

## Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be “used for any purpose other than private study, scholarship, or research.” If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of “fair use” that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

**Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation**

Printing note: If you do not wish to print this page, then select “Pages from: first page # to: last page #” on the print dialog screen



The Van Houten library has removed some of the personal information and all signatures from the approval page and biographical sketches of theses and dissertations in order to protect the identity of NJIT graduates and faculty.

## ABSTRACT

### IMAGE WATERMARKING, STEGANOGRAPHY, AND MORPHOLOGICAL PROCESSING

by  
Yi-Ta Wu

With the fast development of computer technology, research in the fields of multimedia security, image processing, and robot vision have recently become popular. Image watermarking, steganographic system, morphological processing and shortest path planning are important subjects among them. In this dissertation, the fundamental techniques are reviewed first followed by the presentation of novel algorithms and theorems for these three subjects.

The research on multimedia security consists of two parts, image watermarking and steganographic system. In image watermarking, several algorithms are developed to achieve different goals as shown below. In order to embed more watermarks and to minimize distortion of watermarked images, a novel watermarking technique using combinational spatial and frequency domains is presented. In order to correct rounding errors, a novel technique based on the *genetic algorithm* (GA) is developed. By separating medical images into *Region of Interest* (ROI) and non-ROI parts, higher compression rates can be achieved where the ROI is compressed by lossless compression and the non-ROI by lossy compression. The GA-based watermarking technique can also be considered as a fundamental platform for other fragile watermarking techniques. In order to simplify the selection and integrate different watermarking techniques, a novel

adjusted-purpose digital watermarking is developed. In order to enlarge the capacity of robust watermarking, a novel robust high-capacity watermarking is developed. In steganographic system, a novel steganographic algorithm is developed by using GA to break the inspection of steganalytic system.

In morphological processing, the GA-based techniques are developed to decompose arbitrary shapes of big binary structuring elements and arbitrary values of big grayscale structuring elements into small ones. The decomposition is suited for a parallel-pipelined architecture. The techniques can speed up the morphological processing and allow full freedom for users to design any type and any size of binary and grayscale structuring elements.

In applications such as shortest path planning, a novel method is first presented to obtaining *Euclidean distance transformation* (EDT) in just two scans of image. The shortest path can be extracted based on distance maps by tracking minimum values. In order to record the motion path, a new chain-code representation is developed to allow forward and backward movements. By placing the smooth turning-angle constraint, it is possible to mimic realistic motions of cars. By using dynamically rotational morphology, it is not only guarantee collision-free in the shortest path, but also reduce time complexity dramatically. As soon as the distance map of a destination and collision-free codes have been established off-line, shortest paths of cars given any starting location toward the destination can be promptly obtained on-line.

**IMAGE WATERMARKING, STEGANOGRAPHY,  
AND MORPHOLOGICAL PROCESSING**

**by  
Yi-Ta Wu**

**A Dissertation  
Submitted to the Faculty of  
New Jersey Institute of Technology  
in Partial Fulfillment of the Requirements for the Degree of  
Doctor of Philosophy in Computer Science**

**Department of Computer Science**

**May 2005**

**APPROVAL PAGE**

**IMAGE WATERMARKING, STEGANOGRAPHY,  
AND MORPHOLOGICAL PROCESSING**

**Yi-Ta Wu**

Dr. Frank Shih, Dissertation Advisor  
Professor of Computer Science, NJIT

Date

Dr. Stanley Chén, Committee Member  
Senior Associate, Investment Consulting Services of UBS Financial Services Inc.

Date

Dr. Chengjun Liu, Committee Member  
Assistant Professor of Computer Science, NJIT

Date

Dr. James A. M. McHugh, Committee Member  
Professor of Computer Science, NJIT

Date

Dr. Larry O'Gorman, Committee Member  
Member of Avaya Labs, Basking Ridge, NJ

Date

Dr. Yun-Qing Shi, Committee Member  
Professor of Electrical and Computer Engineering, NJIT

Date

Copyright © 2005 by Yi-Ta Wu

ALL RIGHTS RESERVED

## BIOGRAPHICAL SKETCH

**Author:** Yi-Ta Wu  
**Degree:** Doctor of Philosophy  
**Date:** May 2005

### **Undergraduate and Graduate Education:**

- Doctor of Philosophy in Computer Science,  
New Jersey Institute of Technology, Newark, NJ, 2005
- Master of Science in Computer Science and Information Engineering  
National Dong-Hwa University, Hualien, Taiwan, 1997
- Bachelor of Science in Physics,  
Tamkang University, Taipei, Taiwan, 1995

**Major:** Computer Science

### **Presentations and Publications:**

#### **Journal Articles:**

Frank Y. Shih and Yi-Ta Wu,  
“Decomposition of arbitrary gray-scale morphological structuring elements,”  
Pattern Recognition, accepted.

Frank Y. Shih and Yi-Ta Wu,  
“Note: Decomposition of binary morphological structuring elements based on  
genetic algorithms,”  
Computer Vision and Image Understanding, accepted.

Frank Y. Shih and Yi-Ta Wu,  
“Robust watermarking and compression for medical images based on genetic  
algorithms,”  
Information Sciences, in press, 2005.



- Frank Y. Shih and Yi-Ta Wu,  
“Enhancement of image watermark retrieval based on genetic algorithm,”  
Journal of Visual Communication and Image Representation, vol. 16, pp.  
115-133, 2005.
- Frank Y. Shih and Yi-Ta Wu, and B. Chen,  
“Forward and backward chain-code representation for motion planning of cars,”  
Pattern Recognition and Artificial Intelligence, vol. 18, no. 8, pp. 1437-1451,  
2004.
- Yi-Ta Wu and Frank Y. Shih,  
“An adjusted-purpose digital watermarking technique,”  
Pattern Recognition, vol. 37, no. 12, pp. 2349-2359, Dec. 2004.
- Frank Y. Shih and Yi-Ta Wu,  
“The efficient algorithms for achieving euclidean distance transformation,”  
IEEE Trans. Image Processing, vol. 13, no. 8, pp. 1078-1091, 2004.
- Frank Y. Shih and Yi-Ta Wu,  
“Fast euclidean distance transformation in two scans using a  $3 \times 3$  neighborhood,”  
Computer Vision and Image Understanding, vol. 93, no. 2, pp. 195-205, Feb.  
2004.
- Frank Y. Shih and Yi-Ta Wu,  
“Three-dimensional Euclidean distance transformation and its application to  
shortest path planning,”  
Pattern Recognition, vol. 37, no. 1, pp. 79-92, Jan. 2004.
- Frank Y. Shih and Yi-Ta Wu,  
“Combinational image watermarking in the spatial and frequency domains,”  
Pattern Recognition, vol. 36, no. 4, pp. 969-975, April 2003.

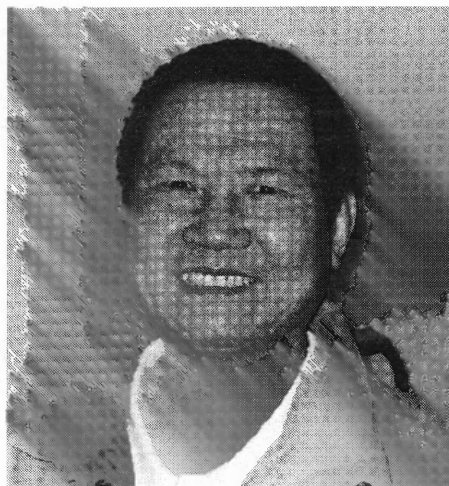
### **Conference Papers:**

- Frank Y. Shih and Yi-Ta Wu,  
“A novel fragile watermarking technique,”  
IEEE Inter. Conf. on Multimedia and Expo, Taiwan, June 2004.
- Frank Y. Shih and Yi-Ta Wu,  
“A deriving two-scan approach for exact Euclidean distance transformation,”  
7th Joint Conferences on Information Sciences, Cary, NC, Sep. 2003.
- Yi-Ta Wu and Frank Y. Shih,  
“Multimedia Compression and Watermarking Using Genetic Algorithms,”  
Proc. 2003 Computer Graphics Workshop, National Dong Hwa University,  
Hualien, Taiwan, Aug. 2003.

Frank Y. Shih and Yi-Ta Wu,  
“Image watermarking in both spatial and frequency domains,”  
Proc. 2002 Computer Graphics Workshop, Taiwan, June 2002.

Frank Y. Shih and Yi-Ta Wu,  
“Genetic algorithms based error correction in image watermarking,”  
Proc. 2002 Computer Graphics Workshop, Taiwan, June 2002.

To my beloved family



Father



Mother



Second Brother



Third Brother

## ACKNOWLEDGMENT

I would like to express my sincere appreciation to Dr. Frank Shih, who patiently served as my research supervisor, providing valuable and countless insight and intuition. Special thanks are given to Dr. Stanley Chen, Dr. Chengjun Liu, Dr. James McHugh, Dr. Larry O'Gorman and Dr. Yun-Qing Shi for actively participating in my committee and providing valuable advice.

I extend special thanks to my fellow graduate students for their assistance and support, in particular to Yoo Jung An, Shouxian Cheng, Chao-Fa Chuang, Yan-Yu Fu, Ming Qu, Peichung Shih, and Kai Zhang.

I also wish to thank my parents and my brothers for their support and encouragement.

## TABLE OF CONTENTS

<b>Chapter</b>	<b>Page</b>
1 INTRODUCTION.....	1
1.1 Motivation.....	1
1.2 Organization of This Dissertation.....	6
2 MULTIMEDIA SECURITY .....	8
2.1 Introduction .....	8
2.2 Notations and Definitions in the Image Watermarking.....	13
2.2.1 Watermark Embedding Approaches .....	13
2.2.2 Wong’s Algorithm .....	13
2.2.3 The Vector Quantization Counterfeiting Attack.....	15
2.2.4 The Hierarchical Watermarking Approach.....	16
2.2.5 The Morphological Approach for Extracting Pixel-Based Features.....	18
2.2.6 Least Signification Bit (LSB) Substitution.....	20
2.2.7 Discrete Cosine Transformation (DCT).....	21
2.2.8 Discrete Wavelet Transformation (DWT) .....	22
2.2.9 Set Partitioning in Hierarchical Tree (SPIHT) .....	24
2.2.10 Chaotic Map .....	24
2.2.11 GA-Based Watermarking .....	26
2.3 The Combinational Image Watermarking .....	28
2.3.1 The Algorithm of Combinational Image Watermarking.....	29

**TABLE OF CONTENTS**  
**(Continued)**

<b>Chapter</b>	<b>Page</b>
2.3.2 Experimental Result.....	31
2.4 GA-Based Watermarking Technique.....	33
2.4.1 The Errors Caused by Deviations in Translating Real Numbers into Integers.....	33
2.4.2 The GA-Based Watermarking for Rounding Error Correction.....	36
2.4.3 Experimental Results.....	39
2.4.4 The Further Enhancement of GA-Based Algorithm.....	40
2.5 GA-Based Fragile Watermarking Technique.....	44
2.5.1 The Improved Scheme of the GA-based Watermarking.....	45
2.5.2 The Platform for the Existing Fragile Watermarking Technique.....	46
2.5.3 Experimental Results.....	47
2.6 GA-Based Robust Watermarking for Medical Image.....	50
2.6.1 The Proposed Algorithm.....	50
2.6.2 The Algorithm for the Proposed Technique.....	53
2.6.3 Experimental Results.....	55
2.7 The Adjusted-Purpose Watermarking Technique.....	57
2.7.1 The Strategies for Adjusting $VSTW$ and $QF$ .....	57
2.7.2 The Proposed Adjusted-Purpose Watermarking Technique.....	61
2.7.3 Experimental Results .....	63
2.8 The Robust High-Capacity Watermarking Technique.....	67
2.8.1 The Block-Based Chaotic Map.....	68
2.8.2 The Reference Register.....	69

