication impossible. Therefore, a balance has to be struck between robustness and processing time.

### 9.2.3 Contribution

This chapter proposes a new model and algorithm for a real-time steganography system. The system records a secret audio data stream synchronously in a WAV acoustic data stream. The embedding positions in the cover data with 16-bit sampling can be arbitrarily assigned. The main processes include PCM setting, embedding, and robustness analysis. The algorithm does a masking calculation to cover the bit stream at each sampling point. The capacity of the first signal at every sampling point of the cover data is calculated to evaluate the exact amplitude of the acoustic data. Our contributions are as follows:

The synchronized steganography of previous studies used the LSB scrambling approach, which makes it possible to assemble and utilize speech information securely to thwart intentional attacks [11, 12].

LSB is susceptible to bit scrambling in that the attacker can remove or disable hidden data simply by setting all the LSBs to be 0 or 1. Additionally, if only LSB is used, the amount of data will be small. Our approach has the following attributes:

#### (1) Embed secret acoustic audio in cover acoustic data

Analog acoustic signals interfere with each other when two acoustic streams are played at the same time [13, 14]. Furthermore, the chance of discrimination depends on environmental factors including the characteristics of the cover audio. The data sampling and interpolation algorithm makes it possible to embed speech data into white noise approaching the level of silence.



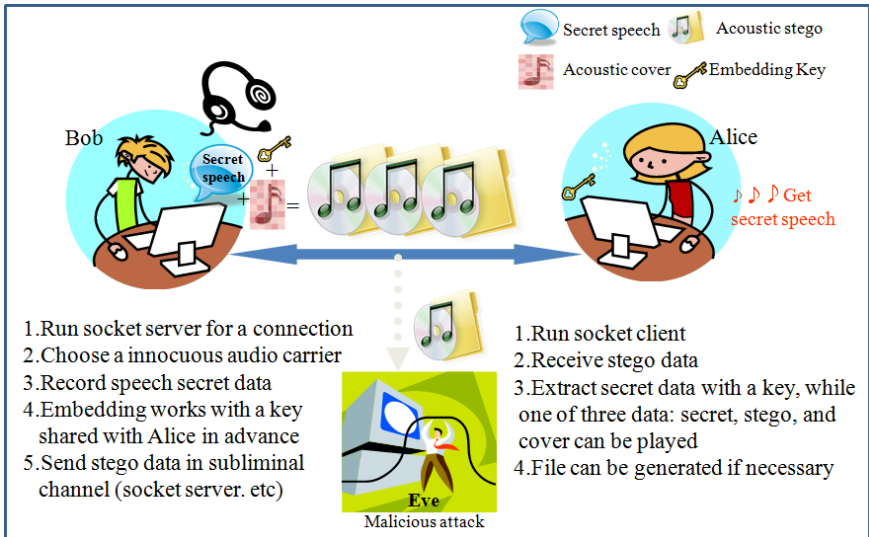
**(2) Significant bit embedding**

Even though multiple significant bit embedding affects the quality of stego data, the data shift and bit masking we propose reduce the degradation to indiscernible levels. The embedding bit positions can be specified by the sender at every sampling point of the cover data. Adaptive data PCM setting promises a variable embedding rate and a large data capacity.

**(3) Synchronized process**

Stego data can be generated even if the secret message is recorded when the cover data is being broadcasted. The attackers have trouble extracting such hidden data, and this makes it possible to avoid intentional attacks. It requires the algorithm to be fast and this may reduce robustness. Thus, the balance between complexity and execution time, including the embedding time and extraction time, has to be considered. With the purpose to enhance the algorithm complexity to attack, most methods use PN random numbers as the key. In our scheme, a 2-bit flexible embedding positions unit is used as the key and it can be arbitrarily specified by the sender when real-time hiding starts. Such flexibility makes prediction of embedding key more difficult as the robustness increases.

**9.2.4 Example Application**



**Fig. 9.1.** Illustration of synchronized acoustic steganography scheme.

Fig. 9.1 shows, an example application of the synchronized acoustic steganography scheme: secure real-time communication between two or more parties. A socket function is used to transmit the stego data. Attackers receive only the stego data stream by eavesdropping, whereas the sender and trusted receiver can exchange private speech information freely. To start a transmission, Alice (the intended receiver) starts the socket client to apply for a connection. After listening, Bob (the supposed sender) selects a piece of acoustic data as a cover and starts the socket server. When the socket starts, Bob records the secret message via microphone, and the speech is synchronously embedded in the cover audio data stream and carried in the stego data that is transmitted to Alice. Bob and Alice share the extraction program including the embedding key in advance. The secret-speech bit stream is extracted in real-time as the stego stream is being received by using the embedding key.

### 9.3 Synchronized Acoustic Steganography Using WAV Data

This chapter proposes a synchronized steganography system for acoustic data. The synchronous aspect means that the secret data is recorded and steganographically embedded at the same time as the cover is made, and the stego data is subsequently sent or broadcast to multiple receivers. However, only a trusted receiver can extract the secret data by using a secret key shared with the sender. The acoustic data is transmitted through socket communication.

#### 9.3.1 Statement of Purpose

Although many methods have been proposed, none of them can be used as a real-time scheme during emergencies when electronic monitors, or the Internet cannot be used. However, acoustic data can be carried through the air via radio in emergencies. Even though multimedia data can be put in multi-media channels of broadcasting, it is necessary to make sure that various channels are available in emergencies. Steganography provides a way to embed many kinds of data in the same carrier and not have them interfere with each other; it does not weaken the quality of the carrier data. The algorithm presented in this work permits a perceptual degradation only if the change to the data causes no noticeable difference in perceptual tolerance. Steganography can be used to protect valuable information from possible sabotage or unauthorized viewing, and it can use multiple media in emergencies. Fig. 9.2 shows the steganography scheme.

The purpose of this research is to secure private digital multi-media information sent through communications networks. The fidelity and integrity of the secret data must be assessed against the human auditory system. Live broadcasting and mobile communications are possible applications. There are three keywords phrases.

1. *Steganography*: To hide and transmit acoustic data through apparently innocuous acoustic carriers to conceal the existence of secret audio data;
2. *Real-time*: The secret audio data is recorded when the embedding scheme starts to run and is dynamically synchronized with cover sequences;

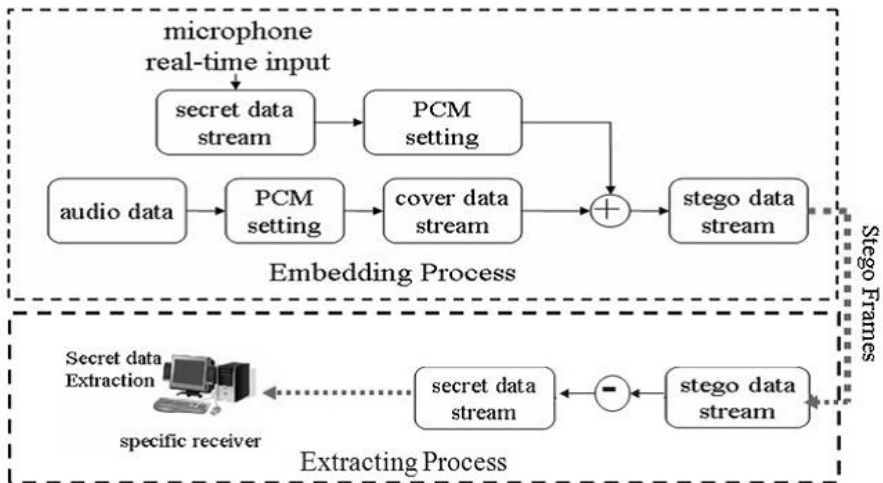


Fig. 9.2. Synchronized steganography scheme.

3. *Arbitrarily assigned multiple significant bits embedding*: A sufficient number of embeddable coefficients are present in each shuffled frame, though shuffling in significant bits may increase the sensitivity to intentional attacks aimed at rendering the embedded signals comprehensible. The embedding capacity and positions can be arbitrarily assigned up to the  $8^{th}$  significant bit from the least significant bit (LSB) at each sampling point of 16-bit cover data.

### 9.3.2 Acoustic Data Function

#### Data Function

Different from image and text data, the acoustical data format has a special function and header file. Sampling frequency, sampling size, and number of channels affect audio quality, which in turn affects embedding quality [15, 16, 17]. Data is stored in 8-bit bytes, arranged in Intel 80x86 format. This format suits the peculiarities of the Intel CPU, such as little-endian byte order, as shown in Fig. 9.3. Figs. 9.4 and 9.5 illustrate how to convert an analog signal into a digital signal. As Fig. 9.3 shows, multiple bytes are stored with the low-order (i.e., least significant bytes first.) As Fig. 9.4 shows, sampling is done at equal intervals  $t_1, t_2, t_n$ , etc. Sampling points with different amplitudes are represented with the same bit depth.

#### WAV Format Structure

A WAVE file is a collection of different chunks. The Format 'fmt' chunk contains important parameters of a waveform, such as sampling rate, and the data chunk contains the actual waveform data. These chunks are required, while other optional

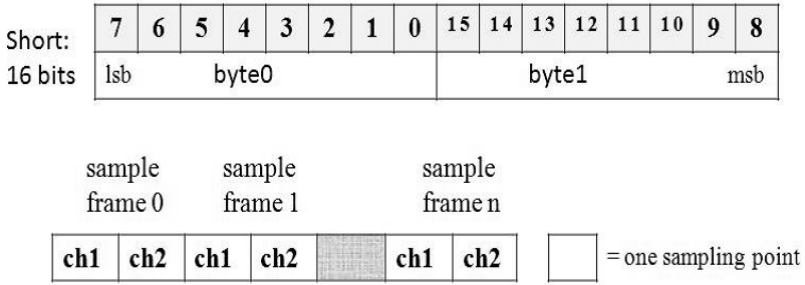


Fig. 9.3. Endian order of stereo data.

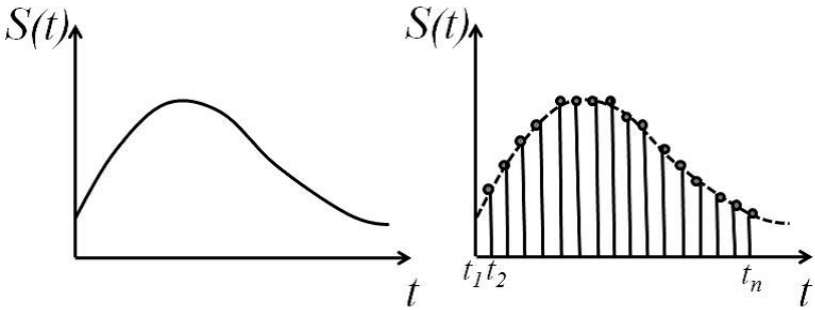


Fig. 9.4. Analog signal.

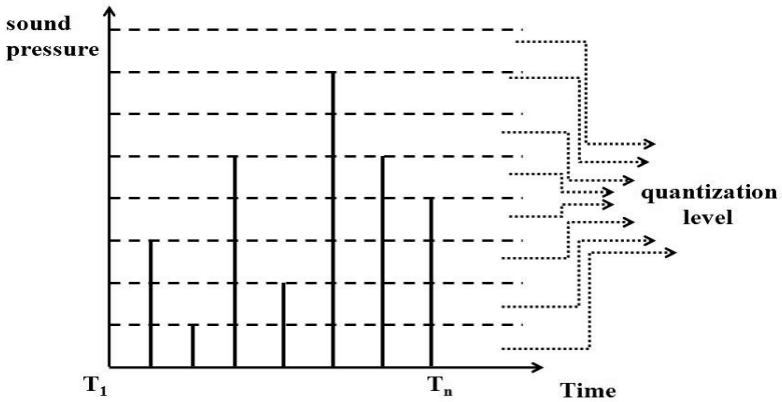


Fig. 9.5. Digital signal.

chunks included in the file structure define cue points, list instrument parameters, store application-specific information. Fig. 9.6 is a graphical overview of a minimal WAVE file consisting of a single WAVE with the two required chunks.

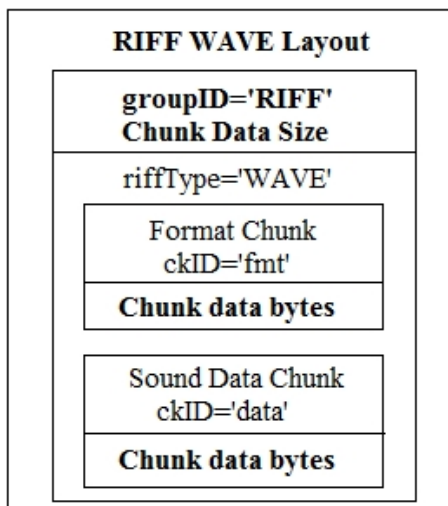


Fig. 9.6. Basic chunk file structure.

### WAV Header Information

After the secret data frames have been extracted from the stego stream, the WAV header information has to be appended to the frames to ensure the integrity of the secret data. Unlike image data, acoustic sequences require the correct time stamp. When the receiver gets continuous frames, the header file has to be appended to each frame to enable the embedded audio to be played. Table 9.2 lists the WAV data information that has to be appended to the extracted frames.