**TABLE OF CONTENTS**  
**(Continued)**

<b>Chapter</b>	<b>Page</b>
2.8.3 The Algorithm of the Robust High-Capacity Watermarking.....	72
2.8.4 Experimental Results .....	77
2.9 Genetic Algorithm Based Methodology for Breaking the Steganalytic Systems.....	84
2.9.1 Generating the Stego-image on the Visual Steganalytic System .....	86
2.9.2 Generating the Stego-image on the IQM-based Steganalytic System...	87
2.9.3 The GA-based Breaking Algorithm on FDSS.....	89
2.9.4 Experimental Results .....	92
3 MORPHOLOGICAL PROCESSING .....	97
3.1 Introduction.....	97
3.2 Notations and Definitions in the Mathematical Morphology.....	100
3.2.1 Binary Mathematical Morphology .....	100
3.2.2 Grayscale Mathematical Morphology .....	102
3.3 Decomposition of Binary Morphological Structuring Elements.....	104
3.3.1 Overview.....	104
3.3.2 An Example.....	106
3.3.3 Decomposition Using Genetic Algorithms.....	107
3.4 Decomposition of Grayscale Morphological Structuring Elements.....	111
3.4.1 Decomposition of 1-D Arbitrary Grayscale Structuring Elements.....	113
3.4.2 Decomposition of 2-D Arbitrary Grayscale Structuring Elements.....	118
3.5 Experimental Results.....	121
3.5.1 Binary.....	121

**TABLE OF CONTENTS**  
**(Continued)**

<b>Chapter</b>	<b>Page</b>
3.5.2 Grayscale.....	125
4 SHORTEST PATH PLANNING .....	127
4.1 Introduction.....	127
4.2 Notations and Definitions in the Shortest Path Planning.....	129
4.2.1 The Properties of a Distance Function.....	129
4.2.2 3-D Image Representation.....	129
4.2.3 Distance Transformation in 3-D Domain.....	130
4.2.4 3-D Neighborhood Used in EDT.....	131
4.2.5 Forward and Backward Chain-Code Representation.....	132
4.2.6 Rotational Morphology.....	133
4.3 Euclidean Distance Transformation.....	134
4.3.1 The Basic Morphological Approach.....	136
4.3.2 The Two-Scan Based Algorithm.....	138
4.3.3 The Fundamental Lemmas.....	141
4.3.4 $TS_1$ - Two-Scan Based EDT Algorithm for General Images.....	143
4.3.5 $TS_\infty$ - Two-Scan Based EDT Algorithm for Images with Obstacles.....	147
4.3.6 Computational Complexity.....	149
4.3.7 The Extension to the 3-D Space.....	151
4.4 The EDT-Based SPP Approach.....	155
4.4.1 Improvement on the Rotational Mathematical Morphology.....	156
4.4.2 The Novel Algorithm for Shortest Path Planning.....	157



**TABLE OF CONTENTS**  
**(Continued)**

<b>Chapter</b>	<b>Page</b>
4.4.3 Examples of Our Algorithm.....	158
4.5 Car-Like Object SPP Problem.....	160
4.5.1 The SPP Algorithm for a Car-Like Object.....	160
4.5.2 Experimental Results.....	162
4.5.3 The Extension to the 3-D Space.....	165
4.5.4 The Rule of Distance Functions for Shortest Path Planning.....	167
5 SUMMARY AND FUTURE RESEARCH.....	168
5.1 The Contribution of This Dissertation.....	168
5.1.1 Theoretical Research.....	168
5.1.2 Application Research.....	170
5.2 Future Research.....	171
REFERENCES.....	173

## LIST OF TABLES

<b>Table</b>		<b>Page</b>
2.1	Position of a Pixel and Its Neighborhood in Different Compression rates .....	19
2.2	Comparisons When Embedding Different Sized Watermarks .....	33
2.3	Total Possible Embedded Watermarks .....	34
2.4	Comparisons Between Round and GAs .....	44
2.5	Bit-Stream Extracted from Specific Positions of the Binary Form .....	49
2.6	PSNR of Signature and of Medical Images .....	57
2.7	Relation Between QF and Embedded Positions .....	58
2.8	General Rules for Determining VSTW and QF .....	60
2.9	Quantitative Measures for Pixel-Based Features .....	64
2.10	Pixel-Based Features Based on Sizes of Structuring Elements .....	65
2.11	Effect of $RR_{th}$ .....	83
2.12	Experimental Results of GA-based Algorithm on IQM-SDSS .....	94
3.1	Comparison Between Traditional and Proposed Algorithm .....	125
4.1	Relationship Between SEDT and Neighborhood Window .....	144
4.2	Lookup Table for Obtaining SED of Neighbors of a Pixel $p$ .....	146
4.3	Experimental Results for Tests 4.2 and 4.3 .....	151
4.4	3-D Chain-Code Representation by Using 26 Alphabets .....	166

## LIST OF FIGURES

<b>Figure</b>		<b>Page</b>
2.1	Generic watermark embedding approach in the spatial domain .....	13
2.2	Generic watermark embedding approach in the frequency domain .....	13
2.3	Wong's watermarking insertion algorithm .....	14
2.4	Wong's watermarking extraction algorithm .....	14
2.5	The VQ counterfeit attack .....	16
2.6	An example of the hierarchy approach with three levels.....	17
2.7	An example of illustrating the property of pixel-based features .....	19
2.8	The circular GMSEs .....	20
2.9	LSB substitution .....	21
2.10	An example of Haar DWT .....	23
2.11	The example of SPIHT .....	23
2.12	An example of enlarging the significant coefficients .....	25
2.13	An example of performing the chaotic map .....	25
2.14	The numbering positions corresponding to 64 genes .....	26
2.15	Basic steps in finding the best solutions by Genetic Algorithms .....	28
2.16	The flowchart in combinational spatial and frequency domains .....	29
2.17	A $64 \times 64$ image is cut from a $256 \times 256$ binary image .....	31
2.18	The results when embedding a $128 \times 128$ watermark into the Lena image .....	32
2.19	The errors caused by the Round technique .....	35
2.20	The positions where watermark is embedded .....	35

**LIST OF FIGURES  
(Continued)**

<b>Figure</b>	<b>Page</b>
2.21 The flowchart in reducing the errors by using Genetic Algorithm .....	37
2.22 The positions corresponding to the Genes .....	38
2.23 The usage of the Genes .....	39
2.24 The Restoration by Genetic Algorithms .....	40
2.25 Better suitable solution obtained by GAs .....	41
2.26 Solution in embedding 1011 by GAs .....	43
2.27 The example when embedding watermark into a real image .....	43
2.28 The encoding procedure of the improved scheme .....	45
2.29 The decoding procedure of the improved scheme .....	45
2.30 The insertion procedure of the GA-based watermarking platform .....	47
2.31 The extraction procedure of the GA-based watermarking platform .....	47
2.32 The example of the improved scheme of our GA-based watermarking .....	48
2.33 An example of the improved genetic algorithms .....	49
2.34 The encoding procedure of embedding a signature image .....	51
2.35 The decoding procedure of embedding a signature image .....	52
2.36 The encoding procedure of embedding textual data .....	52
2.37 The decoding procedure of embedding textual data .....	52
2.38 The example of embedding signature image .....	55
2.39 The example of embedding textual data .....	56
2.40 The example of embedding watermarks into different spectral regions .....	59
2.41 The example of enlarging the capacity of watermark .....	60

**LIST OF FIGURES  
(Continued)**

<b>Figure</b>	<b>Page</b>
2.42 The encoding procedure of our AP watermarking .....	62
2.43 The decoding procedure of our AP watermarking .....	63
2.44 The example of the pixel-based features of an image .....	64
2.45 The example of our AP watermarking technique .....	65
2.46 The example of extracting embedded watermarks after JPEG compression ..	66
2.47 The QFs for their corresponding VSTW's .....	66
2.48 An example of the block-based relocation .....	68
2.49 An example of performing the block-based chaotic map .....	69
2.50 An example shows how the IBPC works .....	70
2.51 The similarity illustration of two sub-images obtained by IBPC .....	71
2.52 The quantization table of JPEG .....	71
2.53 The embedding procedure or the RHC watermarking .....	73
2.54 The extracting procedure or the RHC watermarking .....	75
2.55 An example of the embedding strategy .....	78
2.56 An example of the extracting strategy .....	79
2.57 An example of robustness experiment using JPEG compression .....	80
2.58 An example of robustness experiment using low-pass filter .....	81
2.59 An example of robustness experiment using Gaussian noise .....	82
2.60 Robustness versus JPEG compression .....	82
2.61 Robustness versus Gaussian noise .....	83
2.62 An example of our GA-based algorithm on the JFDSS .....	92

**LIST OF FIGURES**  
**(Continued)**

<b>Figure</b>	<b>Page</b>
2.63 A stego-image generated by our GA-based algorithm and its 8 bit-planes ....	93
2.64 A stego-image and its visual filtered result .....	93
2.65 The relationship between the average iteration and the correct rate .....	93
2.66 An example of adjusting an 8×8 embedded-image .....	96
2.67 The relationship between the modification probability and the iterations ....	96
3.1 An example of translation .....	100
3.2 An example of reflection .....	100
3.3 An example of binary dilation and erosion .....	101
3.4 An example of binary opening and closing .....	102
3.5 An example of grayscale morphological operations .....	103
3.6 The flowchart of recursively decomposing structuring elements .....	105
3.7 The parallel pipelined architecture .....	107
3.8 The positions correspond to the genes .....	108
3.9 Basic steps in finding the best solutions .....	110
3.10 Dilation of a structuring element verifying proposition 1 .....	115
3.11 The flowchart of the 1-D structuring element decomposition .....	118
3.12 The row decomposition of a 2-D structuring element .....	118
3.13 The representation of the translation .....	119
3.14 The flowchart of the combined row-and-column decomposition .....	119
3.15 An example of decomposing a structuring element .....	122
3.16 The representation of translation .....	122

**LIST OF FIGURES**  
(Continued)

<b>Figure</b>	<b>Page</b>
3.17 An example of decomposing a structuring element .....	122
3.18 The decomposition of a $17 \times 17$ structuring element .....	123
3.19 The decomposition of arbitrarily shaped structuring elements .....	124
3.20 The row decomposition of a 2-D structuring element of size $5 \times 5$ .....	125
3.21 The column decomposition of a 2-D structuring element of size $5 \times 5$ .....	125
3.22 The representation of translation .....	126
3.23 The combined dilation and maximum operators .....	126
4.1 The 3-D image representation .....	130
4.2 The representation of the $n$ -dimensional space .....	130
4.3 The 26 neighbors of pixel $p$ .....	131
4.4 The representation of 4- and 8-connectivity chain codes .....	132
4.5 The new chain-code representation .....	132
4.6 The examples of (a) the traditional (b) the new chain codes .....	133
4.7 The flowchart of the Iterative Erosion Algorithm (IEA) .....	138
4.8 Examples of the acquiring procedure .....	138
4.9 Examples of the deriving approach .....	139
4.10 The examples when calculating an image based on the deriving approach ...	140
4.11 The $D_4(p)$ and $D_8(p)$ of pixel $p$ .....	142
4.12 The relation between $p$ and its parents .....	143
4.13 The example of the neighborhood window of pixel $p$ .....	144
4.14 The flowchart of the $TS_1$ algorithm .....	144

**LIST OF FIGURES**  
**(Continued)**

<b>Figure</b>	<b>Page</b>
4.15 The example that some algorithms cannot obtain the exact EDT .....	148
4.16 The testing images .....	150
4.17 Test 4.1: diameters vary from 50 to 1000 pixels .....	150
4.18 The image with obstacles .....	151
4.19 The 13 children in the forward scanning .....	152
4.20 The 13 children in the backward scanning .....	152
4.21 The flowchart of the two-scan based algorithm .....	152
4.22 The forward scanning of the two-scan based algorithm .....	154
4.23 The backward scanning of the two-scan based algorithm .....	155
4.24 The example of DRMM .....	157
4.25 The flowchart for achieving SPP based on DRMM .....	158
4.26 The experimental result for achieving SPP based on DRMM .....	159
4.27 The SPP example of an actual image .....	159
4.28 The flowchart of the shortest path planning .....	161
4.29 The turning cases for different degrees .....	162
4.30 The example of the shortest path planning in the different car orientation ....	163
4.31 The shortest path planning with different starting points .....	163
4.32 The shortest path planning with backward movement .....	164
4.33 Shortest path planning for parallel parking .....	164
4.34 The 26 neighbors of a pixel $p$ and its 3-D chain-code representation .....	166
4.35 The example of the 3-D shortest path planning .....	167



# CHAPTER 1

## INTRODUCTION

### 1.1 Motivation

With the fast development of computer technology, the research in the fields of multimedia security, image processing, and robot vision has recently become popular. Image watermarking, steganographic system, morphological processing, and shortest path planning are important subjects among them. In this dissertation, the fundamental techniques will be first reviewed, and the developments of the novel algorithms and theorems will be presented later.