### 9.3.3 Principle of Synchronized Steganography

There are four steps in this method:

1. The secret data is recorded and embedded in real-time.
2. PCM information of the cover data and the embedded data are set to specified values. The playback lengths of the cover data and embedded data are equal.
3. The coded data (wave format) is divided into fix-sized frames suitable for streaming and reconstructing signals from two different signal sequences.
4. To playback the embedded data after it has been extracted from the stego data, it is necessary to append wave header information corresponding to its PCM information.

The system converts the analog input signal into digital data directly via microphone and embeds bit signals of secret data in the cover audio data. The real-time processing uses a narrow frequency band covering the span of frequency of the audio file. The real-time process is divided into steps during which the secret audio data is synchronously embedded in the cover audio data. Real-time pertains to the whole process of producing the secret data, embedding it in the cover, thereby creating the stego data, and extracting the secret data from the stego data.

**Point 1:** It is possible that a third party will get the transmitted information and the stego-key to render the stego-data vulnerable. To combat against this, the embedded data could be encrypted to make a hybrid encryption system. After the intended receiver extracts the embedded data with the stego key, the embedded data can be generated as a wave format file.

**Point 2:** The cover data is stored on a hard disk. Many different experiments using different types of music as the cover have shown that this method works effectively. For instance, we recorded natural background noise in an unoccupied lab at midnight and used it as a quiet cover to embed a loud piece of audio data.

**Point 3:** The cover data and secret data are as follows:

1. The cover is 32-kHz, 16-bit, 2-channel audio data.
2. The secret audio is 16-kHz, 8-bit, 1-channel data.
3. The cover data and secret data are synchronized; thus, a multiplicative factor determines the number of bits to be embedded in the sampling point.

**Point 4:** The stego data is sent through the Internet or a broadcasting system. The security and integrity of the secret data can be guaranteed even if the stego data is intercepted.

**Point 5:** Audio data are coded into a digital signal and then decoded. For this, it is necessary to append WAV header information to the extracted frames. Both the cover and the embedded data are divided into fix-sized frames, and bit scrambling is done on each frame of the cover.

### 9.3.4 Algorithms

#### Embedding Process

In Fig. 9.7 the sequence  $s(n)$  stands for stego data, which is modeled as a sample drawn from a sampling terminal.  $c(n)$  means there are  $n$  sampling points of cover data, and each point equals a 16-bit data stream. Two bits of secret data are embedded in the cover sampling point. The process of recording the secret data stream is synchronized with the embedding process. The data stream on the left side (presented in 16 bits) is the cover data  $c(n)$ , and the embedded data  $e(n)$  is on the right

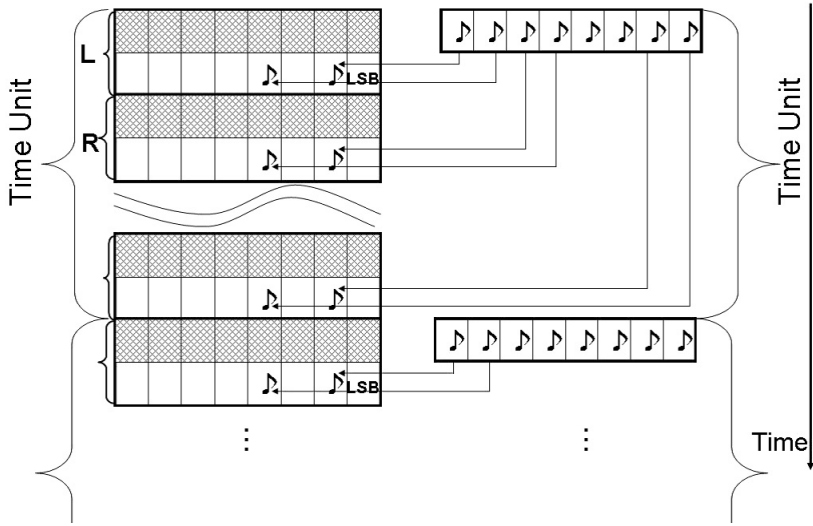


Fig. 9.7. Multi-bit embedding process.

side (presented in 8 bits). It is possible to embed two bits of secret data in two arbitrary bit positions  $[1^{st}, 8^{th}]$  from the least significant bit (LSB). Once the positions have been assigned, all the secret data stream will be embedded in the assigned bit positions.

There are three steps involved in embedding two bits of secret data.

**Input:** cover stream and secret stream;

**Output:** stego stream;

**Step E1:** Clear Data Bit in Cover:

The first step is to set the data at the embedding location to be “0”. A cmask is used to make the designated location available for secret data. The following function accomplishes this task.

$$\begin{aligned}
 cmask1 &= (2^{loc1-1}) \oplus (0xFF) \\
 cmask2 &= (2^{loc2-1}) \oplus (0xFF) \\
 cmask &= cmask1 \wedge cmask2
 \end{aligned}$$

**Step E2:** Copy Secret Data:

The second step is to put two bits of secret data into the bit location assigned by the user. Two consecutive bits of secret data are read from and written into the designated bit location; the other bits in the 8-bit data unit are cleared to be “0”. The following function accomplishes this task.

```

for( $i = 0, 0 \leq i \leq 4, i++$ )do
     $e1 = embed \gg (7 - 2i)$ 
     $e1 = e1 \& (0x01)$ 
     $e1 = e1 \ll (loc1 - 1)$ 
     $e2 = embed \gg (6 - 2i)$ 
     $e2 = e2 \& (0x01)$ 
     $e2 = e2 \ll (loc2 - 1)$ 

```

$i$  is the ordered sequence of 2-bit secret data units counting from the left of the insignificant 8-bit unit of cover data at the sampling point.

**Step E3:** Execute Embedding:

The third step generates the stego data as follows.

$$c = cover \& cmask \text{stego} = c | e1 | e2$$

**Embedding Example:**

The following example concretely illustrates this process. The designated positions are  $loc1 = 4$ ,  $loc2 = 2$ , and  $i = 1$ .

**Step E1:** Generate a  $cmask$ . The 2-bit unit  $\boxed{2}$  of the cover has 8 insignificant bits in which  $loc1$  and  $loc2$  are available for secret data.

| $8^{th}$ | $7^{th}$ | $6^{th}$ | $5^{th}$ | $4^{th}$ | $3^{rd}$ | $2^{nd}$ | lsb |
|----------|----------|----------|----------|----------|----------|----------|-----|
| 1        | 1        | 1        | 1        | 0        | 1        | 0        | 1   |

**Step E2:** Read two bits of secret data and put them into the designated bit positions by shifting and taking bitwise AND. Take the following data as an example: When  $i = 1$ , it means that the  $6^{th}$   $e1$  and  $5^{th}$   $e2$  bit data will be read and put into the  $2^{nd}$  and  $4^{th}$  bit positions of the cover. The consecutive bits of secret data are:

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

Accordingly, we embed the  $6^{th}$  ( $e1 = 1$ ) in the  $4^{th}$  LSB of the cover and the  $5^{th}$  ( $e2 = 0$ ) into the  $2^{nd}$  LSB. The same process is used to put  $e2$  in bit position  $loc2$ .

---

<sup>2</sup> The first 2-bits unit of the loop.  $i = 1$  means ( $7^{st}, 8^{th}$ ) bit positions from LSB.



**Step E3:** Embed  $e1$  and  $e2$  in the cover data prepared by  $cmask$ .

For example, a unit of cover data including 8 consecutive insignificant bits is:

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|

Data  $e1$  takes the place of the 4<sup>th</sup> bit position with the original data “0” in it, and similarly,  $e2$  takes the place of the 2<sup>nd</sup> bit position with the original data “1” in it. Thus, the stego data are:

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

### Extraction Process

Information hidden in the cover data cannot be extracted if only the extraction is successful and the key information is known [18, 19, 20]. To deal with the possibility that the attacker knows of the existence of the secret speech and somehow learns the key, say, by extracting data packages on the Internet, we can use socket transmission to improve the security of the secret data. Even though the attackers may get all of the data frames, the secret speech can not be accessed unless specific WAV header information is appended before the secret data frames. The header information includes the PCM set of the secret data and cover data, and the setting is performed synchronously on the cover data and speech bit stream while they are being read.

The PCM settings of the cover data and stego data are the same. When a trusted receiver uses the key and listens to the socket server, the header information can simply be fetched by copying the parameters from the server. When data transmission starts, the extraction program automatically executes the header appending program on the socket client terminal.

The extraction is done as follows:

**Input:** stego stream;

**Output:** embedded stream;

**Step D1:** Preparation for extraction:

$$mask_1 = 0X01 \ll (loc1 - 1);$$

$$mask_2 = 0X01 \ll (loc2 - 1);$$

**Step D2:** Extract signal from  $loc1$  and  $loc2$ ;  $i$  is the sequence number of 2-bit unit.

$$e1 = \sum_{i=0}^8 (stego(2i) \& mask_1) \gg (loc1 - 1)$$

$$e2 = \sum_{i=0}^8 (stego(2i) \& mask_2) \gg (loc2 - 1)$$

**Table 9.2.** Wav data header information.

| Size(bytes) | Value   | Description   |
|-------------|---|---|
| 4           | 'R' 'I' 'F' 'F'(0x52494646)                             | chunk ID  |
| 4           | (file size)-8   | chunk size  |
| 4           | 'W' 'A' 'V' 'E' (0x57415645)                            | RIFF type   |
| 4           | 'f' 'm' 't' ' ' (0x666D7420)                            | format space  |
| 4           | 16 (10 00 00 00) linear PCM                             | byte of fmt chunk   |
| 2           | 1 (01 00) linear PCM                                    | assumed format ID   |
| 2           | mono 1 (01 00) stereo 2(02 00)                          | number of channels  |
| 4           | 32000Hz (32 AC 00 00)                                   | sampling rate   |
| 4           | 128000(00 F4 01 00)<br>(32000*2*2=128000)               | data transfer speed<br>32 kHz 16-bit stereo                           |
| 2           | 1*1=1 (01 00) 8-bit mono<br>2*2=4 (04 00) 16-bit stereo | block alignment   |
| 2           | 8 (08 00 00 00)<br>16 (10 00 00 00)                     | significant bits per sample   |
| 2           | null (if linear PCM)                                    | size of extra format bytes  |
| n           | 0 -65535 (null if linear PCM)                           | extra format bytes  |
| 4           | 'd' 'a' 't' 'a'   | data chunk ID   |
| 4           | n   | chunk size, type: dword depending on<br>sample length and compression |
| n           | audio data length                                       | wave format chunk values  |

**Step D3:** Create embed signal;  $p$  is the index of the extracted steam.

$$embed[j] = (e1 \ll 6 - 2p)$$

$$embed[j] = (e2 \ll 5 - 2p)$$

**Step D4:** Append WAV header to extracted stream; The extracted stream will have errors and the content will not be recognized unless the WAV header is appended. The information is edited according to the PCM settings. The header information process is included in the extraction process (see Table 9.2).

**Extraction Example:**

Data in buffer shown as follows plots stego stream on the  $x$ -axis and time-scale on the  $y$ -axis. Each sampling point is represented by 16 bits. The least significant bits of each sampling point are 1, 0, 1, and 1 and secret data is only included in the 8-bit unit counting from LSB on the right side.

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

The the extracted bit stream is:

01111001

## 9.4 Experimental Evaluation

The cover data are files with specific PCM settings and the embedding process starts when secret acoustic data are recorded by microphone. The embedding should be of high quality, which means imperceivable even if a large amount of data is embedded in the cover. The extraction reliability of a steganography scheme generally depends on the features of the original data, on the embedding distortion, and on the robustness (the amount of calculations needed by the attacker [21]). There are two ways of evaluating the embedding quality: subjective and objective. The mean opinion score (MOS) provides a numerical indication of the perceived quality of received media after compression or transmission, and it is calculated by averaging the results of a set of standard subjective tests in which a number of listeners rate the audio quality of test sentences read aloud by male and female speakers over the communications medium being tested.

### 9.4.1 Evaluated Data

The steganographic capacity, embedding capacity, embedding efficiency, and data payload affect the embedding quality.

To specify the properties of the acoustic signals in our experiment, we used the first “1” signal from the significant bit (16<sup>th</sup> bit) at each sampling point. We calculated the number of sampling points that had the same bit position as the first “1” signal to determine the physical volume of the signal in an objective way.

Furthermore, there are trade-offs between imperceptibility, differentiation between cover and stego, and data classification. The data classification was thus based on volume, bit-rate, and variety of audio files. Audio magnitude was measured by counting sampling points where the first “1” signal appeared at the same bit position. If most first “1”s appeared at significant bit positions, the auditory impression was of noisiness.

Fig. 9.8 plots bit position on the  $x$ -axis and sampling points on the  $y$ -axis<sup>3</sup>. The objective measurement’s results matched those of subjective auditory impression. Most of the first “1”s in the jazz and classical music data were respectively from the 6<sup>th</sup> to 15<sup>th</sup>, and the 12<sup>th</sup> and 13<sup>th</sup> bits. The speech files thus had about 150000 sampling points that had the first bit in the 14<sup>th</sup> bit. In contrast, background noise was relatively low when most first “1”s were in the least significant bit position.