#### **A. Multimedia Security:**

Digital images, video, and audio have revolutionized in the way of largely stored, manipulated, and transmitted. It gives rise to a wide range of applications in electronics, entertainment and medial industry [7]. Creative ways of storing, accessing and distributing data have generated lots of benefits into the digital multimedia field. However, these benefits brought with concomitant risks of data piracy. One of the solutions to provide security in copyright protection is digital watermarking which embeds special marks into a host image.

There are many aspects to be noticed in watermarking design, for example, imperceptibility, security, capacity and robustness. Many researchers have been focusing on security and robustness, but rarely on the watermarking capacity [5, 58]. Indeed, both security and robustness are important because the watermark images are expected to be indelible and unperceivable. Nevertheless, if a large watermark image can be embedded

into a host image, the application becomes more flexible in many areas. The research for enlarging the capacity of the watermark is concerned in this dissertation.

According to the embedding purposes, watermarks can be categorized into three types: robust, fragile, and semi-fragile. Robust watermarks [18, 19, 54, 59] are designed to withstand arbitrarily malicious attacks, such as image scaling, bending, cropping, and lossy compression. They are usually used for copyright protection to declare the rightful ownership. On the contrary, for the purpose of image authentication, fragile watermarks [15, 16, 66, 103, 104] are adopted to detect any unauthorized modification. The semi-fragile watermarks [1] are designed for detecting any unauthorized modification, at the same time allowing some image processing operations. In other words, it is for selective authentication that detects illegitimate distortion, while ignoring applications of legitimate distortion. A widely-used fragile watermarking technique, presented by Wong [104], is a block-based approach by adopting the *Rivest-Shamir-Adleman* (RSA) public key encryption algorithm [70] and *Message Digest 5* (MD5) [69] for the hash function. Holliman and Memon [43] developed a counterfeiting attack to forge the fragile watermark by exploiting the blockwise independence of Wong's algorithm [104]. In order to defeat the counterfeiting attack, Celik et al. [16] proposed a hierarchical watermarking approach to improve Wong's algorithm.

Since the existing block-based approaches are performed by embedding watermarks into *least significant bit* (LSB) of the spatial domain, the attacks on the watermarking techniques can be considered as the attacks on the cryptography techniques. With the technological advancement, the current cryptography may not be secured by some attacks someday. For example, the algorithms of MD5 and RSA adopted

by Wong [104] are not safe anymore [8, 60, 89]. Although another cryptography can be adopted for watermarking, the scheme of embedding watermarks into the spatial domain is not a good approach. That is because the watermarks can be extracted by some sophisticated calculations. Therefore, a new fragile watermarking technique based on genetic algorithms to embed watermarks into the frequency domain is developed in this dissertation.

As mentioned earlier, there are lots of different watermarking techniques for variant types of watermarks. Therefore, the selection of a suitable watermarking technique is not an easy task. In this dissertation, a novel adjusted-purpose digital watermarking technique is presented to simplify the selection and to integrate different watermarking techniques.

It is obvious that the frequency-domain watermarking possesses the property of robustness since the embedded messages are spread out all over the spatial extent in an image [19]. Moreover, if the messages are embedded into the large values (which are called “*significant coefficients*”) of a transformed image, the watermarking technique will be more robust. However, most images contain only few significant coefficients; this will lower the watermarking capacity. Therefore, researchers have intended to overcome this difficulty. In order to enlarge the watermark capacity, a robust high-capacity watermarking is developed.

Steganalytic systems are used to detect whether an image contains a hidden message. By analyzing various image features between stego-images (the images containing hidden messages) and cover-images (the images containing no hidden messages), a steganalytic system is able to detect stego-images. In this dissertation, a new

concept of developing a robust steganographic system is developed by artificially counterfeiting statistic features instead of the traditional strategy by avoiding the change of statistic features. Genetic algorithm (GA) based methodology is utilized to adjust the gray values of a cover-image while creating the desired statistic features to generate the stego-images that can break the inspection of steganalytic systems.

### **B. Morphological Processing:**

Mathematical morphology, based on set-theoretic concept, can extract object features by designing a suitable structuring shape as a probe [36, 76, 80]. The morphological operations can be employed for many purposes, including digitization, compression, enhancement, restoration, reconstruction, matching, segmentation, representation and description. Those operations deal with two images; the image being processed is referred to as the active image, and the other image being a kernel is referred to as the structuring element. With the help of various designed structuring shapes, one can simplify image data representation and expose object shape characteristics.

Most image processing architectures adapted to morphological operations use structuring elements of a limited size. Implementation becomes difficult when dealing with a large sized structuring element. Hence, in this dissertation, the techniques are developed for efficiently decomposing a large structuring element into combined structures of small components.

### **C. Shortest Path Planning:**

In the field of robotics and artificial intelligence, the stimulated considerable interests have focus on the robot motion planning and the shortest path planning problems [51]. The path planning is in general concerned with finding paths connecting different

locations in an environment. Depending on the specific applications, the desired paths often need to satisfy some constraints and optimize certain criteria. The problems of planning shortest paths arise in many disciplines, and in fact are one of the several most powerful tools for modeling the combinatorial optimization problems.

Two approaches in path planning are often used. One is via configuration space where the geometry of the robot is added to the obstacle in order to create a new free space [55]. Another approach uses the computational geometry to search the space directly without transforming the workspace to the configuration space [11, 47]. Generally, the computational complexity becomes extremely high when the degree of freedom is large. Therefore, how to efficiently achieve the shortest-path-planning problem is an important topic. In this dissertation, the author focuses on the configuration space approach and develop a new approach to obtain the shortest collision-free path by combining rotational mathematical morphology and distance transformation.

Distance transformation (DT) is used to convert a digital binary image that consists of object (foreground) and nonobject (background) pixels into another image in which each object pixel has a value corresponding to the minimum distance from the background by a distance function [98]. The interior of a closed boundary is considered as object pixels and the exterior as background pixels. Among different kinds of distance transformation, *Euclidean distance transform* (EDT) is often-used because of its rotation invariance property, but it involves the time-consuming calculations such as square-root, and the minimum over a set of floating-point numbers. Although many techniques have been presented [20, 22, 24, 45, 82, 85] for obtaining EDT, most of them are either

inefficient or complex to implement and understand. Therefore, to develop an efficient algorithm for performing EDT is one of goals in this dissertation.

## 1.2 Organization of This Dissertation

The organization of this dissertation will be given in this section. The outline of this dissertation is described follows.

Chapter 1 Introduction

Chapter 2 Multimedia Security

Chapter 3 Morphological Processing

Chapter 4 Shortest Path Planning

Chapter 5 Summary and Future Research

The brief statements of each chapter are also given which will be the abstract of the topic discussed in each chapter.

In Chapter 2, a novel technique using the combinational spatial and frequency domains is first presented. The splitting of the watermark image into two parts respectively for spatial and frequency insertion relies on the user's preference and data importance. Then, the GA-based watermarking algorithm is developed to provide a novel approach and resolve some problems such as rounding error problem. The GA-based watermarking can be utilized to the medical image for increasing the compression rate. Also, it can be considered as a fundamental platform for other fragile watermarking techniques. In order to simplify the selection and integrate currently different watermarking techniques, a novel adjusted-purpose digital watermarking is developed. Finally, a novel robust high-capacity watermarking is developed. In the steganographic

system, a novel steganographic algorithm is developed by using GA to break the inspection of steganalytic system.

In Chapter 3, several efficient techniques are developed to decompose arbitrary shapes of big binary structuring elements by applying genetic algorithms, and to decompose arbitrary values of big grayscale structuring elements into small ones. The advantages of the new algorithms are not only decomposing arbitrary structuring elements, but also simplifying the computation. Furthermore, the decomposition is suited for a parallel pipelined architecture.

In Chapter 4, the shortest path is obtained based on the distance maps using the minimum value tracing. Secondly, a new chain-code representation is developed to record the motion path when forward and backward movements are allowed. By placing the smooth turning-angle constraint, more realistic results can be obtained to the actual motion of cars. Meanwhile, by combining rotational mathematical morphology and distance transformation, the shortest collision-free path can be derived. In order to generate the distance map, EDT is adopted. The efficient algorithms for performing EDT are also developed. The improved iterative erosion algorithm is proposed to avoid the redundant calculations in the iterative erosion algorithm. Furthermore, to avoid the iterative operations, the two-scan based algorithm by a deriving approach is developed for achieving EDT correctly and efficiently in a constant time. Besides, when obstacles appear in the image, many algorithms cannot achieve the correct EDT except the proposed two-scan based algorithm.

In Chapter 5, the further works and summary of this dissertation are provided.

## **CHAPTER 2**

### **MULTIMEDIA SECURITY**

Several image-watermarking and steganographic techniques are investigated in this chapter. In image watermarking, several algorithms are developed to achieve different goals as shown below. In order to embed more watermarks and to minimize distortion of watermarked images, a novel watermarking technique using combinational spatial and frequency domains is presented. In order to correct rounding errors, the genetic-algorithm (GA) based watermarking is developed. By separating medical images into Region of Interest (ROI) and non-ROI parts, higher compression rates can be achieved. The GA-based watermarking can be considered as a fundamental platform for other fragile watermarking techniques. In order to simplify the selection and to integrate currently different watermarking techniques, a novel adjusted-purpose digital watermarking is developed. In order to enlarge the capacity of robust watermarking, a novel robust high-capacity watermarking is developed. In steganographic system, a novel steganographic algorithm is developed by using GA to break the inspection of steganalytic system.

#### **2.1 Introduction**

##### **A. Digital Watermarking:**

Digital watermarking has been identified as a suitable tool to identify the source, creator, owner, distributor, or authorized consumer of a document or an image [7]. It can also be used for tracing images that have been illegally distributed. Watermarking, when



complemented with encryption, can serve for many purposes, such as copyright protection, broadcast monitoring, and data authentication.

There are two methods of performing watermarking: one in spatial domain and the other in frequency domain. Generally, the size of watermark is limited. Therefore, in order to provide more watermark data and to minimize the distortion of the watermarked image, a novel technique using the combinational spatial and frequency domains is developed in this chapter.

After embedding watermarks into the frequency domain of a host image, the inverse transform must be performed to translate the image from its frequency domain to spatial domain. However, the pixel values of the transformed image must be real numbers. That is, the data embedded in the coefficients of a transformed image will be somewhat disturbed in the process of transforming the image from its frequency domain into its spatial domain because of deviations in converting real numbers into integers in the spatial domain. In order to reduce the rounding errors, the GA-based watermarking is developed in this dissertation. Note that, when embedding watermarks into the frequency domain of a transformed image, the rounding errors are usually happened in the fragile watermarks since they are usually embedded into the LSB of the coefficients of a transformed image. That is, they are so sensitive. However, the GA-based watermarking can successfully correct the rounding errors for both robust and fragile watermarks since the rounding errors may be happened on the perceptually significant bits of the coefficients of a transformed image.

According to embedding purposes, watermarks can have two types: robust and fragile watermarks. Robust watermarks [18, 19, 54, 59] are designed to withstand arbitrarily malicious attacks, such as image scaling, bending, cropping, and lossy compression. They are usually used for copyright protection to declare the rightful ownership. On the contrary, for the purpose of image authentication, fragile watermarks [15, 16, 66, 103, 104] are adopted and designed to detect any unauthorized modification.

A widely-used fragile watermarking technique, presented by Wong [104], is a block-based approach by adopting the *Rivest-Shamir-Adleman* (RSA) public key encryption algorithm [70] and *Message Digest 5* (MD5) [69] for the hash function. Holliman and Memon [43] developed a counterfeiting attack to forge the fragile watermark by exploiting the blockwise independence of Wong's algorithm. In order to defeat the counterfeiting attack, Celik et al. [16] proposed a hierarchical watermarking approach to improve Wong's algorithm. Since the existing block-based approaches are performed by embedding watermarks into LSB of the spatial domain, the attacks on the watermarking techniques can be considered as the attacks on the cryptography techniques. With the technological advancement, the current cryptography may not be secured by some attacks someday. For example, the algorithms of MD5 and RSA adopted by Wong [104] are not safe anymore [8, 60, 89]. Although the security of watermarking can be improved by combining with cryptanalysis, the scheme of embedding watermarks into the spatial domain is not a good approach. That is because watermarks can be extracted by some sophisticated calculations. Therefore, the GA-based watermarking can be adopted to enhance the security. Also, it can also be considered as the fundamental platform of current fragile watermarking techniques.

Selection of a suitable watermarking technique is not easy since there exist many kinds of different watermarking techniques for variant types of watermarks. In this dissertation, a novel adjusted-purpose digital watermarking technique is presented to simplify the selection and to integrate different watermarking techniques. The quantity factor is used to affect the embedded watermarks to become fragile, semi-fragile or robust watermarks. The varying sized transform window is designed to determine whether the embedded strategy should be in spatial or frequency domains. Furthermore, by extracting the pixel-based features, the adjusted-purpose watermarking technique not only can improve robustness of embedded watermarks, but also can provide an integrated view in studies of digital watermarking

Usually, the capacity of a robust watermarking is limited. For example, a robust watermarking algorithm presented by Miller et al. [57] can only embed 1380 bits of information in images with dimensions  $240 \times 368$  pixels. That is, the capacity of their robust watermarking is 0.015625 bits/pixel. In order to enlarge the watermark capacity, a robust high-capacity watermarking is developed to achieve the goal.

### **B. Steganographic System:**

Steganography is an ancient art of hiding secret messages within another apparently innocuous carrier in such a way that only the receiver knows the existence of the messages. It is different from cryptography which encodes messages, so that nobody can read it without the specific key. Although steganography is an ancient subject, computer technology provides a new aspect of applications by hiding messages in the multimedia materials, such as audio and images.

A famous classic steganographic model presented by Simmons [90] is the prisoners' problem that Alice and Bob in a jail plan to escape together. All communications between them are monitored by Wendy, a warden. Therefore, they must hide the messages in other innocuous-looking media (cover-object) in order to obtain the stego-object. Then, the stego-object is sent through the public channel. Wendy is free to inspect all the messages between Alice and Bob with two options, passive or active. The passive way is to inspect the message in order to determine whether it contains a hidden message and then to conduct a proper action. On the other hand, the active way is always to alter messages though Wendy may not perceive any trace of a hidden message. Note that, only the passive warden is concerned in this dissertation.

In this dissertation, a novel concept for breaking steganalytic systems is presented. The emphasis is shifted from traditionally avoiding the change of statistic features to artificially counterfeiting the statistic features. Therefore, the novel algorithm is based on the methodology: in order to manipulate the statistic features for breaking the inspection of steganalytic systems, the GA-based approach is adopted to counterfeit several stego-images (candidates) until one of them can break the inspection of steganalytic systems. This is the first paper of utilizing the evolutionary algorithms in the field of steganographic systems.

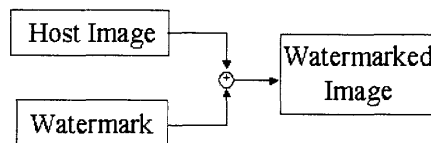
The rest of this chapter is organized as follows. Section 2.2 presents the notations and definitions used in the image watermarking. The combinational image watermarking algorithms is shown in Section 2.3 and the GA-based watermarking technique is developed in Section 2.4. The applications of the GA-based watermarking to fragile and robust watermarking techniques are presented in Sections 2.5 and 2.6, respectively. The

adjusted-purpose watermarking is presented in Section 2.7. The robust high-capacity watermarking is presented in Section 2.8. Finally, The GA-based methodology for breaking the steganalytic systems is presented in Section 2.9.

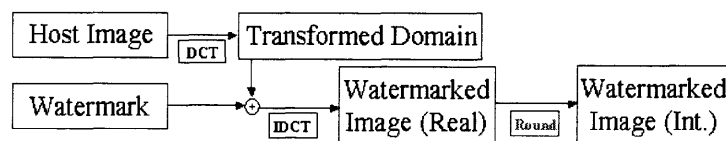
## 2.2 Notations and Definitions in the Image Watermarking

### 2.2.1 Watermark Embedding Approaches

There are two methods of performing watermarking: one in spatial domain and the other in frequency domain. In the spatial domain [13, 59], people can simply insert watermark into a host image by changing the gray levels of some pixels in the host image, but the inserted information may be easily detected using computer analysis. In the frequency domain [6, 46, 54], the watermark can be inserted into the coefficients of a transformed image. Figures 2.1 and 2.2 show the generic watermark embedding approaches on spatial and frequency domains, respectively. Note that, least significant bit (LSB) substitution is usually adopted for performing the watermark embedding procedure.



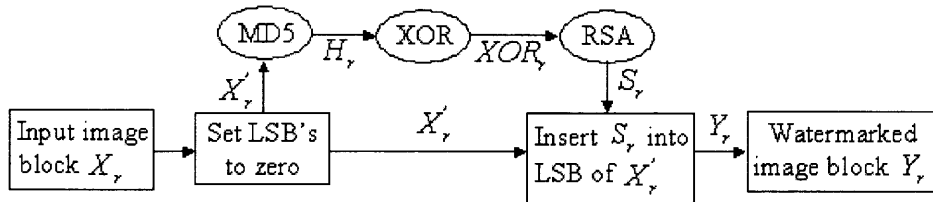
**Figure 2.1** Generic watermark embedding approach in the spatial domain.



**Figure 2.2** Generic watermark embedding approach in the frequency domain.

### 2.2.2 Wong's Algorithm

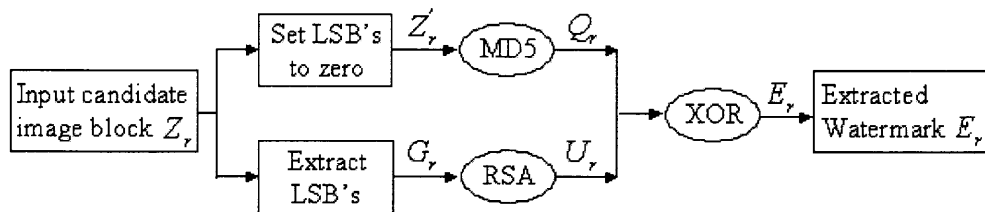
Wong's algorithm [104] is block-based and capable of detecting changes, such as pixel values and image size. Wong's watermarking insertion and extraction algorithms are introduced below, and their flowcharts are shown in Figures 2.3 and 2.4, respectively.



**Figure 2.3** Wong's watermarking insertion algorithm.

#### Wong's Watermarking Insertion Algorithm:

1. Divide the original image  $X$  into sub-images  $X_r$ .
2. Divide the watermark  $W$  into sub-watermarks  $W_r$ .
3.  $X'_r$  can be obtained from each sub-image  $X_r$  by setting LSB to be 0.
4. For each  $X'_r$ , the corresponding codes  $H_r$  can be obtained by a cryptographic hash function, e.g., MD5 or SHA [56, 69].
5.  $XOR_r$  is obtained by adopting the XOR operation of  $H_r$  and  $W_r$ .
6.  $S_r$  is obtained by encrypting  $XOR_r$  using the RSA encryption with a private key  $K'$ .
7. The watermarked sub-image  $Y_r$  is generated by embedding  $S_r$  into LSB of  $X'_r$ .



**Figure 2.4** Wong's watermarking extraction algorithm.

### **Wong's Watermarking Extraction Algorithm:**

1. Divide the candidate image  $Z$  into sub-images  $Z_r$ .
2.  $Z'_r$  is obtained by setting LSB of  $Z_r$  to be 0.
3.  $G_r$  is obtained by extracting LSB of  $Z_r$ .
4.  $U_r$  is obtained by decrypting  $G_r$  using RSA with a public key  $K$ .
5. For each  $Z'_r$ , the corresponding codes  $Q_r$  is obtained by a cryptographic hash function, e.g., MD5 or SHA.
6.  $E_r$  is the extracted watermark by adopting the XOR operation of  $Q_r$  and  $G_r$ .

### **2.2.3 The Vector Quantization Counterfeiting Attack**

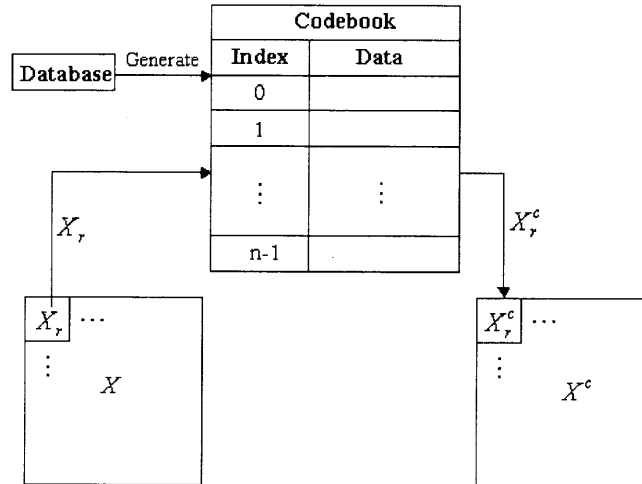
Holliman and Memon [43] developed a VQ counterfeiting attack to forge the fragile watermark by exploiting the blockwise independence of Wong's algorithm. By the exploitation, the blocks of an image can be rearranged to form a new collage image in which the embedded fragile watermarks are not altered. Therefore, given a large database of watermarked images, the VQ attack can approximate a counterfeit collage image with the same visual appearance as the original unwatermarked image from a codebook.

Let  $D$  be the large database of watermarked images and  $CB$  be the codebook with  $n$  items generated from  $D$ . The algorithm of the VQ attack is described below, and its flowchart is shown in Figure 2.5.

### **VQ Counterfeiting Attack Algorithm:**

1. Generate the codebook  $CB$  from  $D$ .
2. Divide the original image  $X$  into sub-images  $X_r$ .

3.  $X_r^c$  is obtained from each  $X_r$  by selecting the data from the codebook with the minimum distance (difference) to  $X_r$ . Finally, a counterfeit collage image  $X^c$  is generated with the same visual appearance as the original unwatermarked image.



**Figure 2.5** The VQ counterfeit attack.

#### 2.2.4 The Hierarchical Watermarking Approach

In order to defeat the counterfeiting attack, a hierarchical block-based watermarking technique was developed by Celik et al. [16]. The idea of the hierarchical watermarking approach is simply to break the blockwise independence of Wong's algorithm. That is, the embedded data not only include the information of its corresponding block, but also possess the relative information of its higher-level blocks. Therefore, the VQ attack cannot approximate an image based on the codebook that only records the information of each watermarked block.

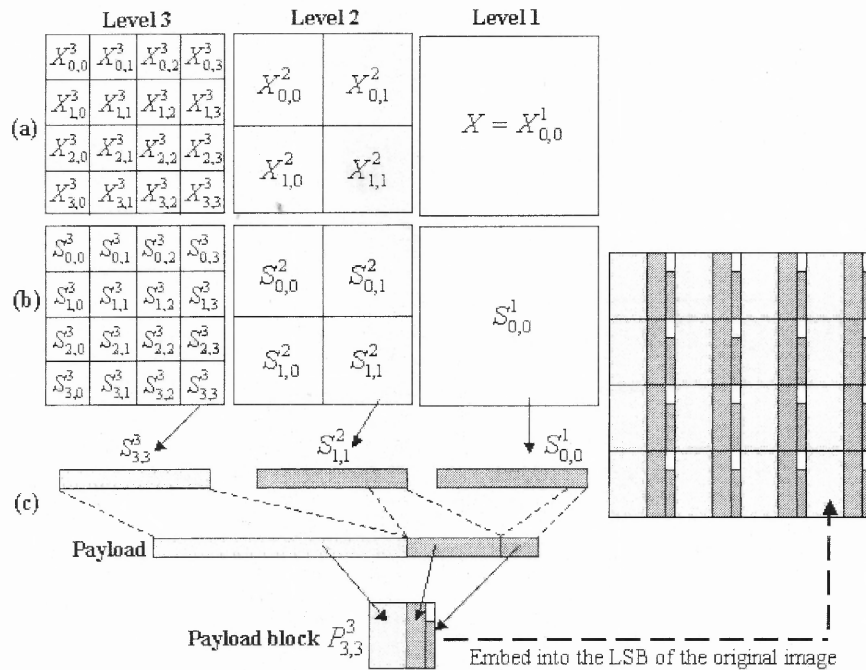
Let  $X_{i,j}^l$  denote a block in the hierarchical approach, where  $i, j$  represent the spatial position of the block and  $l$  is the level to which the block belongs. The total number of levels in the hierarchical approach is denoted by  $L$ . At each successive level, the higher-level block is divided into  $2 \times 2$  lower-level blocks. That is,



for  $l = L-1$  to 2

$$\begin{bmatrix} X_{2i, 2j}^{l+1} & X_{2i, 2j+1}^{l+1} \\ X_{2i+1, 2j}^{l+1} & X_{2i+1, 2j+1}^{l+1} \end{bmatrix} = X_{i,j}^l$$

For each block  $X_{i,j}^l$ , its corresponding ready-to-insert data (RID),  $S_{i,j}^l$ , can be obtained after the processes of MD5, XOR and RSA. Then, a payload block  $P_{i,j}^l$  is constructed based on the lowest level block  $X_{i,j}^L$ . For each payload block, it contains both RID and the data that belong to the higher-level block of  $X_{i,j}^L$ .