We also evaluated our method on the RWC Music Database [23]. We did an experiment with an L-channel waveform with a 44.1-kHz sampling and 16-bit quantization. We used 10 tracks in RWC-MDB-G2001 as cover data and a piece of speech

<sup>3</sup> Each piece of sampled data was 30 seconds long (32 kHz, 16 bits, two channels).

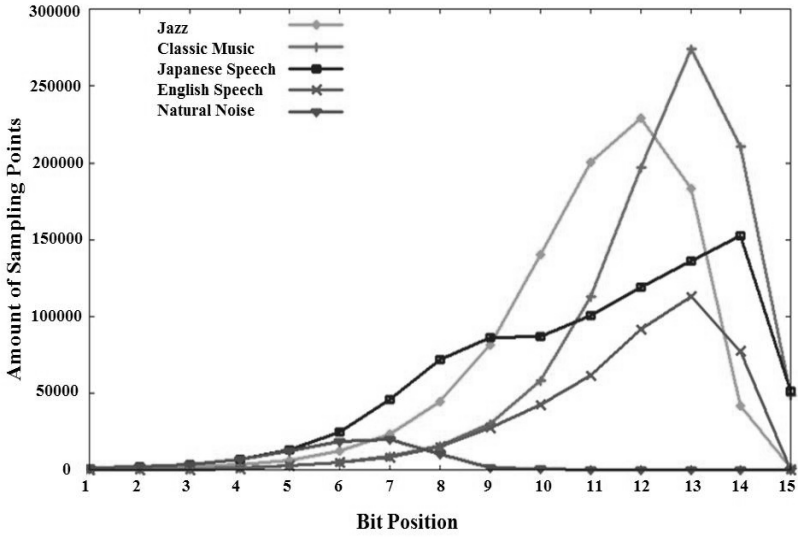


Fig. 9.8. First “1”: Structural information of significant bits.

data ATR 503 PB-5 from Japanese Newspaper Article Sentences (JNAS) [24] published by the Acoustical Society of Japan as the embed data. For this experiment, the cover data was re-sampled with and the secret data was re-sampled with 32000 samples at 8 kHz and 8 bits, in mono with about 4 seconds playback time for the evaluation. The signal power level is plotted in Table 9.3. It indicates Track No.80, No.90 and natural noise had a low power level.

### 9.4.2 Evaluation Methods of Embedding Quality

#### SNR Analysis

As a measure of embedding quality, the SNR of each stego data can be calculated from the quantization error of each sampling point of cover and stego data as follows:

$$SNR = 10 \log_{10} \frac{\sum_n c^2}{\sum_n d^2(c, s)} [dB] \tag{9.1}$$

Here,  $c$  stands for the original signal,  $s$  stands for the stego signal, and  $d$  stands for the relative difference in amplitude between the corresponding sampling points of the cover and stego data, with  $n$  meaning the number of sampling points during a certain time interval.

#### MOS Evaluation of Imperceptibility

Besides the SNR evaluation, we took the subjective MOS (mean opinion score) to be an indicator of the perceived quality of the stego data. About 15 members of our lab

**Table 9.3.** Signal Power Level of Evaluated Data

| Track                                | L-Channel(dB) | R-Channel (dB) |
|--------------------------------------|---------------|----------------|
| No.1                                 | -32.043       | -31.826        |
| No.10                                | -18.571       | -18.573        |
| No.20                                | -18.469       | -18.146        |
| No.30                                | -19.759       | -19.962        |
| No.40                                | -24.530       | -24.534        |
| No.50                                | -23.042       | -21.340        |
| No.60                                | -27.623       | -32.344        |
| No.70                                | -18.628       | -17.915        |
| No.80                                | -45.689       | -43.754        |
| No.90                                | -45.321       | -41.891        |
| No.100                               | -28.052       | -27.481        |
| Japspeech32-16-2.wav <sup>a</sup>    | -13.436       | -13.192        |
| Engspeech32-16-2.wav <sup>b</sup>    | -15.992       | -13.518        |
| Classical32-16-2.wav <sup>c</sup>    | -11.044       | -13.981        |
| NaturalNoise32-16-2.wav <sup>d</sup> | -41.998       | -30.051        |
| ATR 503 PB-5                         | -29.153       | null           |
| Japspeech8-8-1.wav <sup>e</sup>      | -13.431       | null           |
| pop8-8-1.wav <sup>f</sup>            | -14.079       | null           |
| Rock8-8-1.wav <sup>g</sup>           | -12.732       | null           |

<sup>a</sup> Japanese speech(Male) sampled at 32kHz, 16bits and stereo, 30 seconds.

<sup>b</sup> English speech(Female) sampled at 32kHz, 16bits and stereo, 30 seconds.

<sup>c</sup> Classic music sampling at 32kHz, 16bits and stereo, 30 seconds.

<sup>d</sup> Nature weighted noise recorded at midnight indoors at 32kHz, 16bits and stereo, 30 seconds.

<sup>e</sup> Japanese speech(Male) sampled at 8kHz, 8bits and mono, 30 seconds.

<sup>f</sup> Pop music sampled at 8kHz, 8bits and mono, 30 seconds.

<sup>g</sup> Rock music sampled at 8kHz, 8bits and mono, 30 seconds.

rated the quality of the stego data. Cover data and stego data were played at random. Listeners were required to distinguish the displayed music was stego data or cover data. Then, they rated the embedding quality by using the ranking of Table 9.4.

### 9.4.3 Experiments and Results

#### Experiment 1: Evaluation of Extreme Case

Digital speech signals typically have white noise as well as quiet parts. To test whether it is possible to embed an acoustic signal sequence in another quiet signal, we recorded natural background noise occurring at the midnight in our lab as cover data. The secret Japanese speech data was recorded when the embedding process started. To analyze the differences between the stego and cover data, the cover quiet natural noise was initially normalized to “0”.

**Table 9.4.** Mean Opinion Score

| MOS | Embed Quality | Impairment               |
|-----|---------------|--------------------------|
| 5   | Excellent     | Imperceptible            |
| 4   | Good          | Difficult to Distinguish |
| 3   | Fair          | Slightly Annoying        |
| 2   | Poor          | Different Audio          |
| 1   | Bad           | Annoying                 |

The results of test with weighted noise cover data are shown in Figs. 9.9(a),(b), and(c). the stego data sounded like quiet natural noise even when it had high-amplitude speech embedded in it.

### Experiment 2: Evaluation by Significant Bit

This experiment tried to determine whether our method is robust when secret data is embedded into significant bit positions in the cover. In it, we

1. considered the interference spectrum of the significant-bit embeddings,
2. designated the bit locations:  $1^{st}$  and  $2^{nd}$ ,  $1^{st}$  and  $4^{th}$ , etc., and
3. repeated the embedding process with two designated bit positions from the  $1^{st}$  bit (LSB) to the  $n^{th}$  bit ( $n \in [1, 8]$ ) of the cover. Once the embed positions were decided, the secret data was embedded in those positions of successive frames of the cover.

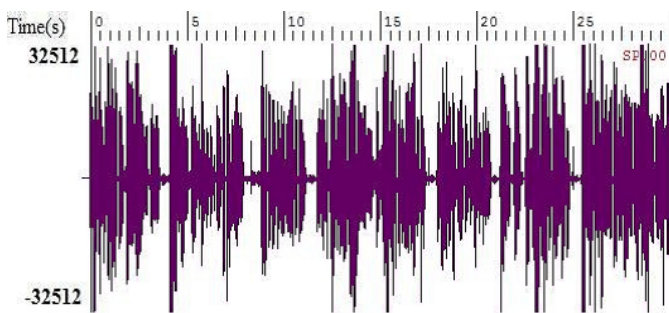
We did simulations to see how much of a difference there is between stego data with embeddings at positions with different levels of significance. We found that

1. the embedding quality was the best when the secret data was embedded in the least significant bit positions:  $1^{st}$  and  $2^{nd}$ ;
2. it was still difficult to distinguish the differences between cover data and stego data even when the secret data was embedded in the  $7^{th}$  and  $8^{th}$  significant bit positions.
3. The SNR ranks were [52.15, 88.28], and there were few differences when one of the embedding positions was the  $8^{th}$  bit (group (1,8), (5,8), (7,8), as shown in Fig. 9.10.

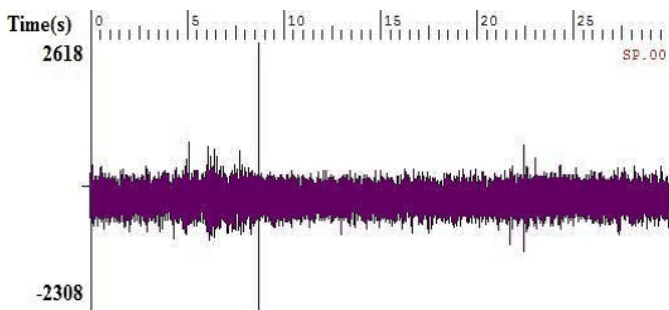
### Experiment 3: Evaluation by Audio Category

This experiment tested whether music data of different genre should have different embedding efficiencies. We

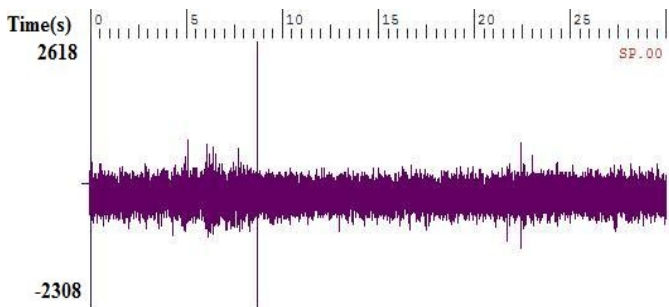
1. considered the efficiency of the data structures' coincidence,
2. used music data bit streams (the cover and embedded data were of the same category) of jazz, pop, rock, classic, natural noise, and speech, and
3. set the embedding position to be the  $7^{th}$  and  $8^{th}$  significant positions.



(a) Embedded data: Japanese speech



(b) Cover data: natural noise



(c) Stego data: natural noise with speech

**Fig. 9.9.** Embed speech in weighted natural noise.

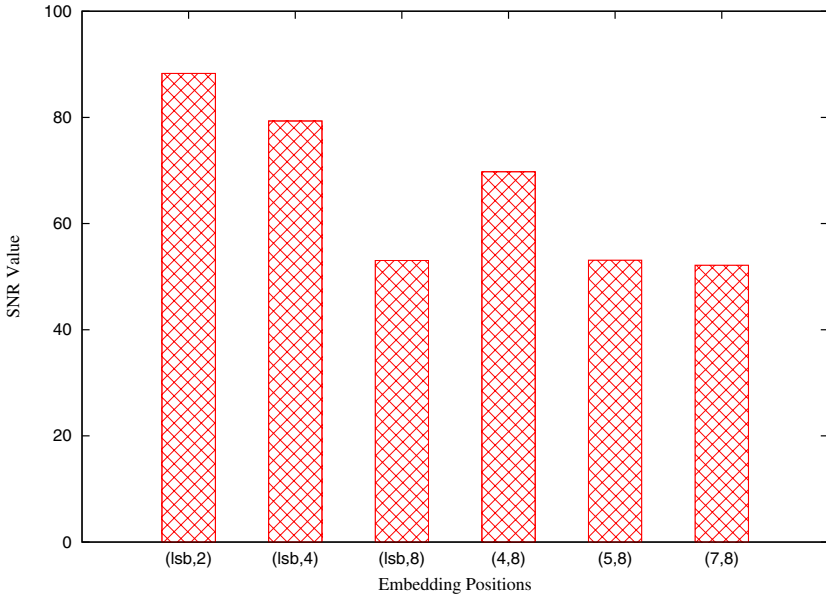


Fig. 9.10. SNR of 2-bit unit embedding in different bit positions.