**Figure 2.6** An example of the hierarchy approach with three levels.

Figure 2.6 illustrates an example of the hierarchy approach. Figure 2.6(a) denotes an original image  $X$  and its three-level hierarchical results. Note that,  $X_{0,0}^1$  is the top level of the hierarchy consisting of only one block  $X$ . Figure 2.6(b) shows the corresponding RID of each block  $X_{i,j}^l$ . In Figure 2.6(c), when dealing with the block

$X_{3,3}^3$ , only the following three RID's,  $S_{3,3}^3$ ,  $S_{1,1}^2$ , and  $S_{0,0}^1$ , are considered. The generated payload block  $P_{3,3}^3$  is embedded into LSB of the original image  $X$  on  $X_{3,3}^3$ .

### 2.2.5 The Morphological Approach for Extracting Pixel-Based Features

Mathematical morphology, which is based on set-theoretic concept, can extract object features by choosing a suitable structuring shape as a probe. The analysis is geometric in character and it approaches image processing from the vantage point of human perception. The morphological operators deal with two images. The image being processed is referred to as the active image, and the other image being a kernel is referred to as the structuring element. With the help of various designed structuring shapes, one can simplify image data representation and explore shape characteristics.

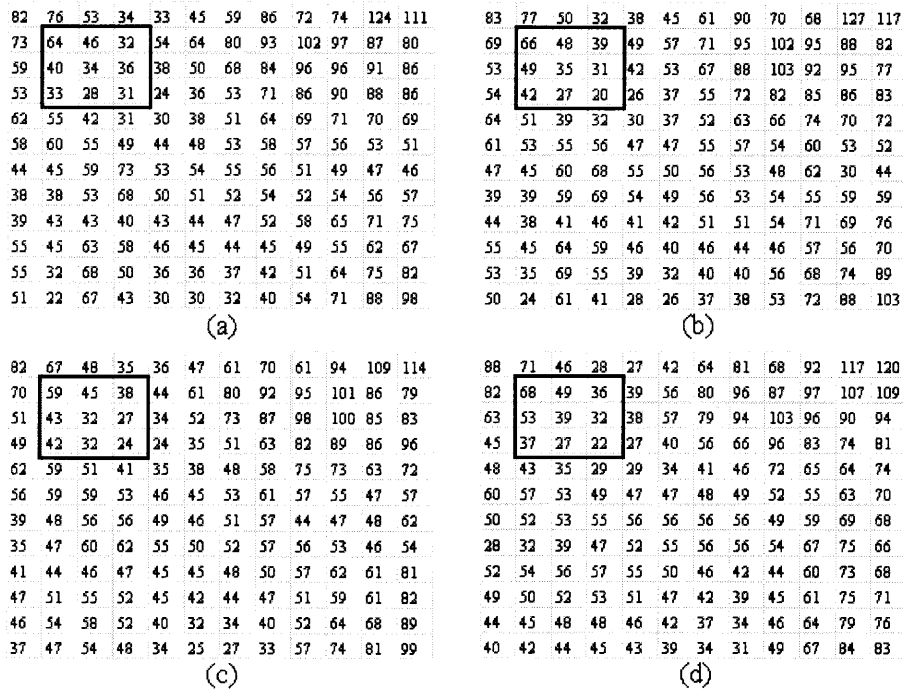
In order to extract the *pixel-based* (PB) features of an image, mathematical morphology is adopted. Let  $S$  be a grayscale structuring element. Let  $H^{Di}$  and  $H^{Er}$  be the dilation and the erosion of  $H$  by  $S$ , respectively. Let  $PB(i, j)$ ,  $H(i, j)$ ,  $H^{Di}(i, j)$  and  $H^{Er}(i, j)$  respectively denote the grayscale value of  $PB$ ,  $H$ ,  $H^{Di}$  and  $H^{Er}$  at pixel  $(i, j)$ . The pixel-based features extraction algorithm is presented below.

#### The Pixel-Based Features Extraction Algorithm:

1. Design the grayscale structuring element  $S$ .
2. Obtain  $H^{Di}$  by a dilation of  $H$  by  $S$ .
3. Obtain  $H^{Er}$  by an erosion of  $H$  by  $S$ .
4. Obtain the pixel-based features by the following rule:

If  $(0 \leq \frac{H(i,j) - H^{Er}(i,j)}{H^{Di}(i,j) - H^{Er}(i,j)} \leq T_1)$  or  $(T_2 \leq \frac{H(i,j) - H^{Er}(i,j)}{H^{Di}(i,j) - H^{Er}(i,j)} \leq 1)$ , then  $PB(i,j) = 1$ ;

otherwise,  $PB(i,j) = 0$ . Note that  $T_1$  and  $T_2$  are threshold values where  $T_1 < T_2$ .



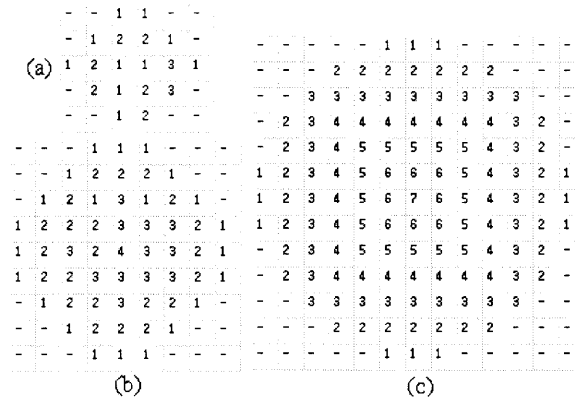
**Figure 2.7** An example of illustrating the property of pixel-based features.

**Table 2.1** Position of a Pixel and Its Neighborhood in Different Compression rates

Figure	Pixel Value	Sorted Sequence	Order
<b>3 (a)</b>	34	28,31,32,33, <b>34</b> ,36,40,46,64	5
<b>3 (b)</b>	35	20,27,31, <b>35</b> ,39,42,48,49,66	4
<b>3 (c)</b>	32	24,27,32, <b>32</b> ,38,42,43,45,59	4
<b>3 (d)</b>	39	22,27,32,36,37, <b>39</b> ,49,53,68	6

It is obviously that almost all the pixel-based features of an image will remain the same after compression. Figure 2.7 illustrates the property of pixel-based features. Figure 2.7(a) is the original image and Figures 2.7(b), (c) and (d) are the compressed images in the compression rates of 80%, 50% and 20%, respectively. By randomly selecting a pixel  $p$  and a  $3 \times 3$  neighborhood as shown in the bold box, the value of  $p$  (i.e., the central pixel)

is originally ranked “5” and is now ranked “4” or “6” after compression as shown in Table 2.1. Using the PB features extraction algorithm, the pixel  $p$  will all receive “0” in the original and compressed images.

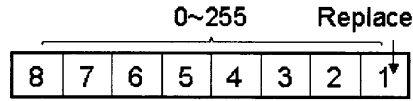


**Figure 2.8** The circular GMSEs.

Rotation is one of the techniques to attack watermarking. In order to extract the pixel-based features accurately and avoid rotational attack, the *grayscale morphological structuring element* (GMSE) is designed to be circular. Figure 2.8 shows the circular GMSEs of different sizes. Apparently, the larger the GMSE is, the more circular it is. Note that, the symbol “-” indicates “don’t care” that is equivalent to placing “ $-\infty$ ” in the GMSE.

### 2.2.6 Least Signification Bit (LSB) Substitution

LSB substitution is often used in image watermarking for embedding watermarks into a host image. The idea of LSB substitution is to replace the least significant bits of pixel value the watermarks. For example, a grayscale value, with range from 0 to 255, can be represented by 8 bits as shown in Figure 2.9. Sometime, in order to embed more watermarks, the least  $n$  bits are replaced.



**Figure 2.9** LSB substitution.

### 2.2.7 Discrete Cosine Transformation (DCT)

DCT is a transformation that translates an image from spatial domain to its frequency domain. It helps separate an image into parts (or spectral sub-bands) of differing importance (with respect to the image's visual quality).

The general equation for a 2D DCT is defined below:

**Spatial to frequency:**

$$F(u, v) = C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos\left(\frac{(2x+1)u\pi}{2N}\right) \cos\left(\frac{(2y+1)v\pi}{2N}\right)$$

**Frequency to spatial:**

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)F(u, v) \cos\left(\frac{(2x+1)u\pi}{2N}\right) \cos\left(\frac{(2y+1)v\pi}{2N}\right)$$

where  $C(u) = C(v) = \sqrt{\frac{1}{N}}$ , if  $u = v = 0$ . Otherwise,  $C(u) = C(v) = \sqrt{\frac{2}{N}}$ .

A more convenient method for expressing the 2D DCT is with matrix products as  $F = MfM^T$ , and its inverse DCT is  $f = M^T FM$  where  $F$  and  $f$  are  $8 \times 8$  data matrices, and  $M$  is the matrix as shown below:

$$M = \begin{bmatrix} \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ \frac{1}{2} \cos \frac{1}{16} \pi & \frac{1}{2} \cos \frac{3}{16} \pi & \frac{1}{2} \cos \frac{5}{16} \pi & \frac{1}{2} \cos \frac{7}{16} \pi & \frac{1}{2} \cos \frac{9}{16} \pi & \frac{1}{2} \cos \frac{11}{16} \pi & \frac{1}{2} \cos \frac{13}{16} \pi & \frac{1}{2} \cos \frac{15}{16} \pi \\ \frac{1}{2} \cos \frac{2}{16} \pi & \frac{1}{2} \cos \frac{6}{16} \pi & \frac{1}{2} \cos \frac{10}{16} \pi & \frac{1}{2} \cos \frac{14}{16} \pi & \frac{1}{2} \cos \frac{18}{16} \pi & \frac{1}{2} \cos \frac{22}{16} \pi & \frac{1}{2} \cos \frac{26}{16} \pi & \frac{1}{2} \cos \frac{30}{16} \pi \\ \frac{1}{2} \cos \frac{3}{16} \pi & \frac{1}{2} \cos \frac{9}{16} \pi & \frac{1}{2} \cos \frac{15}{16} \pi & \frac{1}{2} \cos \frac{21}{16} \pi & \frac{1}{2} \cos \frac{27}{16} \pi & \frac{1}{2} \cos \frac{33}{16} \pi & \frac{1}{2} \cos \frac{39}{16} \pi & \frac{1}{2} \cos \frac{45}{16} \pi \\ \frac{1}{2} \cos \frac{4}{16} \pi & \frac{1}{2} \cos \frac{12}{16} \pi & \frac{1}{2} \cos \frac{20}{16} \pi & \frac{1}{2} \cos \frac{28}{16} \pi & \frac{1}{2} \cos \frac{36}{16} \pi & \frac{1}{2} \cos \frac{44}{16} \pi & \frac{1}{2} \cos \frac{52}{16} \pi & \frac{1}{2} \cos \frac{60}{16} \pi \\ \frac{1}{2} \cos \frac{5}{16} \pi & \frac{1}{2} \cos \frac{15}{16} \pi & \frac{1}{2} \cos \frac{25}{16} \pi & \frac{1}{2} \cos \frac{35}{16} \pi & \frac{1}{2} \cos \frac{45}{16} \pi & \frac{1}{2} \cos \frac{55}{16} \pi & \frac{1}{2} \cos \frac{65}{16} \pi & \frac{1}{2} \cos \frac{75}{16} \pi \\ \frac{1}{2} \cos \frac{6}{16} \pi & \frac{1}{2} \cos \frac{18}{16} \pi & \frac{1}{2} \cos \frac{30}{16} \pi & \frac{1}{2} \cos \frac{42}{16} \pi & \frac{1}{2} \cos \frac{54}{16} \pi & \frac{1}{2} \cos \frac{66}{16} \pi & \frac{1}{2} \cos \frac{78}{16} \pi & \frac{1}{2} \cos \frac{90}{16} \pi \\ \frac{1}{2} \cos \frac{7}{16} \pi & \frac{1}{2} \cos \frac{21}{16} \pi & \frac{1}{2} \cos \frac{35}{16} \pi & \frac{1}{2} \cos \frac{49}{16} \pi & \frac{1}{2} \cos \frac{63}{16} \pi & \frac{1}{2} \cos \frac{77}{16} \pi & \frac{1}{2} \cos \frac{91}{16} \pi & \frac{1}{2} \cos \frac{105}{16} \pi \end{bmatrix}$$

### 2.2.8 Discrete Wavelet Transformation (DWT)

DWT is also a simple and fast transformation approach that translates an image from spatial domain to its frequency domain. Unlike the standard Fourier transformation which represents a signal either in spatial domain or in frequency domain, DWT is able to provide a representation for both spatial and frequency interpretations. Now it becomes much more popular for some areas such as JPEG 2000 compression.

The Haar DWT [26, 27, 32, 35] is the simplest of all wavelets and is adopted in this dissertation because of its features as shown below:

- (a) Haar wavelets are orthogonal, symmetric and the simplest.
- (b) The minimum support property allows arbitrary grid intervals.
- (c) Boundary conditions are easier than other wavelet-based methods.
- (d) Haar DWT works very well to detect the characteristics like edges and corners.

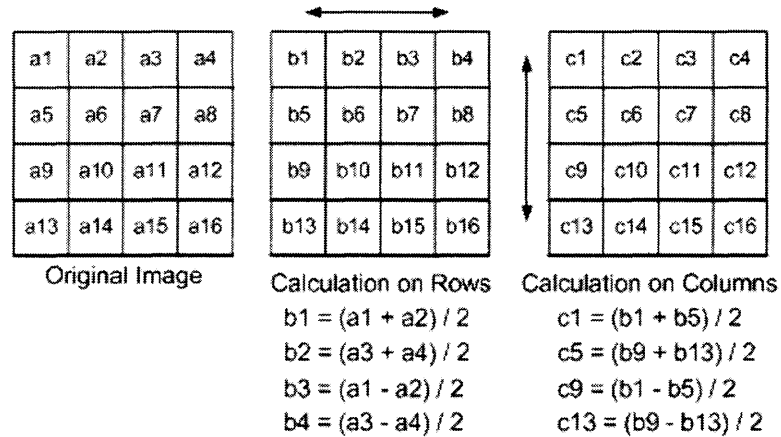


Figure 2.10 An example of Haar DWT.

The idea for performing Haar DWT is quite simple. Suppose the original size of input image is  $N \times M$ . At the first filtering in horizontal direction and down-sampling procedure, the size of images will be reduced to  $N \times \frac{M}{2}$ . And further filtering and down-sampling in vertical direction, four images of size  $\frac{N}{2} \times \frac{M}{2}$  can be obtained.

Figure 2.10 shows an example of DWT with size of  $4 \times 4$ .

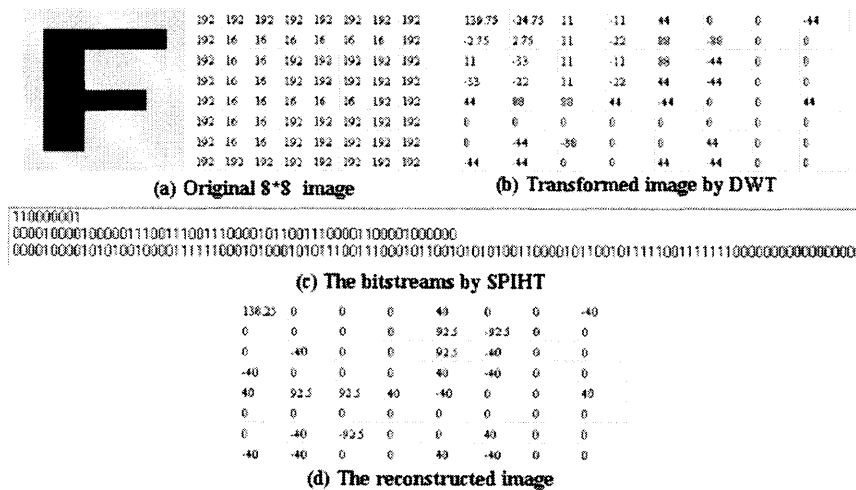


Figure 2.11 The example of SPIHT.

### 2.2.9 Set Partitioning in Hierarchical Tree (SPIHT)

The SPIHT is a zerotree structure based on DWT. Figure 2.11 shows an example of SPIHT. Figure 2.11(a) is the original  $8 \times 8$  image. the transformed image is obtained as shown in Figure 2.11(b) by DWT. Figure 2.11(c) lists the bitstreams generated by applying SPIHT three times. the image is reconstructed by using these bitstreams and the result is shown in Figure 2.11(d).

### 2.2.10 Chaotic Map

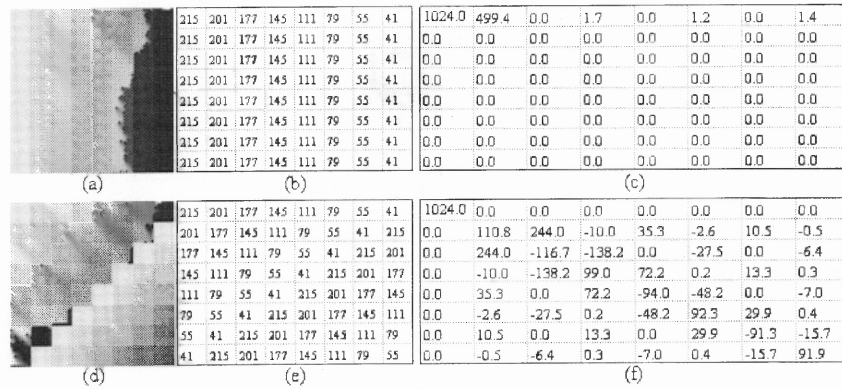
Figure 2.12 shows an example of enlarging the significant coefficients by rearranging the locations of an image. Figures 2.12(a) and (b) respectively show the original images in pictorial and text formats, and Figures 2.12(d) and (e) are the relocated results of Figures 2.12(a) and (b), respectively. Figures 2.12(c) and (f) are obtained from Figures 2.12(b) and (e) by DCT. It is obvious that there are only 5 non-zero coefficients containing two significant coefficients in Figure 2.12(c), but 44 non-zero coefficients containing 14 of significant coefficients (greater than 70) in Figure 2.12(f).

In recent years, chaotic maps [100, 110] have been widely used for digital watermarking to increase the security. The chaotic maps can be considered as a tool to relocate the pixels of an image. Voyatzis and Pitas [100] presented a well-known chaotic system called ‘‘Toral Automorphism’’ to a squared image, which first rearranges the locations of an image, and embeds the watermarks into the spatial domain of the relocated image. The equation for performing the chaotic map is described below

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ l & l+1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \pmod{N} \quad (2.1)$$

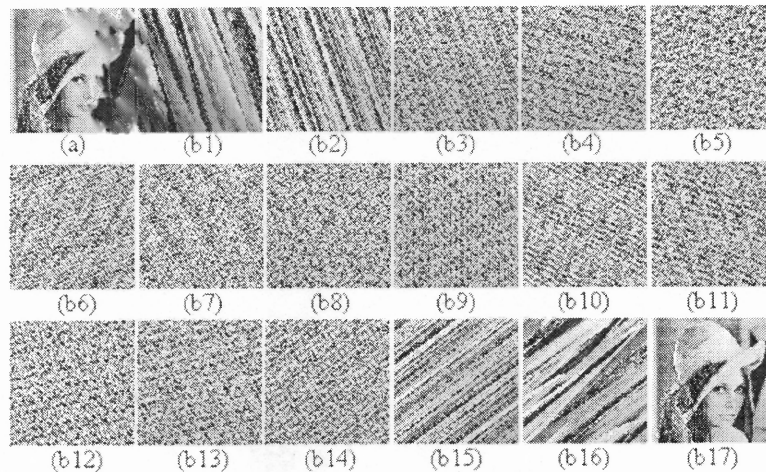
where  $\det\left(\begin{bmatrix} 1 & 1 \\ l & l+1 \end{bmatrix}\right) = 1 \text{ or } -1$ , and  $N$  is the width of an squared image.





**Figure 2.12** An example of enlarging the significant coefficients.

Figure 2.13 shows an example of performing the chaotic map. Figure 2.13(a) is the original image of size  $101 \times 101$  where  $l = 2$  and  $N = 101$ . Figures 2.13(b1)-(b17) are the results after performing the chaotic map by Equation (2.1). It is obvious that the reconstructed image will be same to the original image after 17 iterations. Note that, each new relocated image is performed based on the previous result. For example, Figure 2.13(b1) is first obtained from Figure 2.13(a) by Equation (2.1). Then, Figure 2.13(b2) is obtained from Figure 2.13(b1).



**Figure 2.13** An example of performing the chaotic map.

### 2.2.11 GA-Based Watermarking

After embedding watermarks into the frequency domain of a host image, the inverse transform has to be performed to translate the image from its frequency domain to spatial domain. However, the pixel values of the transformed image must be real numbers. That is, the data embedded in the coefficients of a transformed image will be somewhat disturbed in the process of transforming the image from its frequency domain into its spatial domain because of deviations in converting real numbers into integers in the spatial domain. Since it is difficult to predict what the impact will be happened in the frequency domain of a host image if some values of pixels in the spatial domain of the host image are changed, how to correct the rounding errors becomes a difficult task. Therefore, the GA-based watermarking is developed to solve the problem

$\xi_0$	$\xi_1$	$\xi_2$	$\xi_3$	$\xi_4$	$\xi_5$	$\xi_6$	$\xi_7$	0	0	1	0	0	2	0	1
$\xi_8$	$\xi_9$	$\xi_{10}$	$\xi_{11}$	$\xi_{12}$	$\xi_{13}$	$\xi_{14}$	$\xi_{15}$	1	2	0	0	1	1	0	1
$\xi_{16}$	$\xi_{17}$	$\xi_{18}$	$\xi_{19}$	$\xi_{20}$	$\xi_{21}$	$\xi_{22}$	$\xi_{23}$	2	1	1	0	0	-2	0	1
$\xi_{24}$	$\xi_{25}$	$\xi_{26}$	$\xi_{27}$	$\xi_{28}$	$\xi_{29}$	$\xi_{30}$	$\xi_{31}$	3	1	2	5	0	0	0	0
$\xi_{32}$	$\xi_{33}$	$\xi_{34}$	$\xi_{35}$	$\xi_{36}$	$\xi_{37}$	$\xi_{38}$	$\xi_{39}$	0	2	0	0	0	1	0	0
$\xi_{40}$	$\xi_{41}$	$\xi_{42}$	$\xi_{43}$	$\xi_{44}$	$\xi_{45}$	$\xi_{46}$	$\xi_{47}$	0	-1	0	2	1	1	4	2
$\xi_{48}$	$\xi_{49}$	$\xi_{50}$	$\xi_{51}$	$\xi_{52}$	$\xi_{53}$	$\xi_{54}$	$\xi_{55}$	1	1	1	0	0	0	0	0
$\xi_{56}$	$\xi_{57}$	$\xi_{58}$	$\xi_{59}$	$\xi_{60}$	$\xi_{61}$	$\xi_{62}$	$\xi_{63}$	0	0	0	0	1	-2	1	3

(a)

(b)

**Figure 2.14** The numbering positions corresponding to 64 genes.

In order to apply the genetic algorithm for embedding messages into the frequency domain of a cover-image to obtain the stego-image, the chromosome,  $\xi$ , consisting of  $n$  genes as  $\xi = g_0, g_1, g_2, \dots, g_n$  is utilized. Figure 2.14 gives an example of a chromosome ( $\xi \in Z^{64}$ ) containing 64 genes ( $g_i \in Z$  (integers)). Figure 2.14(a) shows the distribution order of a chromosome in an  $8 \times 8$  block, and Figure 2.14(b) shows an example of the corresponding chromosome.

### **Fitness function:**

Two fitness functions used for GAs are defined below.

#### **A. The First Fitness Function:**

$$Evaluation\_1(\xi) = \sum_{i=0}^{all\ pixels} | Watermark_i^\alpha - Watermark_i^\beta |,$$

where  $\xi$  is a chromosome, and  $Watermark^\alpha$  and  $Watermark^\beta$  are the embedded and extracted watermarks, respectively. The first fitness function is used to calculate the differences between the two watermarks. Based on the first fitness function, if  $Evaluation\_1(\xi)$  is less than the threshold value, the chromosome can be selected as the next population. For example, if they are the same,  $Evaluation\_1(\xi) = 0$ .