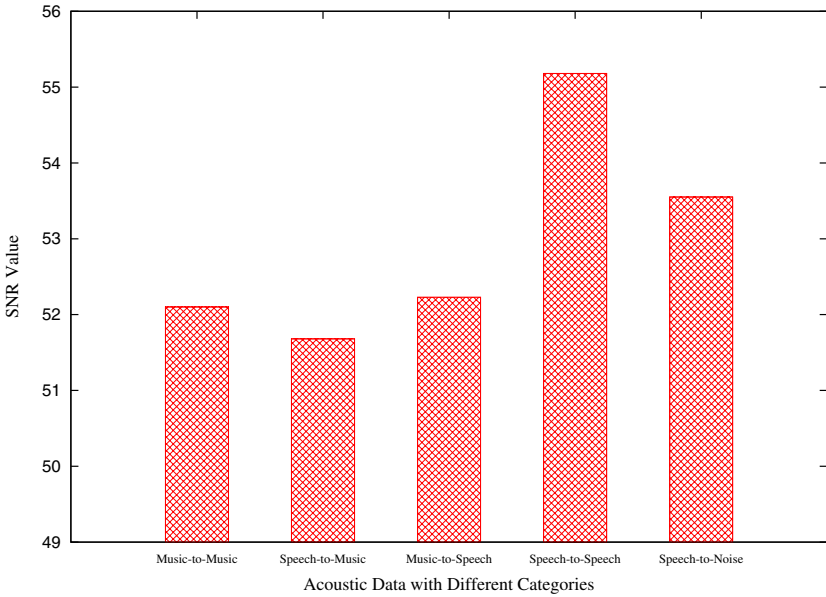


Fig. 9.11. SNR for different audio categories



**Table 9.5.** System configuration

|                             |  |
|-----------------------------|--|
| <b>OS</b>                   | Windows Vista Business                 |
| <b>CPU</b>                  | Intel Core 2 Quad CPU Q9650 @ 3.00 GHz |
| <b>Memory(RAM)</b>          | 2GB                                    |
| <b>Programming Language</b> | JAVA                                   |

As shown in Fig. 9.11

1. There were no obvious differences in embedding efficiency using different acoustic categories;
2. The embedding system worked even when speech was embedded in white noise;
3. The SNR ranks [51.68,55.18]. The embedding quality was the best when the cover and secret data were both *speech* data. In this case, the cover and embed data consisted of a large speech data streams. Furthermore, the more the cover and embed data coincided, the better the embedding quality was.

### Subjective MOS Results

According to auditory impression, most of the listeners said they could not distinguish between the cover data and stego data at all (MOS score: 5), while two of them gave MOS scores of 4, meaning that they found it difficult but possible to distinguish the two audio data. These results indicated to us that the embedding was successful and indicated a low audio-quality distortion for most tunes after data embedding.

#### 9.4.4 Evaluation of Real-Time Performance

We determined that it would be possible to have real-time secure communications between two PC terminals by recording, embedding, transmitting, and extracting the secret data in real-time. We performed an experiment to check whether the stego data could be generated and transmitted to another host. The cover data was sampled for about 30 seconds. The sender used a microphone to record the secret speech while the cover data was being played as a stream. Stego data was generated by the algorithm and sent to the receiver terminal.

To assess the running times of hiding and extraction, we embedded speech in other speech with LSB insertion. The running times with different hiding bit positions and different audio categories are supposed to be similar. In experiment A, we recorded secret data in real-time when the hiding process started (stream-into-file) ; in experiment B, we embedded the wav speech file generated in experiment A (secret data) in another speech (English, Female) file (file-into-file). The specifications of the PC we used in this experiment are listed in Table 9.5. Many experiments were run and Table 9.6 and Table 9.7 lists the maximum, minimum and average values.

According to ITU-T Y.1541 Standard, for telephony to be successful, callers can speak simultaneously if the latency is 150 milliseconds or less [22]. In this chapter, the maximum hiding time is 0.9997 microseconds and the extraction time is

**Table 9.6.** Real-time ability: Experiment A (with real-time recording)

| Source signal          | Hiding time ( $\mu$ s) | Extraction time( $\mu$ s) |
|------------------------|------------------------|---------------------------|
| Music                  | 0.733                  | 0.5997                    |
| Natrue Noise           | 1.3329                 | 0.8664                    |
| Speech English(Female) | 1.3329                 | 0.5998                    |

**Table 9.7.** Real-time ability: Experiment B

| Value   | Hiding time ( $\mu$ s) | Extraction time( $\mu$ s) |
|---------|------------------------|---------------------------|
| Maximum | 0.9997                 | 0.9330                    |
| Minimum | 0.0666                 | 0.0666                    |
| Average | 0.4704                 | 0.26655                   |

0.9330 microseconds, which means hiding and extraction are done in real-time, as shown in Table 9.7. Furthermore, by comparing the running-time performances, it became clear that there was little difference between hiding a stream in a file (while recording in real-time) and hiding a file in another file.

## 9.5 Conclusion

We developed and evaluated a real-time multi-bit audio-to-audio steganography method. Embedding quality varies according to the significant bit of the cover data in which the secret data is embedded. In this method, secret speech data is recorded using a microphone and embedded in acoustic cover data synchronously. The intended receiver extracts the secret speech data from the stego data using a shared secret key. We measured the embedding performance of different acoustic categories.

Our method had optimal embedding performance even when 2 bits of secret data were simultaneously embedded in the 7<sup>th</sup> and 8<sup>th</sup> bit positions at a sampling point of the cover data stream. Subjective tests proved that it is difficult to distinguish the cover data from the stego data.

The SNR analysis revealed a few differences when the acoustic data came from different categories. Socket covert channels can enhance the robustness of this scheme.

## 9.6 Future Work

We used a two-bit arbitrarily assignment embedding model in our experiment. In the future, we shall analyze the coincidence between the cover and the embedded data before evaluating the embedding performance. Furthermore, we can extend our method to 4-bit embedding. It still remains for us to work out whether it is safe to

embed more continuous secret data bits and how many bits at most can be embedded in a sampling point of the cover at one time.

1. The embedding positions were arbitrarily assigned. However, once the embedding positions are determined, all of the secret data will be embedded in these specified positions. This is a weak point that can be exploited by statistical analysis or repetition attacks. This chapter did not discuss extraction or countermeasures against such attacks. Steganalysis will be considered in a future study.
2. Our scheme uses only wav data. Wav data has high acoustic quality but its amount grows large if the sampling rate is high. Speech recognition doesn't require high acoustic quality, so embedding together with a synchronized compression process should be considered.

## Appendix

Since embedding quality depends on acoustic source, we performed laboratory experiments to validate the proposed algorithm using different acoustic files as shown in Figs. 9.12, 9.13, 9.14, 9.15, and 9.16. Groups of sample data were used for the validation. They were:

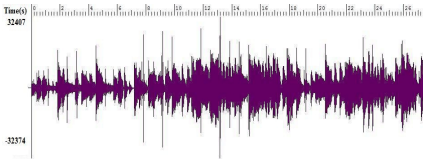
1. Group 1: Embed Music (Rock) into Music (Jazz)
2. Group 2: Embed Music (Pop) into Speech (JPNSpeech-Male)
3. Group 3: Embed Speech (JPNSpeech-Male) into Music (Classic)
4. Group 4: Embed Speech (JPNSpeech-Male) into Noise (Background Natural Noise Recorded at Midnight in the Lab)
5. Group 5: Embed Speech (JPNSpeech-Male) into Speech (ENGSpeech-Female)

To check the embedding capacity, the embedding positions were assigned to be the 7<sup>th</sup> and 8<sup>th</sup> bits. The figures plot the frequency of the difference between the stego signal and cover signal, for the spectrum<sup>4</sup> and cepstrum<sup>5</sup> information. The graphs with the sound pressure values<sup>6</sup> on the y-axis and time on the x-axis((a),(b),(c), and (d)) show the target data in for each results graph. The difference graphs (d) illustrate the difference between stego data and cover data. To detect the tiny variation, we zoom in on the y-axis.

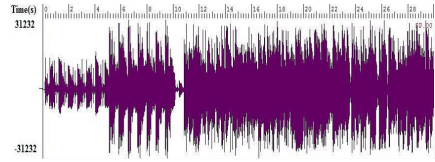
<sup>4</sup> This spectrum signal was generated from a pre-emphasized and windowed section of the signal.

<sup>5</sup> The cepstrum is the spectrum of the log magnitude spectrum of the signal.

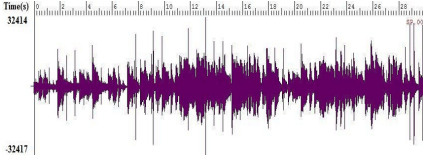
<sup>6</sup> Each sample point was stored as a linear, 2's-complement value. Complement here is the significant bit order, for example, 7<sup>th</sup> (complement value is 7) 8<sup>th</sup>, etc.



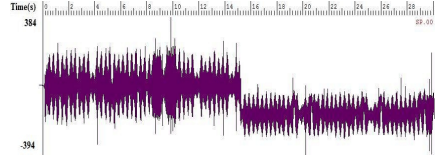
(a)Cover: Jazz32-16-2.wav



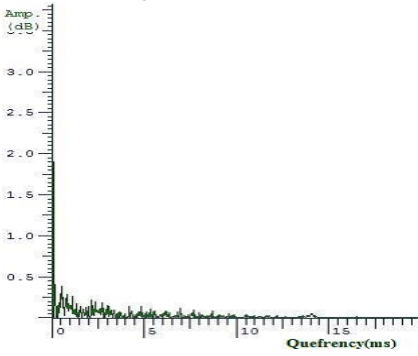
(c)Embed: Rock-8-8-1.wav



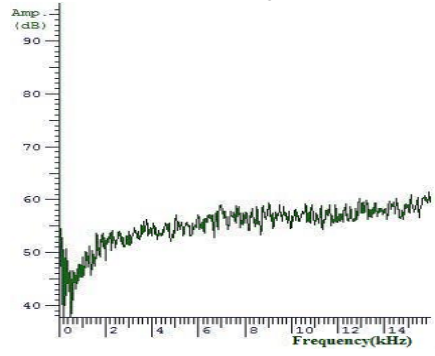
(b)Stego: Rock-to-Jazz.wav



(d)Difference: Stego/Cover

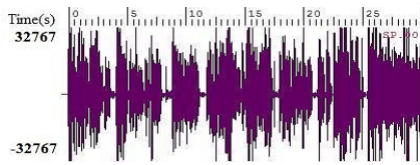


(d1)Cepstrum of difference

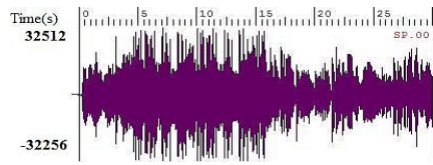


(d2)Spectrum of difference

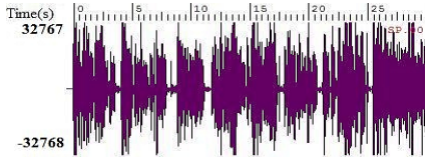
**Fig. 9.12.** Group 1: Embed Music into Music



(a)Cover: JPNSpeech32-16-2.wav



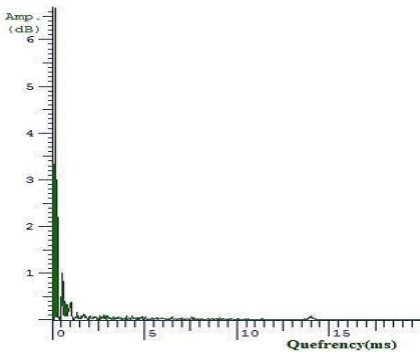
(c)Embed: Pop-8-8-1.wav



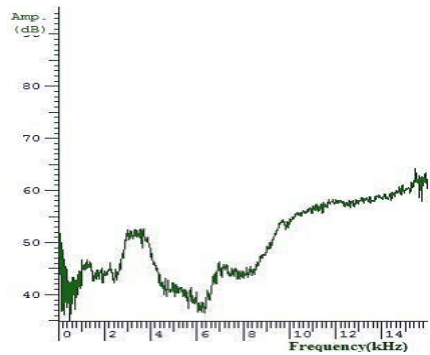
(b)Stego: Pop-to-JPNSpeech.wav



(d)Difference: Stego/Cover

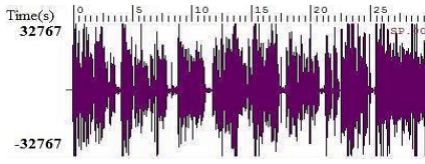


(d1)Cepstrum of difference

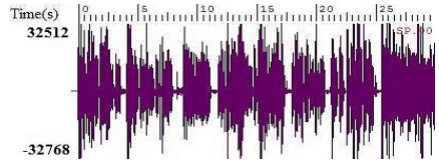


(d2)Spectrum of difference

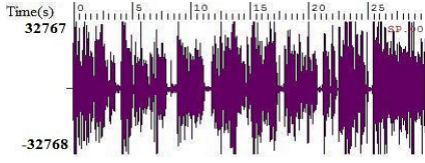
**Fig. 9.13.** Group 2: Embed Music into Speech



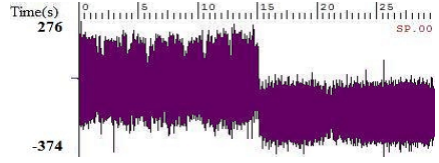
(a)Cover: Classical32-16-2.wav



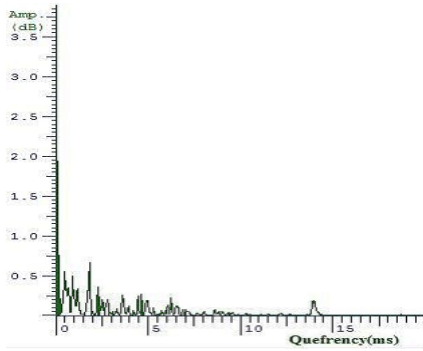
(c)Embed:JPNSpeech-8-8-1.wav



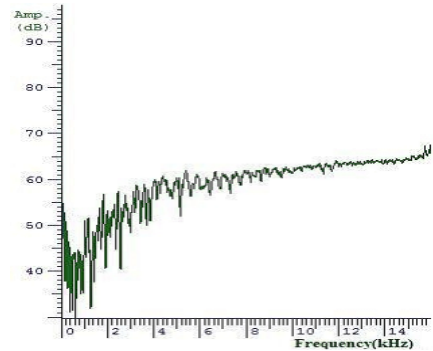
(b)Stego: JPNSpeech-to-Classic Music.wav