#### **B. The Second Fitness Function:**

$$Evaluation\_2(\xi) = \sum_{i=0}^{all\ pixels} | Image_i^\alpha - Image_i^\beta |,$$

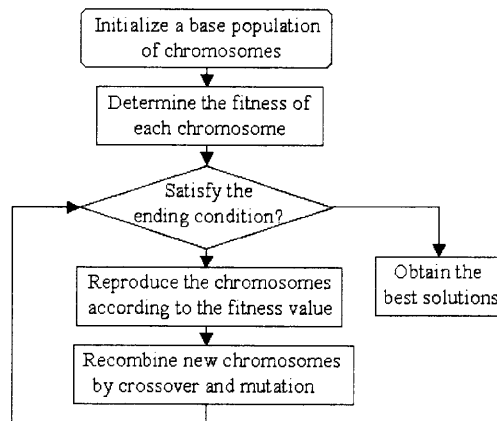
where  $Image^\alpha$  and  $Image^\beta$  are the original and the watermarked images, respectively. The second fitness function is used to calculate the differences between the two images. Based on the second fitness function, if  $Evaluation\_2(\xi)$  is less than the threshold value, the chromosome can be selected as the next population.

The GA-based watermarking is presented below and its flowchart is shown in Figure 2.15.

#### **Algorithm:**

1. Define the fitness function, numbers of genes, sizes of population, crossover rate, critical value and mutation rate.
2. Generate the first generation by random selection.

3. Translate real numbers into integers based on each corresponding chromosome.
4. Evaluate the fitness value for each corresponding chromosome.
5. Obtain the better chromosomes based on the fitness value.
6. Recombine new chromosomes by using crossover.
7. Recombine new chromosomes by using Mutation.
8. Repeat Steps 3 to 8 until a predefined condition is satisfied, or a constant number of iterations is reached.



**Figure 2.15** Basic steps in finding the best solutions by Genetic Algorithms.

### 2.3 The Combinational Image Watermarking

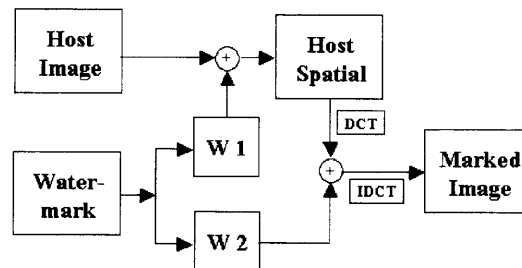
In order to provide more watermarks and to minimize the distortion of the watermarked image, a novel technique using the combinational spatial and frequency domains is presented in this section. The splitting of the watermark image into two parts respectively for spatial and frequency insertion relies on the user's preference and data importance. The proposed combinational image watermarking possesses the following advantages. More watermark data can be inserted into the host image, so that the capacity is increased. The splitting of the watermark into two parts makes the degree of protection double. The splitting strategy can be designed even more complicated to be unable to

compose. Furthermore, to enhance robustness, a random permutation of the watermark is used to defeat the attacks of signal processing, such as image crops.

### 2.3.1 The Algorithm of Combinational Image Watermarking

In order to insert more data into a host image, the simple way is to embed them in the spatial domain of the host image. However, the disadvantage is that the inserted data could be detectable by some simple extraction skills. How can people insert more signals but unperceivable? A new strategy of embedding large watermarks into a host image is developed by splitting the watermark image into two parts. One is embedded in the spatial domain of a host image, and the other in the frequency domain.

Let  $H$  be the original gray-level host image with size  $N \times N$  and  $W$  be the binary watermark image with size  $M \times M$ .  $W^1$  and  $W^2$  are the two separated watermarks from  $W$ .  $H^S$  is the image combined from  $H$  and  $W^1$  in the spatial domain.  $H^{DCT}$  is the image where  $H^S$  is transformed into the frequency domain by DCT.  $H^F$  is the image where  $H^{DCT}$  and  $W^2$  are combined in the frequency domain. Let  $\oplus$  denote the operation that substitutes bits of watermark for the specific bits of the host image in this chapter. For example, in order to avoid degrading the quality of the host image, the LSB is widely selected.



**Figure 2.16** The flowchart in combinational spatial and frequency domains.

The algorithm of the proposed combinational image watermarking is presented below. Its flowchart is shown in Figure 2.16.

**Algorithm:**

1. Separate watermark into two parts.

$$W = \{ w(i, j) , 0 \leq i, j < M \}, \text{ where } w(i, j) \in \{0,1\}$$

$$W^1 = \{ w^1(i, j) , 0 \leq i, j < M_1 \}, \text{ where } w^1(i, j) \in \{0,1\}$$

$$W^2 = \{ w^2(i, j) , 0 \leq i, j < M_2 \}, \text{ where } w^2(i, j) \in \{0,1\}$$

$$M = M_1 + M_2$$

2. Insert  $W^1$  into the spatial domain of  $H$  to obtain  $H^S$  as

$$H^S = \{ h^S(i, j) = h(i, j) \oplus w^1(i, j), 0 \leq i, j < N \}, \text{ where}$$

$h(i, j) , h^S(i, j) \in \{0,1,2,\dots,2^L - 1\}$  and  $L$  is the number of bits used as in the gray level of pixels.

3. Transform  $H^S$  by DCT to obtain  $H^{DCT}$ .

4. Insert  $W^2$  into the coefficients of  $H^{DCT}$  to obtain  $H^F$  as

$$H^F = \{ h^F(i, j) = h^{DCT}(i, j) \oplus w^2(i, j), 0 \leq i, j < N \}, \text{ where}$$

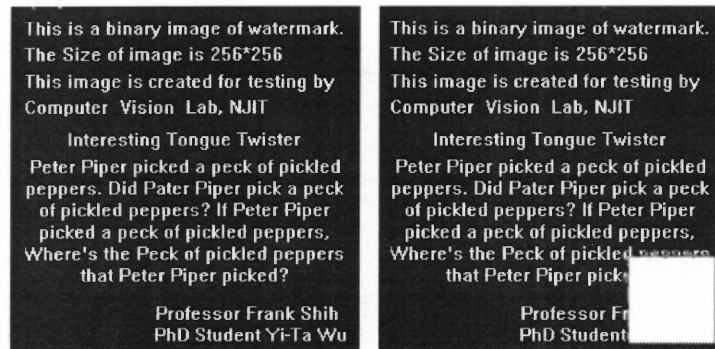
$$h^F(i, j) \in \{0,1,2,\dots,2^L - 1\}$$

5. Transform the embedded host image by Inverse DCT.

The criterion of splitting the watermark image into two parts, which are individually inserted into the input image in the spatial and frequency domains, depends on users and applications. In principle, the most important data exist in the center of the image. Therefore, a simply way of splitting is to select the central window in the watermark image to be inserted into the frequency domain. With the user's preference, the most private data can be selected to be inserted into the frequency domain.

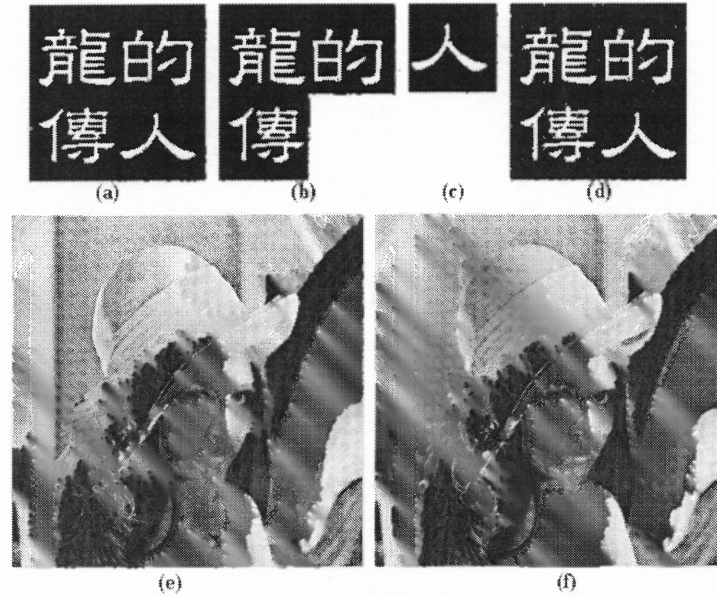
### 2.3.2 Experimental Result

Figure 2.17 illustrates that some parts of a watermark are important to be split for security purpose. For example, people cannot view who the writer is in the images. Therefore, it is embedded into the frequency domain and the rest into the spatial domain. In this way, not only the capacity is enlarged, but also secure the information that people are concerned with.



**Figure 2.17** A  $64 \times 64$  image is cut from a  $256 \times 256$  binary image.

Figure 2.18 demonstrates the embedding of a large watermark, a  $128 \times 128$  image, into a host image. Figure 2.18(a) is the original  $128 \times 128$  watermark image. Figures 2.18(b) and (c) are the two divided images from the original watermark, respectively. Figure 2.18(e) is obtained by embedding Figure 2.18(b) into the spatial domain of Lena image and obtain Figure 2.18(f) by embedding Figure 2.18(c) into the frequency domain of the watermarked Lena image in Figure 2.18(e). Figure 2.18(d) is the extracted watermark from Figure 2.18(f).



**Figure 2.18** The results when embedding a  $128 \times 128$  watermark into the Lena image.

The error measures used in this chapter are *Normalized Correlation* (NC) and *Peak Signal to Noise Ratio* (PSNR). They are defined as follows, where the symbols are defined in the previous sections.

$$NC = \frac{\sum_{i=1}^N \sum_{j=1}^N H(i, j) * H^F(i, j)}{\sum_{i=1}^N \sum_{j=1}^N [H(i, j)]^2},$$

$$PSNR = 10 \times \log_{10} \left( \frac{\sum_{i=1}^N \sum_{j=1}^N [255]^2}{\sum_{i=1}^N \sum_{j=1}^N [h(i, j) - h^*(i, j)]^2} \right).$$

Table 2.2 presents the results of embedding different sizes of watermark into a host image. PSNR in the first step means comparing both the original and the embedded Lena images in the spatial domain. PSNR in the second step means comparing both



images in the frequency domain. NC means the normalized correlation by comparing both the original and the extracted watermarks.

**Table 2.2** Comparisons When Embedding Different Sized Watermarks

	<b>64×64</b>	<b>128×128</b>	<b>256×256</b>
<b>PSNR in First Step</b>	None	56.58	51.14
<b>PSNR in Second Step</b>	64.57	55.93	50.98
<b>NC</b>	1	0.9813	0.9644

## 2.4 GA-Based Watermarking Technique

A watermark hidden in an image is retrieved differently from the original watermark due to the frequently-used rounding approach. The simple rounding will cause numerous errors in the embedded watermark especially when it is large. The GA-based watermarking is presented in this section to correct the rounding errors. Furthermore, it can also be adopted to both fragile and robust watermarking techniques.

### 2.4.1 The Errors Caused by Deviations in Translating Real Numbers into Integers

Figure 2.19 shows the errors caused by using the round technique in translating real numbers into integers. Figure 2.19(a) is the original host image, an  $8 \times 8$  gray-level image, in the spatial domain and Figure 2.19(b) is the transformed image of Figure 2.19(a) by DCT. Figure 2.19(c) is a binary watermark, in which “0” and “1” denote the embedded value in its location; the minus sign “-” indicates no change in its position. Figure 2.19(d) is obtained by embedding Figure 2.19(c) into Figure 2.19(b) based on LSB modification. Note that, the watermark is embedded into the integer part of absolute real number. After transforming Figure 2.19(d) into its spatial domain by IDCT, an image as shown in Figure 2.19(e) is obtained where all pixels are real numbers. Figure 2.19(f) is generated

after rounding the real numbers into integers. Figure 2.19(f) is the watermarked image and will be used for some applications. The method for extracting the watermarks is quite simple. If someone wants to check the watermarks from the watermarked image, he/she just needs to extract the watermarks from the coefficients of the frequency domain of the watermarked image. Therefore, Figure 2.19(g) is obtained by performing DCT on Figure 2.19(f). Finally, the embedded watermark is obtained as shown in Figure 2.19(h) by extracting data from the specific positions in Figure 2.19(g). For example, the integer part of an absolute value “-46.94” is “46”. The binary format “00101110” is generated by translating the decimal value “46” into a binary format. Then, the LSB of “00101110” is “0.” Finally, the extracted watermarks (0,0,1,0) can be obtained from (-46.94,30.87,-5.16,94.24). By comparing the embedded (1101) and extracted (0010) watermarks, the terrible result is happened that the watermark is totally different. In other words, a big mistake of losing the original watermark is made.