(d)Difference: Stego/Cover



(d1)Cepstrum of difference



(d2)Spectrum of difference

Fig. 9.14. Group 3: Embed Speech into Music

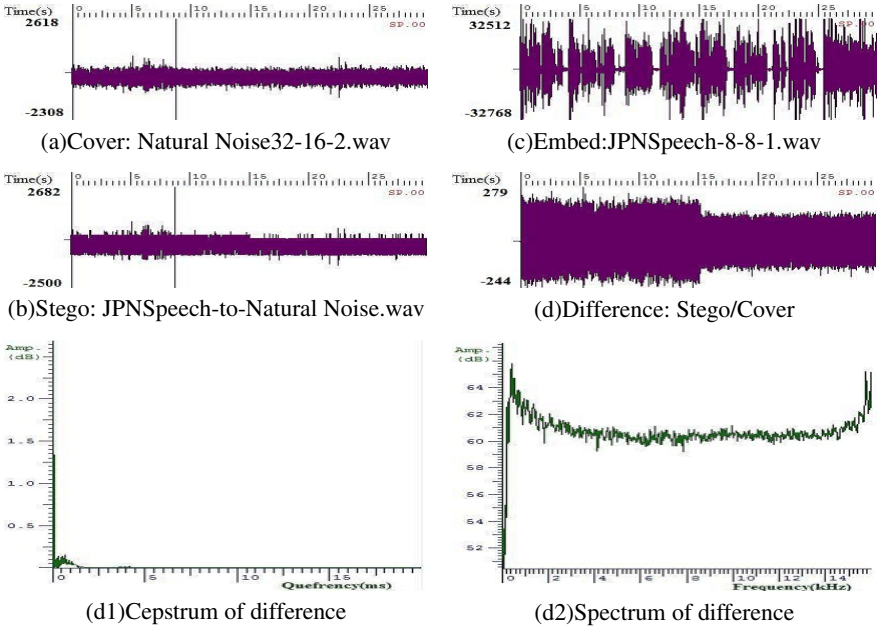


Fig. 9.15. Group 4: Embed Speech into Weighted Noise

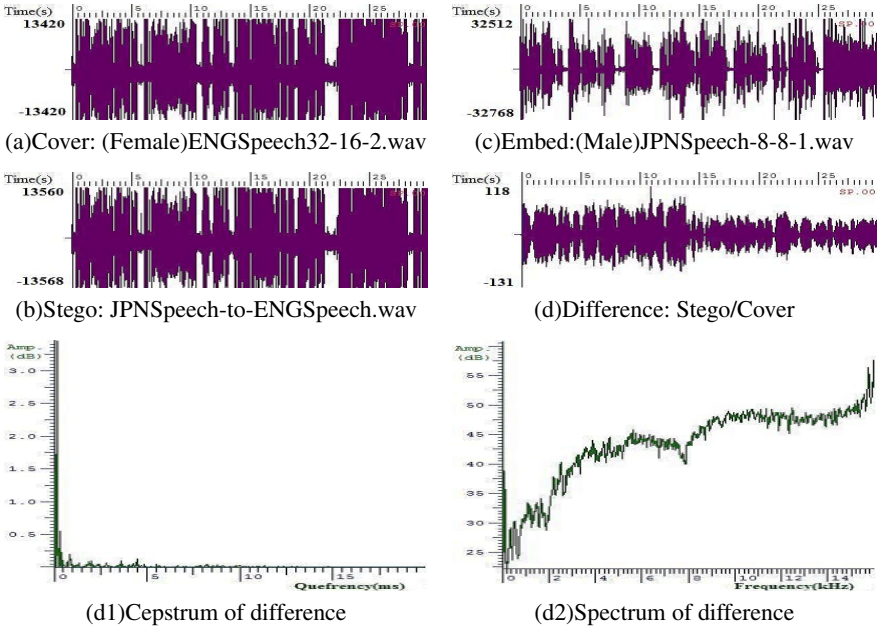


Fig. 9.16. Group 5: Embed Speech into Speech

## References

1. Johnson, N.F., Duric, Z., Jajodia, S.: *Information Hiding Steganography and Watermarking-Attacks and Countermeasures*. Kluwer Academic Publishers (2001)
2. Segawa, N., Murayama, Y., Miyazaki, Y.M.: Information hiding with a handwritten message in vector-drawing codes. In: Proc. 35th Hawaii Int'l Conf. System Sciences, pp. 4027–4033 (2002)
3. Matsuoka, H.: Acoustic data hiding method using sub-band phase shifting. Technical report of IEICE EA 106, 7–11 (2006)
4. Pan, J.S., Huang, H.C., Jain, L.C.: *Intelligent Watermarking Techniques*. World Scientific Publishing Company, Singapore (2004)
5. Pan, J.S., Huang, H.C., Jain, L.C., Fang, W.C.: *Intelligent Multimedia Data Hiding-New Directions*. Springer, Heidelberg (2007)
6. Cao, G., Zhao, Y., Ni, R.R.: Edge-based blur metric for tamper detection. *Journal of Information Hiding and Multimedia Signal Processing* 1, 20–27 (2010)
7. Wang, H., Liang, J., Kuo, C.C.J.: Overview of robust video streaming with network coding. *Journal of Information Hiding and Multimedia Signal Processing* 1, 36–50 (2010)
8. Huang, H.C., Chen, Y.H., Abraham, A.: Optimized watermarking using swarm-based bacterial foraging. *Journal of Information Hiding and Multimedia Signal Processing* 1, 51–58 (2010)
9. Huang, X.P., Kawashima, R., Segawa, N., Abe, Y.: The real-time steganography based on audio-to-audio data bit stream. Technical report of IEICE 106, 15–22 (2006)
10. Huang, X.P., Kawashima, R., Segawa, N., Abe, Y.: Design and implementation of synchronized audio-to-audio steganography scheme. In: Proc. Int'l Conf. Intelligent Information Hiding and Multimedia Signal Processing, pp. 331–334 (2008)
11. Wu, M., Liu, B.: *Multimedia Data Hiding*. Springer, New York (2003)
12. Katzenbeisser, S., Petitcolas, F.A. (eds.): *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House London (2000)
13. Chakrabarty, S., Deng, Y., Cauwenberghs, G.: Robust speech feature extraction by growth transformation in reproducing kernel hilbert space. *IEEE Trans. Audio, Speech, and Language Processing* 15, 473–480 (2007)
14. Boney, L., Tewfik, A.H., Hamdy, K.H.: Digital watermarks for audio signals. In: Proc. Int'l Conf. Multimedia Computing and Systems, pp. 473–480 (1996)
15. Ito, A., Makino, S.: Designing Side Information of Multiple Description Coding. *Journal of Information Hiding and Multimedia Signal Processing* 1, 10–19 (2010)
16. Kusatsu, I., Niimi, M., Noda, H., Kawaguchi, E.: A large capacity steganography using acoustic dummy data. Technical Report of IEICE EA98-69-78, 27–32 (1998)
17. Pal, S.K., Saxena, P.K., Muttoo, S.K.: The future of audio steganography. In: Proc. Pacific Rim Workshop on Digital Steganography, pp. 136–145 (2002)
18. Cox, I., Miller, M.L., Bloom, J.A.: *Digital Watermarking*. Morgan Kaufmann Pub. (2002)
19. Wang, S., Yang, B., Niu, X.: A secure steganography method based on genetic algorithm. *Journal of Information Hiding and Multimedia Signal Processing* 1, 28–35 (2010)



20. Wang, Z.H., Kieu, T.D., Chang, C.C., Li, M.C.: A novel information concealing method based on exploiting modification direction. *Journal of Information Hiding and Multimedia Signal Processing* 1, 1–9 (2010)
21. Stinson, D.R.: *Cryptography-Theory and Practice*. CRC Press (1995)
22. Oliver, G.: Understanding and managing delay in the global voice network. PMC-Sierra Webinar (2003), <http://www.pmc-sierra.com/webinars/dec2003.html>
23. Goto, M., Hashiguchi, H., Nishimura, T., Oka, R.: RWC music database: Music genre database and musical instrument sound database. In: *Proc. 4th Int'l. Conf. Music Information Retrieval*, pp. 229–230 (2003)
24. Kobayasi, T., Itahashi, S., Hayamizu, S., Takezawa, T.: ASJ continuous speech corpus for research. *The Journal of the Acoustical Society of Japan* 48, 888–893 (1992)

---

## Video Watermarking with Shot Detection

Yueh-Hong Chen<sup>1</sup> and Hsiang-Cheh Huang<sup>2</sup>

<sup>1</sup> Department of Computer Science and Information Engineering,  
Far East University,  
Tainan 744, Taiwan, R.O.C.  
yuehhong@gmail.com

<sup>2</sup> Department of Electrical Engineering,  
National University of Kaohsiung,  
Kaohsiung 811, Taiwan, R.O.C.  
huang.hc@gmail.com

<https://sites.google.com/site/hch888dr/>

**Summary.** This chapter presents a method to embed a watermark in a video for recording copyright information to prevent the video from being pirated. Firstly, a video is cut into small shots, and the shots are classified with clustering algorithm in accordance with their similarity. Then, the DWT (Discrete Wavelet Transform) is applied to transform a watermark into wavelet domain, and embed the wavelet coefficients in appropriate shots according to the level of its coefficients. Up to this step, the procedure of watermark embedding is completed. On the other hand, while extracting watermark, it is also necessary to cut a video into shots and extract the frame from each shot to extract the wavelet coefficient embedded. The last step for extracting watermark is collocating all parameters to extract the original watermark by using IDWT (Inverse DWT). With the method proposed, most portion of watermark could be extracted even when only partial video shots are available. It is able to obtain not only higher NC (normalized cross-correlation) value, but also better visual quality compared to directly embedding partial watermark data into shots without adopting DWT.

### 10.1 Introduction

Digital watermarking is one of the hot research topics in the multimedia area. By using this technology, one can embed copyright information in digital content to prevent some possible piracy attempts. Watermarks that are about to be embedded need to meet few qualifications, such as (a) imperceptibility, (b) security, and (c) robustness to common image processing methods [1].

Currently, many image watermarking methods have been proposed [2]–[10]; however those watermarking methods generally applied to image are somewhat ineffective on videos. In addition to conventional watermarking techniques, video watermarking must be able to handle problems due to (1) massive frames, (2) dependency of neighboring frames, and (3) video-based attacking, such as frame dropping, frame averaging, collusion, etc. [11]

In order to conquer the difficulties above, some methods based on videos scene or shot have been proposed respectively [12, 13, 14]. Zhu, Swanson and Tewfik [12] embedded a watermark in each scene, and enhanced the watermark robustness with the characteristics of human visual system (HVS). Method [13] proposed by Swanson et al. suggested that one should save the watermark correspondent to certain objects, and then detect objects contained in a video to embed a watermark in each object. While Koz and Alaton [14] applied temporal FFT (Fast Fourier Transform) to decide temporal sensitivities of human visual for watermark embedding.

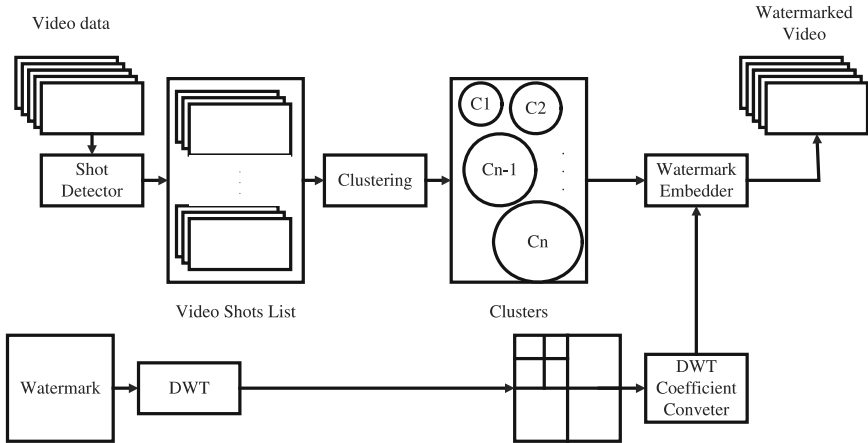
Generally, those methods only process individual fundamental units in videos, like scene, shot, object, etc. [15] If a watermark is repetitively embedded in the fundamental units, say objects, it is easy to verify the copyright holder when some frames of a video are pirated. Nevertheless, the watermark embedded is also highly susceptible to averaging attacks.

To prevent a watermark from being susceptible, we propose a novel method to separate a watermark into numerous parts with Discrete Wavelet Transform (DWT). Then, cluster video shots together and embed the more important (low frequency) part of the watermark in each frame of bigger clusters that possess more shots. The method differs from others by not embedding the whole watermark in individual fundamental units. The main advantage of it is that the quality of the extracted watermark is raised since it makes low frequency coefficients appear in more shots. Moreover, most portion of the watermark information can be extracted even when only partial watermark coefficients are available; meanwhile, the embedded watermarks can hardly be filtered out by frame averaging attacks. The experimental results presented in Section 10.4 shows that the proposed method possesses the advantage to resist most attacks. After color noise, MPEG coding, frame averaging, and shot dropping attacks are applied to the watermarked video, the extracted watermark can still reaches high qualities.

The rest of this chapter is organized as follows. In Section 10.2, the embedding algorithm is illustrated in detail, while in Section 10.3, the detecting and extracting algorithm are introduced. In Section 10.4, experimental data sets and results are presented and discussed. Finally, Section 10.5 briefs some concluding remarks.

## 10.2 Embedding Algorithm

The digital watermarking scheme proposed here includes watermark embedding and extraction. Embedding algorithm would be discussed in this section, and the flow of embedding algorithm is shown as Fig. 10.1. First, shot detecting algorithm is applied to analyze the video in which is about to be embedded a watermark, and from which to obtain a shot list. Then, all shots are classified to form clusters of shots in accordance with their similarity. Third, adopt DWT to transform the watermark into wavelet domain, and embed low frequency coefficients in bigger clusters, which contain more shots than others. At last, complete the operation of embedding partial coefficients in a shot by embedding coefficients in all frames of the shot with



**Fig. 10.1.** Diagram of embedding algorithm.

watermark embedder. Detailed procedure and analysis and discussion of each part is illustrated respectively in following sections.

### 10.2.1 Shot Detector

Many methods have been proposed to [16, 17, 18, 19] analyze video shots. In the proposed method here, an algorithm that combines [16] and [20] is applied to obtain shots of a video about to be embedded a watermark. First, the method proposed in [16] is applied to obtain the spatial-temporal graph from the video, and then detect where edges are in the spatial-temporal graph with Sobel masking [20]. The edges here are the boundaries between two shots.

### 10.2.2 Clustering

So far, numerous of clustering algorithms have been proposed [21, 22]. The one we adopt in this chapter is k-means clustering [21], which is the easiest and most effective among all [22]. Surely, this is not to imply that other clustering algorithms are not applicable. We set the parameter  $k$  in k-means clustering to be  $n$ .

### 10.2.3 DWT and DWT Coefficient Converter

Most portion of a watermark can be extracted even when only low frequency coefficients are available. This is due to the fact that wavelet information collocates in low frequency areas after a watermark being transformed into wavelet domain. Consequently, the DWT proposed by Haar [20] is adopted to transform the representation of the watermark into wavelet domain, which is qualified to be embedded. The result of transforming Fig. 10.2(a) into wavelet domain is shown as Fig. 10.2(b).

交大資工  
NNLAB

(a)



(b)

**Fig. 10.2.** (a) An example of watermark and (b) the result of applying DWT on this watermark.

After transformation, more important coefficients (lower frequency) and less important ones (higher frequency) are able to be embedded in appropriate shots. It is quite obvious that the possibility to extract low frequency coefficients from shots is much higher than that to extract high frequency ones if low frequency coefficients are embedded more times than high frequency ones are. The scheme effectively leads to enhancement of a watermark's robustness.

Once a watermark is transformed by DWT, the coefficients of each level differs from each other dramatically. To make up the difference, DWT coefficient converter is applied to normalize coefficients of each level after transformation, and the method to normalize coefficients is shown below:

Suppose a set of coefficients of the same level is  $W_m = \{w_1, w_2, \dots, w_k\}$ ,  $1 \leq m \leq n$ . DWT coefficient converter is normalized to DWT coefficient  $W'_m = \{w'_i\}$  according to Eq. (10.1).

$$w'_i = T_m \cdot \left( \frac{w_i - \min W_m}{\max W_m - \min W_m} \right), \quad 1 \leq i \leq k. \quad (10.1)$$

In Eq. (10.1) the value of  $T_m$  is assigned according to the level of  $w_i$ ,  $\max W_m$  and  $\min W_m$  are the maximum and minimum values for  $W_m$ , and  $w_i$ ,  $1 \leq i \leq k$  is the actual value embedded in each frame.

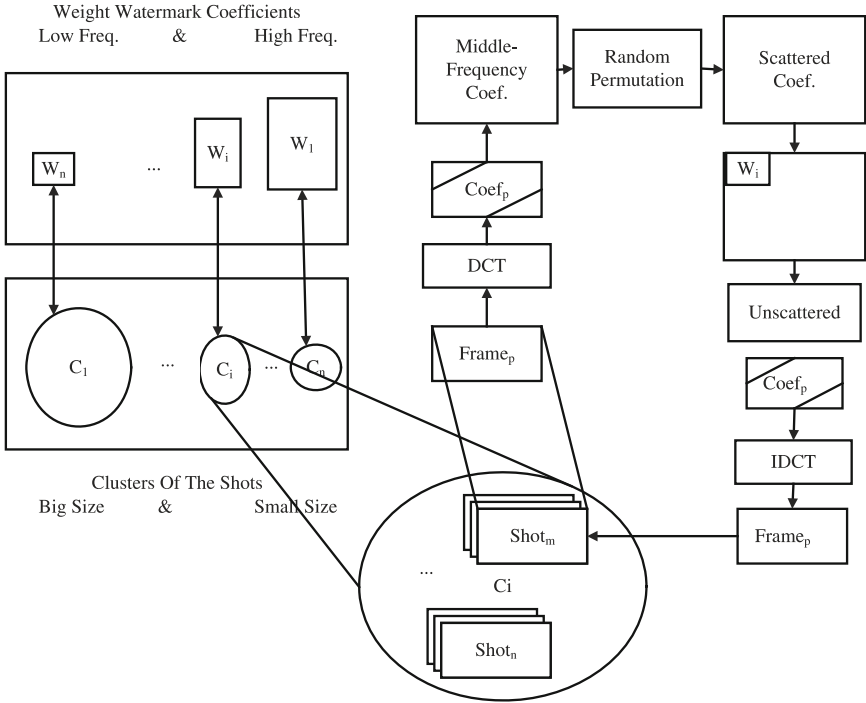


Fig. 10.3. Diagram of watermark embedder.

Because of setting  $n$  to parameter  $k$  in  $k$ -means clustering, we also set level count to  $n$  in DWT, that is, we have  $n$  level coefficients need to be embedded.

### 10.2.4 Watermark Embedder

In this step, the coefficient  $W'_m$  gained by DWT coefficient converter is embedded in each frame of all shots from correspondent cluster, which is shown as Fig. 10.3. Suppose that cluster of shot, in which the information is about to be embedded is,  $C_m, 1 \leq m \leq n$ . Then, randomly embed  $W_m, 1 \leq m \leq n$ , the result of DWT coefficient converter in medium frequency of frames, and make the embedded information remain at the same position in frames of each shot of  $C_m$ . Eq. (10.2) shows how to embed information  $b \in \{0, 1\}$  in a certain medium frequency coefficient  $c$  [10]:

$$c' = \begin{cases} (\lfloor \frac{c}{step} \rfloor + 1) \times step, & \text{if } \lfloor \frac{c}{step} \rfloor \bmod 2 \neq b; \\ c, & \text{otherwise.} \end{cases} \quad (10.2)$$

In Eq. (10.2),  $step$  is a strength factor. Lastly, we replace the original coefficient  $c$  with  $c'$ , and operate IDWT on revised frame to complete the watermark embedding. Please note that the result of  $w'_i, 1 \leq i \leq k$  gained by DWT coefficient converter should be regarded as a string of 0 and 1 in order to embed  $w'_i$  in each frame with the method.

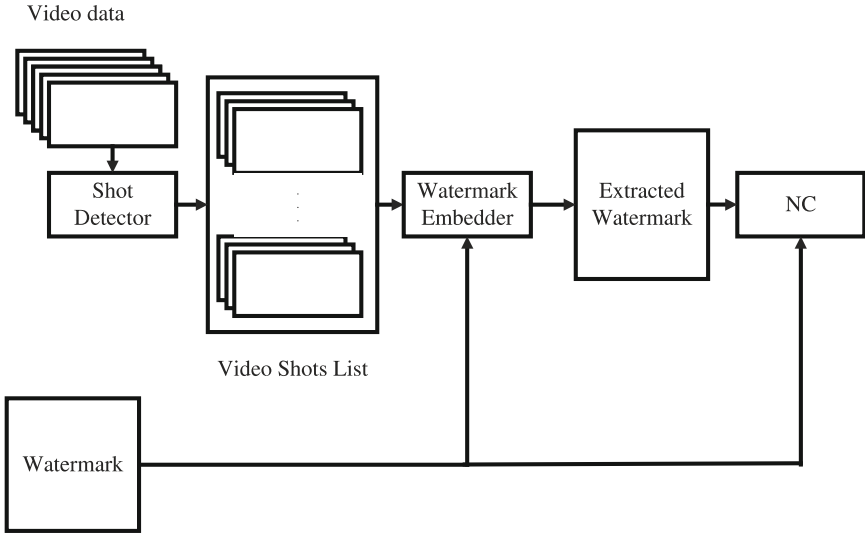


Fig. 10.4. Diagram of extracting algorithm.

### 10.3 Extracting Algorithm

The flow of the extracting algorithm is shown as Fig. 10.4. First, obtain shots of test video with shot detection algorithm, and extract watermarks from shots with watermark extractor. Then, compare the extracted watermark with the original one with normalized cross-correlation (NC) [2] measure. The flow of watermark extractor is shown as Fig. 10.5. The original watermark information is transformed into normalized DWT coefficients. When trying to extract watermark, one can randomly draw a frame from each shot since all frames of a shot have been embedded the same coefficient. Since the embedding algorithm randomly embeds the result gained by DWT coefficient converter in medium frequency coefficients of frames and reserves their seeds, the extracting algorithm can try to extract information according to the order which they are embedded earlier. While extracting embedded information from a DCT coefficient,  $c$ , one can see if the embedded information in  $c$  is 0 or 1 from  $\lfloor \frac{c}{step} \rfloor \bmod 2$ . At last, test the correlation between the extracted information and the normalized DWT coefficient  $W_m$ ,  $1 \leq m \leq n$ . If  $c$  value of correlation is larger than the pre-described threshold, it is assumed that the set of coefficients in the frame is  $W_m$ . If not, pick another frame from the shot to detect again until all frames are detected. If there is no desired coefficients detected, it is assumed that the shot has no watermark embedded. Finally, collocate all coefficients gained and transform them into pixel domain with IDWT to obtain the extracted watermark.

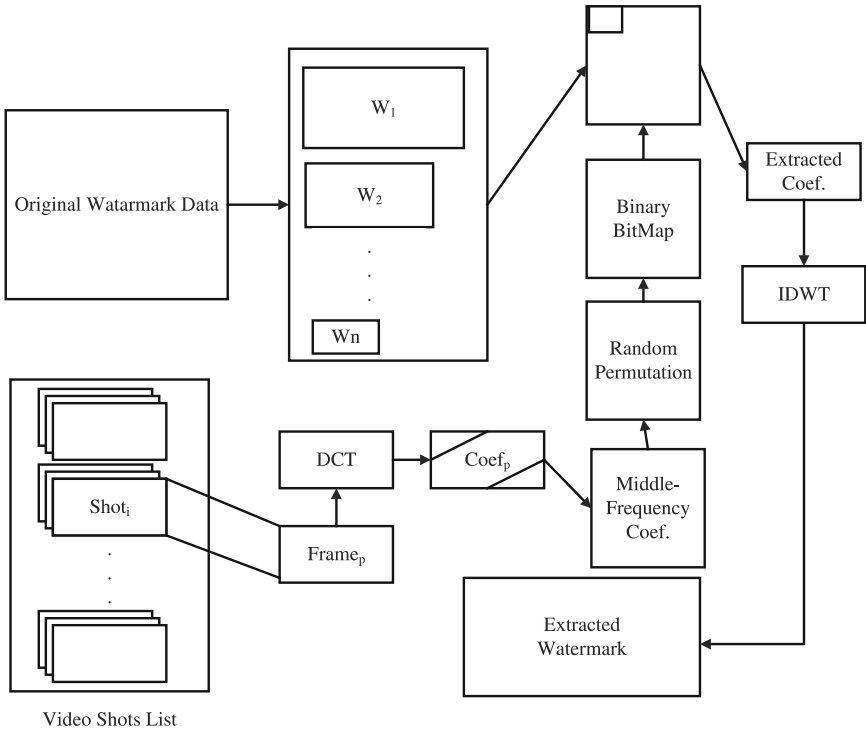


Fig. 10.5. Diagram of watermark extractor.

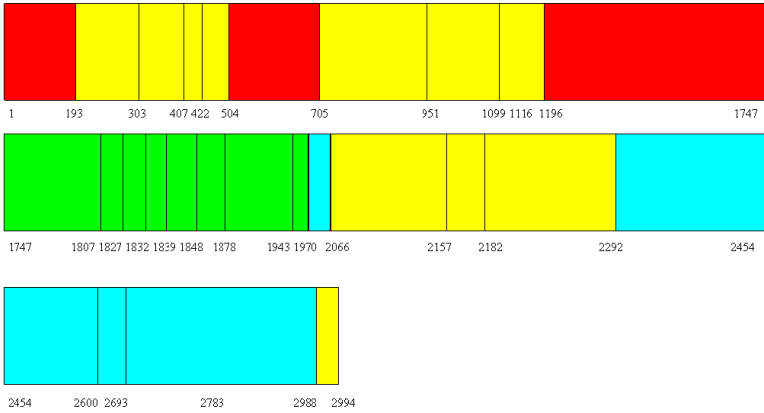
## 10.4 Experimental Results

In this section, some experimental results are presented show to present robustness and invisibility of the watermark scheme proposed here. The video used in the experiment is a clip of grayscale (8 bpp) video played on CTS Evening News on December 10th, 2003. The 1'39"-long video clip contains 2994 frames, each of which is  $352 \times 240$  pixel, and the parameter  $k$  in k-means clustering algorithm is 4. Fig. 10.2(a) is the grayscale watermark being embedded in the video, which is  $64 \times 64$  pixel, and the parameter  $step$  is 3. Fig. 10.6 shows the distribution of shots and the clusters they belong to. Table 10.1 presents the number of shots belonging to each cluster.

### 10.4.1 Visual Quality

Let Fig. 10.7(a) be the original frame, which is frame number is 195 in the news sequence. Fig. 10.7(b) is the corresponding watermarked frame in the video sequence, in which the most information is embedded. Fig. 10.8(a) and (b) are another frame selected from the original and the watermarked video, in which the least information is embedded. Compare the frame of the original video with that of the watermarked





**Fig. 10.6.** The shot structure of the video, in which shots belonging to the same cluster have the same color.

**Table 10.1.** Number of shots belonging to each cluster.

| Clusters (represented in colors) | # of shots |
|----------------------------------|------------|
| red                              | 3          |
| yellow                           | 12         |
| green                            | 8          |
| blue                             | 6          |
| total                            | 29         |

video with PSNR (Peak Signal-to-Noise Ratio). Results shows that both PSNR values in Fig. 10.7 and Fig. 10.8 are relatively high, which implies that the watermark embedded in a video possesses high invisibility.

### 10.4.2 Robustness

To be effective, the watermark must be robust to incidental and intentional signal distortions incurred by the video. Clearly, any lossy signal operations performed in the video effect the embedded watermark [12].

We present the experimental results after the watermarked video being attacked by color noise, MPEG coding, frame averaging, and shot dropping in the following paragraphs. By comparing the correlation of the original watermark with the extracted watermark with NC, the result shows how serious the embedded watermark is damaged.

1. The first experiment tests the effect brought by color noise attack performed in videos. To attack by using color noise, it is necessary to embed the white noise



(a) A frame of the original video whose frame number is 195.



(b) A frame of the watermarked video. (PSNR = 27.90 dB)

**Fig. 10.7.** The first result of experiment in Sec. [10.4.1](#)

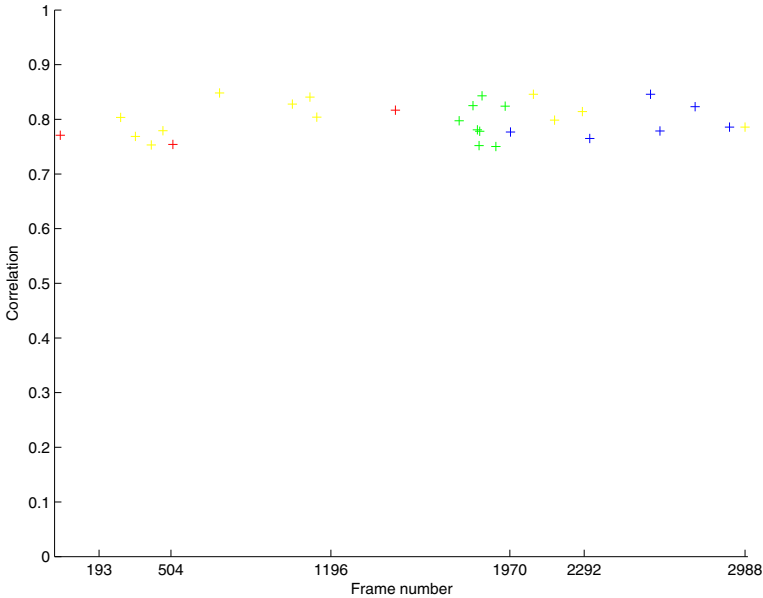


(a) A frame of the original video whose frame number is 1201.



(b) A frame of the watermarked video. (PSNR = 39.40 dB)

Fig. 10.8. The second result of experiment in Sec. 10.4.1



(a)



(b) The extracted watermark. (NC=0.9645)

**Fig. 10.9.** Results of the first experiment for color noise attack in Sec. 10.4.2

in the video to be damaged [12]. The result is shown as Fig. 10.9(a). The abscissa represents the frame number while the ordinate, the highest correlation between the information obtained from each shot of the video and the normalized DWT coefficients. Colors distinguish the different levels of coefficients. The frame number tells the position of a frame in the shot to which we detect the coefficient in, finally. Fig. 10.9(b) is the watermark extracted.

From Fig. 10.9, the value of correlation between information obtained from each shot and its correspondent normalized DWT coefficient appears rather high. The result indicates that the color noise attack does not affect the watermark embedded in the video. Moreover, the NC values of the original

**Table 10.2.** Number of frames belonging to each shot that are needed to be detected.

| Shot (from frame number $i$ to $j$ ,<br>$\forall i, j \in \{1, \dots, 2994\}$ ) | Number of needed<br>to be detected frames |
|---|---|
| 2292–2554   | 10 (max)                                  |
| 1196–1747   | 3 (min)                                   |

watermark and the extracted one are both high, which helps us to assume without difficulty that a watermark is embedded in the video.

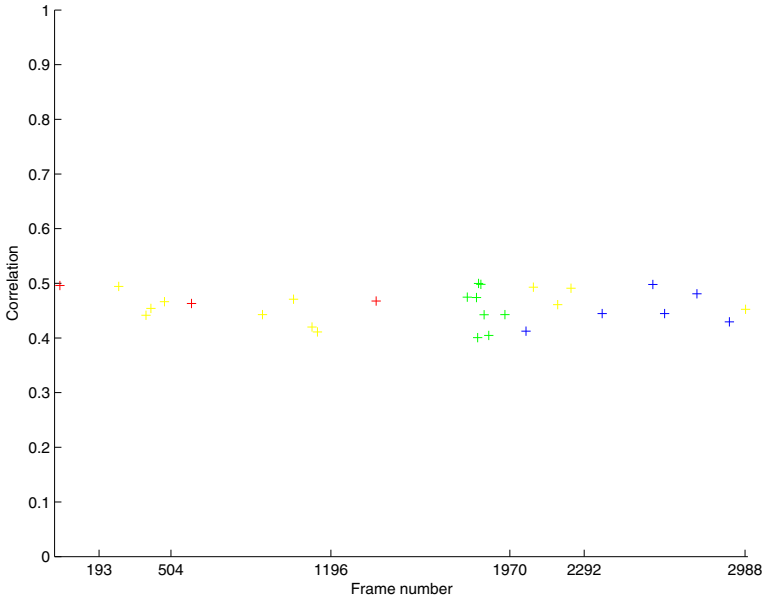
2. The second experiment is to observe the damage on a watermark done by MPEG coding. MPEG increases its compression ratio by dropping high frequency information of DCT. Once the compression ratio increases, the watermark information embedded can possibly be dropped by the compression process [12]. Fig. 10.10(a) shows the result when the compression rate of the video with a watermark embedded increases 1.5 times. In this way, the value of correlation between information collocated from each shot and its correspondent normalized DWT coefficient drops dramatically. Even the extracted watermark from the video becomes vague accordingly, and NC values of the original and the extracted watermarks decrease as well. Regardless the result, the value of correlation between information collocated from each shot and its correspondent normalized DWT coefficient still lie between 0.4 to 0.6. In addition to that, the frame number needed to be detected to identify a shot's correspondent normalized DWT coefficient rises slightly. As shown on Table 10.2, the maximum frame number to be detected is 10 whereas the minimum, 3.

3. The third experiment presents how serious the frame averaging damages the watermark embedded. The frame averaging is quite similar to the frame dropping in that both attacks try to fix the video being damaged, which results in our incapability to extract a watermark from the video [12]. In this experiment, We replace the even index frames, i.e., 2, 4,  $\dots$ , with the average of the two neighboring frames,

$$F_{2n+1} = \frac{1}{2}(F_{2n} + F_{2n+2}).$$

From Fig. 10.11(a), one can see that the digital watermark scheme possesses high resistance to frame averaging.

4. The fourth experiment tests the effect brought by the proposed scheme to extract watermarks when only partial video information is available. Make the video that has a watermark embedded to (a) drop all shots in the red clusters on Fig. 10.6, (b) drop all shots in the blue and red clusters on Fig. 10.6, and (c)



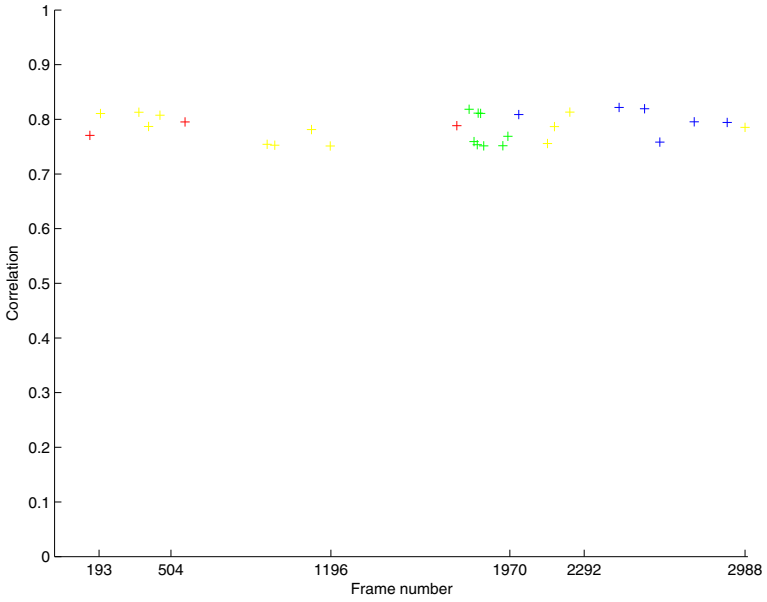
(a)



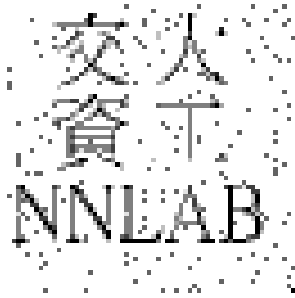
(b) The extracted watermark. (NC=0.851271)

**Fig. 10.10.** Results of the second experiment for compression attack with different compression ratios in MPEG in Sec. [10.4.2](#)

only retain the shots in the yellow clusters on Fig. [10.6](#). Then, extract the watermark with watermark extracting algorithm proposed in the chapter. Results are shown as Fig. [10.12](#). Obviously, the more video shots are obtained, the clearer the watermark extracted becomes. The shots in bigger clusters are more readily to be pirated. After the watermark being transformed into wavelet domain by DWT and then embedded in frames, the lower frequency of the coefficients, the higher opportunity for the shots containing the coefficients to be available. As a result, the watermark extracted can reach extremely high quality.

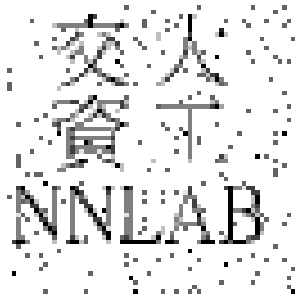


(a)



(b) The extracted watermark. (NC=0.93471)

**Fig. 10.11.** Results of the third experiment for frame averaging attack in Sec. [10.4.2](#)



(a) The extracted watermark extracted from the video without all shots in the red cluster on Fig. [10.6](#)



(b) The extracted watermark extracted from the video without all shots in the blue and red clusters on Fig. [10.6](#)



(c) The extracted watermark extracted from the only containing the shots in the yellow clusters on Fig. [10.6](#)

**Fig. 10.12.** Results of the fourth experiment for shot dropping attack in Sec. [10.4.2](#)



## 10.5 Conclusion

In this chapter, we propose a novel video watermarking scheme. First, the shot detection algorithm is applied to the original video. Then, all video shots are clustered in according to their similarity with clustering algorithm to form clusters of shots. Third, apply DWT on the watermark that is about to be embedded in a video and separate its coefficients according to the frequency. The shots in a cluster are so similar that allows us to embed the same information to enhance their robustness. To improve the quality of extracted watermark, we embed low frequency DWT coefficients in shots of the bigger shot clusters, and high frequency ones in those of the smaller shot clusters. Experimental results show that the superior quality of the video frames with watermarks embedded remains. Watermarks can still be extracted successfully after being attacked by color noise, MPEG coding, frame averaging, and shot dropping.

## References

1. Cox, I.L., Miller, M.L., Bloom, J.A., Fridrich, J., Kalker, T.: *Digital Watermarking and Steganography*. Elsevier Science & Technology, London (2007)
2. Hsu, C.T., Wu, J.L.: Hidden digital watermarks in images. *IEEE Transactions on Image Processing* 8, 58–68 (1999)
3. Zeng, W., Liu, B.: On resolving rightful ownerships for digital images by invisible watermarks. In: *Proc. Int'l Conf. on Image Processing*, pp. 552–555 (1997)
4. Inoue, H., Miyazaki, A., Yamamoto, A., Katsura, T.: A digital watermark technique based on the wavelet transform and its robustness on image compression and transformation. *IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences E82-A*, 2–10 (1999)
5. Huang, H.C., Chen, Y.H., Abraham, A.: Optimized watermarking using swarm-based bacterial foraging. *Journal of Information Hiding and Multimedia Signal Processing* 1, 51–58 (2010)
6. Huang, H.C., Fang, W.C.: Metadata-based image watermarking for copyright protection. *Simulation Modelling Practice and Theory* 18, 436–445 (2010)
7. Huang, H.C., Chen, Y.H.: Genetic fingerprinting for copyright protection of multicast media. *Soft Computing* 13, 383–391 (2009)
8. Huang, H.C., Chang, F.C., Fang, W.C.: Reversible data hiding with histogram-based difference expansion for QR code applications. *IEEE Trans. Consumer Electronics* 57, 779–787 (2011)
9. Huang, H.C., Chu, C.M., Pan, J.S.: The optimized copyright protection system with genetic watermarking. *Soft Computing* 13, 333–343 (2009)
10. Kii, H., Onishi, J., Ozawa, S.: The digital watermarking method by using oth patchwork and DCT. In: *Proc. Int'l Conf. on Multimedia Computing and Systems*, pp. 895–899 (1999)
11. Ko, C.C., Yang, B.Z.: An Integrated Technique for Video Watermarking. In: *Proc. Int'l Cong. on Computer and Information Science*, pp. 37–42 (2007)
12. Zhu, B., Swanson, M.D., Tewfik, A.H.: Multiresolution scene-based video watermarking using perceptual models. *IEEE Journal on Selected Areas in Communications* 16, 540–550 (1998)

13. Swanson, M.D., Zhu, B., Chau, B., Tewfik, A.H.: Object-based transparent video watermarking. In: Proc. IEEE First Workshop on Multimedia Signal Processing, pp. 369–374 (1997)
14. Koz, A., Alatan, A.A.: Video watermarking using temporal sensitivities of human visual system. In: Proc. Workshop on Transmitting, Processing and watermarking Multimedia Contents (2003), <http://www.eee.metu.edu.tr/~alatan/PAPER/COST276.Workshop.03.B.pdf>
15. Cotsaces, C., Nikolaidis, N., Pitas, I.: Video shot boundary detection and condensed representation: A review. *IEEE Signal Processing Magazine* 23, 28–37 (2006)
16. Tang, X., Gao, X., Liu, J., Zhang, H.J.: A Spatial-Temporal Approach for Video Caption Detection and Recognition. *IEEE Transactions on Neural Networks* 13, 961–971 (2002)
17. Cernkova, Z., Kotropoulos, C., Pitas, I.: Video Shot Segmentation Using Singular Value Decomposition. In: Proc. Int'l. Conf. on Multimedia and Expo, pp. 301–304 (2003)
18. Dugad, R., Ratakonda, K., Ahuja, N.: Robust video shot change detection. In: Proc. IEEE Second Workshop on Multimedia Signal Processing, pp. 376–381 (1998)
19. Lin, T., Zhang, H.J.: Automatic video scene extraction by shot grouping. In: Proc. Int'l Conf. on Pattern Recognition, pp. 39–42 (2000)
20. Gonzalez, R.C., Woods, R.E.: *Digital Image Processing*. Prentice-Hall, Englewood Cliffs (2008)
21. Alsabti, K., Ranka, S., Singh, V.: An Efficient K-Means Clustering Algorithm. In: Proc. IPPS/SPDP Workshop on High Performance Data Mining (1999)
22. Hastie, T., Tibshirani, R., Friedman, J.: *The Elements of Statistical Learning*. Springer, Basel (2001)

---

# Subject Index

- Android, [19](#) [20](#) [27](#) [31](#)
  - Android 2.3.3, [27](#)
- attack, [103](#)
- audio, [139](#) [159](#) [182](#)
- basis vector, [56](#)
- bit correct rate (BCR), [140](#) [141](#)
- bit correct ratio (BCR), [46](#) [47](#)
- block truncation coding (BTC), [119](#)-[121](#)  
[123](#) [132](#) [137](#)
- capacity, [1](#) [9](#) [33](#)-[37](#) [41](#)-[43](#) [48](#)-[50](#) [103](#)  
[105](#) [106](#) [114](#) [141](#) [155](#) [157](#) [182](#) [195](#)
- Cat Swarm Optimization (CSO), [119](#) [121](#)  
[123](#) [125](#) [126](#) [128](#) [129](#) [137](#)
- codebook, [1](#) [2](#) [4](#)-[6](#) [9](#) [33](#) [37](#)-[39](#) [42](#) [43](#)  
[48](#) [50](#)
- codeword, [1](#) [2](#) [4](#)-[6](#) [8](#) [33](#) [38](#) [39](#) [42](#) [160](#)
- coefficient adjustment (CA), [33](#) [37](#)-[46](#)  
[48](#)-[50](#)
- compression, [1](#) [31](#) [36](#) [61](#) [121](#) [155](#) [158](#)  
[159](#) [183](#)
- constrained optimization problem, [141](#)
- copyright protection, [139](#)
- correlation, [105](#)
- covariance matrix, [54](#)
- data hiding, [4](#) [34](#) [159](#)
- difference expansion, [34](#) [157](#)
- digital rights management (DRM), [20](#)
- discrete cosine transform, [59](#) [66](#) [69](#)-[71](#) [89](#)
  - inverse DCT, [66](#)
- discrete cosine transform (DCT), [21](#) [140](#)  
[142](#) [182](#) [216](#) [222](#)
- discrete wavelet transform (DWT), [57](#) [59](#)  
[78](#) [139](#) [140](#) [142](#) [211](#)-[216](#) [221](#)
  - inverse DWT, [57](#) [211](#) [215](#) [216](#)
- distortion, [34](#) [156](#) [182](#)
- downsampling, [55](#) [57](#) [69](#) [71](#)
- eigenvalue, [54](#) [62](#)
- eigenvector, [54](#) [55](#)
- error function, [67](#)
- Euclidian distance, [3](#)
- exclusive or,  $\oplus$ , [74](#) [80](#) [141](#)
- feature extraction, [55](#) [60](#) [104](#) [114](#)
- FFT (Fast Fourier Transform), [212](#)
- fidelity, [140](#)
- fitness function, [143](#) [145](#) [147](#) [150](#)-[152](#)
- Fourier-Mellin transform, [104](#)
- frequency domain, [108](#) [140](#) [142](#) [212](#) [223](#)
- Gaussian noise, [68](#) [73](#) [85](#) [91](#) [92](#) [95](#)
- genetic algorithm, [139](#) [141](#)-[148](#) [150](#) [151](#)
  - crossover, [144](#)
  - mutation, [144](#) [147](#)
  - selection, [144](#) [147](#)
- graph coloring, [1](#)-[3](#) [8](#) [16](#)
- Haar wavelet, [23](#)
- histogram, [34](#) [35](#) [157](#)
- Huffman coding, [155](#) [159](#) [160](#) [165](#)
- human auditory system (HAS), [183](#)
- human visual system (HVS), [36](#) [53](#) [69](#)  
[183](#) [212](#)

- image, [103](#), [139](#), [140](#), [155](#), [159](#), [182](#)
- imperceptibility, [1](#), [104](#), [111](#), [182](#)
- independent component analysis, [53-57](#), [60](#), [61](#), [64](#), [67](#), [69](#), [71](#), [80](#), [86](#), [94](#)
- FastICA, [53-56](#), [78](#), [80](#)
- ICA transform, [57](#), [59](#)
- independent component, [60](#)
- inverse ICA transform, [59](#)
- PICA, [55](#), [62](#)
- PICF, [62](#)
- information hiding, [155](#)
- integer wavelet transform (IWT), [33](#), [41](#), [42](#), [45](#), [46](#), [48](#)
- interpolation, [36](#), [107](#), [184](#)
- invisibility, *see* imperceptibility, *see* imperceptibility
- ITU-T Y.1541 Standard, [201](#)
- Java, [20](#)
- JPEG, [21](#), [27](#), [31](#), [36](#), [65](#), [71](#), [72](#), [141](#), [160](#)
- quality factor, [31](#)
- JPEG2000, [36](#)
- kurtosis, [54](#)
- least significant bit, [194](#)
- least significant bit (LSB), [182](#), [184](#), [187](#), [192](#), [198](#), [201](#)
- Lempel-Ziv-Welch (LZW) method, [159](#), [164](#)
- linear combination, [53](#), [54](#)
- Linux, [20](#)
- locally adaptive data compression (LADC), [159](#)
- location map, [157](#)
- lossless compression method (LDAC), [161](#)
- mean opinion score (MOS), [195](#), [196](#)
- mean square error (MSE), [24](#), [137](#)
- metadata, [182](#)
- Microsoft Word, [158](#)
- mixture ratio (MR), [122](#), [126](#), [128](#)
- mobile phone, [183](#)
- motion compensation, [61](#)
- MPEG, [73](#), [76](#), [212](#), [222](#), [226](#)
- MPEG-2, [75](#), [81](#), [85](#), [92](#), [93](#), [95](#)
- Non-sampled Contourlet Transform (NSCT), [107](#), [108](#), [114](#)
- normalized cross-correlation (NC), [67](#), [71](#), [73](#), [90-93](#), [97](#), [98](#), [211](#), [216](#), [218](#), [221](#), [222](#)
- particle swarm optimization (PSO), [2-5](#), [8](#), [9](#), [16](#)
- pattern recognition, [104](#)
- payload, *see* capacity, *see* capacity, *see* capacity
- PCM, [181](#), [184](#), [189](#), [193-195](#)
- Peak Signal-to-Noise Ratio (PSNR), [140](#), [145](#), [147](#), [156](#)
- peak signal-to-noise ratio (PSNR), [9](#), [24](#), [34](#), [36](#), [41-45](#), [48](#), [49](#), [65](#), [80](#), [91](#), [111](#), [113](#), [126](#), [137](#), [218](#)
- prediction-error expansion, [35](#)
- principal component analysis (PCA), [54](#), [55](#), [62](#), [121](#)
- quality, [2](#), [31](#), [80](#), [107](#), [111](#), [195](#), [212](#), [223](#)
- quantization, [182](#), [195](#), [196](#)
- quantization index modulation, [70](#), [87-89](#)
- real-time, [183](#), [185](#), [186](#), [190](#), [201](#)
- reversible data hiding, [34](#), [36](#), [46](#), [50](#), [157](#)
- robustness, [21](#), [27](#), [46](#), [104](#), [106](#), [107](#), [109](#), [113](#), [114](#), [140](#), [184](#)
- shot detection, [211](#), [212](#), [216](#), [226](#)
- signal-to-noise ratio (SNR), [196](#), [198](#), [199](#), [202](#)
- Smartphone, [19](#), [27](#), [31](#)
- spatial domain, [60](#), [104](#), [105](#), [140](#)
- speech, [183](#)
- steganography, [9](#), [16](#), [155](#), [156](#), [181](#), [182](#), [184](#), [186](#), [195](#)
- subband, [23](#), [26](#), [108](#)
- synchronization, [79](#), [82](#), [95](#), [104-106](#), [182](#), [184](#)
- temporal domain, [60](#), [212](#)
- trusted communication, [139](#)
- vector quantization, [11-3](#), [16](#), [158](#)
- codebook, [158](#)
- side match VQ, [158](#)

vector quantization (VQ), [140](#)

video, [81](#), [139](#), [155](#), [159](#)

watermarking, [19](#), [20](#), [55](#), [64](#), [71](#), [73](#), [74](#), [76](#)

[78](#)-[80](#), [82](#), [85](#), [87](#), [89](#)-[91](#), [93](#)-[95](#), [103](#)

[139](#), [140](#), [156](#), [211](#), [212](#)

Watson's perceptual model, [89](#)

WAV, [181](#), [184](#), [187](#), [189](#), [193](#), [194](#)

wavelet packet transform (WPT), [139](#), [142](#)

[143](#), [145](#), [147](#), [151](#)

basis, [139](#)

zerotree, [142](#)