Table 2.3 shows the results when embedding different kinds of watermark into the same transformed image in Figure 2.19(d). The embedded rule is shown in Figure 2.20. That is, for a four bits watermark “1234,” the bit “1” is embedded into position A, “2” into position B, “3” into position C, and “4” into “D.” There are  $2^4$  possible embedded watermarks. Only two cases can be extracted correctly.

**Table 2.3** Total Possible Embedded Watermarks

Embedded	Extracted	Error Bits	Embedded	Extracted	Error Bits
0000	0000	0	1000	0000	1
0001	0000	1	1001	0000	2
0010	0000	1	1010	0000	2
0011	0000	2	1011	0000	3
0100	0000	1	1100	0000	2
0101	0000	2	1101	0010	4
0110	0000	2	1110	1110	0
0111	1110	2	1111	1110	1

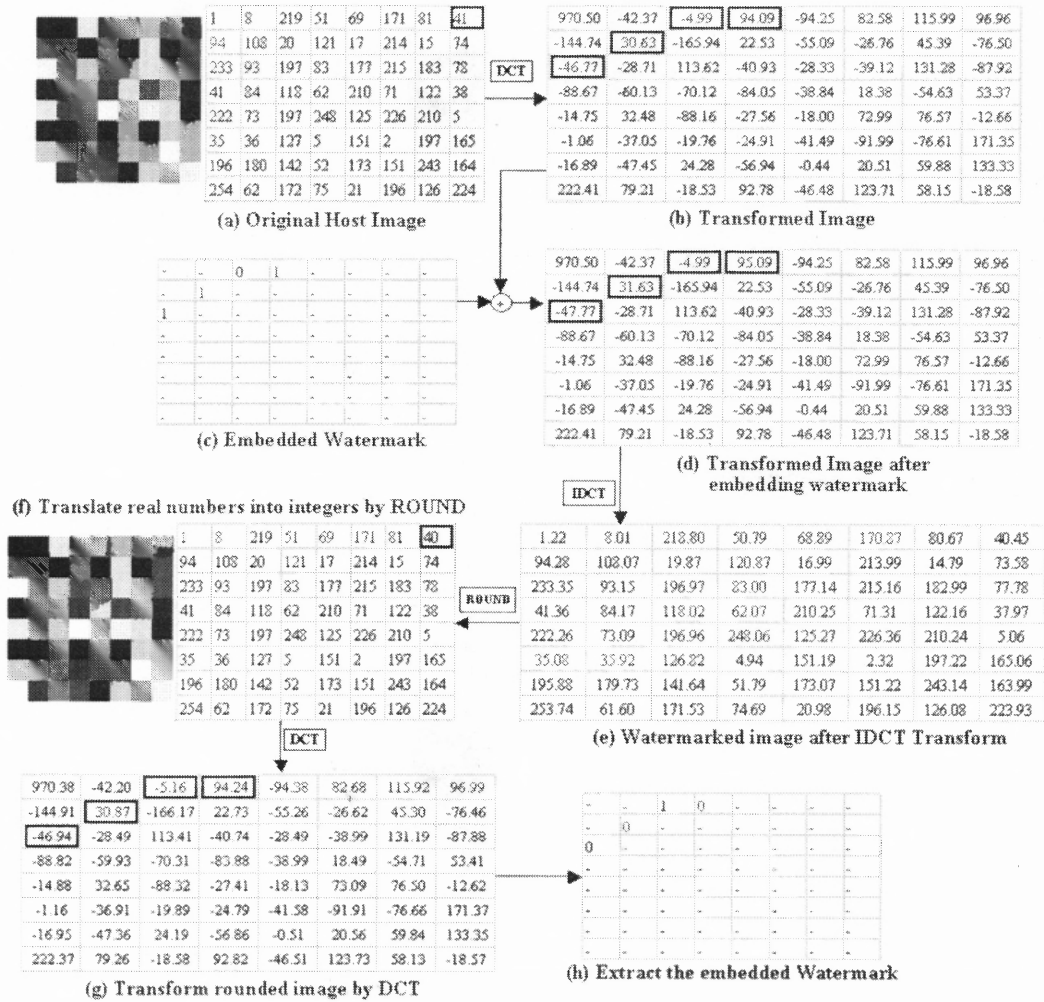


Figure 2.19 The errors caused by the Round technique.

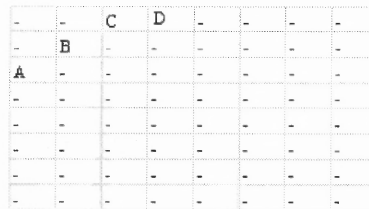


Figure 2.20 The positions where watermark is embedded.

### 2.4.2 The GA-Based Watermarking for Rounding Error Correction

Before presenting the basic concept of reducing errors by genetic algorithms, two primary issues have to be noted:

- (1) The embedded data should be accurately extracted.
- (2) The changes in order to modulate the errors should be minimal.

Note that the coefficients in the frequency domain will be changed dramatically even though only one pixel in the spatial domain is changed. Moreover, the changes of pixels in the frequency domain are difficult to derive intuitively. Therefore, it is unable to know whether the embedded data are variant or not in the process of translating real numbers into integers.

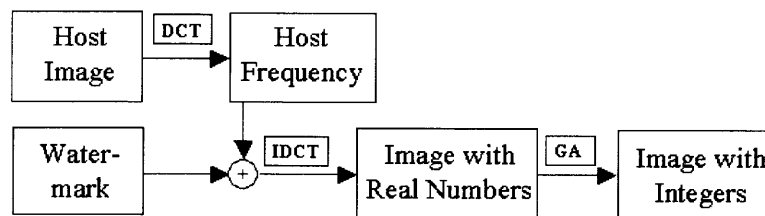
On the other hand, how to determine the proper pixels should be changed in order to correct the errors and then obtain the exact embedded data? As mentioned above, it is not only difficult to modulate intuitively, but also probable to cause the worst result by random modulation.

In order to solve above problems, the genetic algorithm is utilized to find the suitable solution in translating the real numbers into the integers. That is, the novel method not only restores the embedded data exactly, but also changes the least pixels in the host image.

Let  $H$  be the original gray-level host image with size  $N \times N$  and  $W$  be the binary watermark image with size  $M \times M$ .  $H^{DCT}$  is the image where  $H$  is transformed into the frequency domain by DCT.  $H^{WF}$  is the watermarked image where  $H^{DCT}$  and  $W$  are combined in the frequency domain.  $H^{WRS}$  is the watermarked real-number image where  $H^{WF}$  is transformed into the spatial domain by *Inverse Discrete Cosine*

*Transform* (IDCT).  $H^{GA}$  is the watermarked integer image where all real numbers in  $H^{WRS}$  are translated into integers by genetic algorithms. Let  $\oplus$  denote the operation that substitutes bits of a watermark for bits of a host image based on LSB substitution.

Note that although the LSB substitution avoids degrading the image appearance the least, it can be easily affected by noise. To reduce the effect of noise, the higher significant bits substitution can be used. However, in this way the quality of the watermarked image is degraded significantly and can be simply observed by human eyes.



**Figure 2.21** The flowchart in reducing the errors by using Genetic Algorithm.

The algorithm of the proposed watermarking is presented below. Its flowchart is shown in Figure 2.21. Let  $R$  denote the set of real numbers.

**Algorithm:**

1. Transform the host image  $H$  by DCT to obtain  $H^{DCT}$ .

$H = \{ h(i, j), 0 \leq i, j < N \}$ , where  $h(i, j) \in \{0, 1, 2, \dots, 2^L - 1\}$  and  $L$  is the number of bits used to represent gray level of pixels. For example,  $L=8$  for an image with 256 gray levels.

$$H^{DCT} = \{ h^{DCT}(i, j), 0 \leq i, j < N \}, \text{ where } h^{DCT}(i, j) \in R.$$

2. Insert  $W$  into the coefficients of  $H^{DCT}$  to obtain  $H^{WF}$

$$W = \{ w(i, j), 0 \leq i, j < M \}, \text{ where } w(i, j) \in \{0, 1\}$$

$$H^{WF} = \{ h^{WF}(i, j) = h^{DCT}(i, j) \oplus w(i, j), 0 \leq i, j < N \}, \text{ where } h^{WF}(i, j) \in R.$$

3. Transform  $H^{WF}$  by IDCT to obtain  $H^{WRS}$ .  
 $H^{WRS} = \{ h^{WRS}(i, j), 0 \leq i, j < N \}$ , where  $h^{WRS}(i, j) \in R$ .
4. Find the suitable solution to translate all real numbers in  $H^{WRS}$  into integers, and obtain  $H^{GA} = \{ h^{GA}(i, j), 0 \leq i, j < N \}$ , where  $h^{GA}(i, j) \in \{0, 1, 2, \dots, 2^L - 1\}$ .

The criterion to determine the suitable policy in translating real numbers into integers by genetic algorithms depends on the size of the embedded data. That is, the more data embedded in the coefficients of the frequency domain, the more difficult it can find the suitable solution to restore the errors. In order to apply genetic algorithms to solve this problem, a chromosome  $G_1$  consisting of 64 genes is generated as

$$G_1 = 1101010100010001001110010110111010101010100101011100101101111000$$

and can be mapping into Figure 2.22(b).

0	1	2	3	4	5	6	7	1	1	0	1	0	1	0	1
8	9	10	11	12	13	14	15	0	0	0	1	0	0	0	1
16	17	18	19	20	21	22	23	0	0	1	1	1	0	0	1
24	25	26	27	28	29	30	31	0	1	1	0	1	1	1	0
32	33	34	35	36	37	38	39	1	0	1	0	1	0	1	0
40	41	42	43	44	45	46	47	1	0	0	1	0	1	0	1
48	49	50	51	52	53	54	55	1	1	0	0	1	0	1	1
56	57	58	59	60	61	62	63	0	1	1	1	1	0	0	0

(a)

(b)

**Figure 2.22** The positions corresponding to the Genes.

The usage of chromosomes in this section is different from the traditional methods. That is, it is not necessary to to decode each chromosome, the binary string, into a number for evaluating the fitness value. The chromosome is utilized to represent the policy in translating real numbers into integers. Let  $r$  be a real number. Therefore, an integer  $r^*$  can be determined by the following rules:

(1) If the signal is “1,”  $r^* = Trunc(r) + 1$ ,

(2) If the signal is “0,”  $r^* = Trunc(r)$ ,

where  $Trunc(r)$  denotes the integer part of  $r$ .

An example is illustrated in Figure 2.23. Figure 2.23(a) shows an image where all pixels are real numbers. Figure 2.23(b) is obtained by translating the real numbers into integers by using the chromosomes shown in Figure 2.23(b).

1.22	8.01	218.80	50.79	68.89	170.87	80.67	40.45	2	9	218	51	68	171	80	41
94.28	108.07	19.87	120.87	16.99	213.99	14.79	73.58	94	108	19	121	16	213	14	74
233.35	93.15	196.97	83.00	177.14	215.16	182.99	77.78	233	93	197	84	178	215	182	78
41.36	84.17	118.02	62.07	210.25	71.31	122.16	37.97	41	85	119	62	211	72	123	37
222.26	73.09	196.96	248.06	125.27	226.36	210.24	5.06	223	73	197	248	126	226	211	5
35.08	35.92	126.82	4.94	151.19	2.32	197.22	165.06	36	35	126	5	151	3	197	166
195.88	179.73	141.64	51.79	173.07	151.22	243.14	163.99	196	180	141	51	174	151	244	164
253.74	61.60	171.53	74.69	20.98	196.15	126.08	223.93	253	62	172	75	21	196	126	223

(a)

(b)

**Figure 2.23** The usage of the Genes.

### 2.4.3 Experimental Results

Figure 2.24 shows the result that corrects the errors by using GAs in translating real numbers into integers. Figure 2.24(a) is the original host image, an  $8 \times 8$  gray-level image, in the spatial domain and Figure 2.24(b) is the transformed image of Figure 2.24(a) by DCT. Figure 2.24(c) is a binary watermark, in which “0” and “1” denote the embedded data in its location; the minus sign “-” indicates no change in its position. Figure 2.24(d) is obtained by embedding Figure 2.24(c) into Figure 2.24(b) based on LSB modification. It is obviously there are three differences when comparing Figures 2.24(b) and (d), for instance, (-46.77 and -47.77), (30.63 and 31.63), and (94.09 and 95.09). After transforming Figure 2.24(d) into its spatial domain by IDCT, Figure 2.24(e) is obtained where all pixels are real numbers. Figure 2.24(f) is obtained by translating the real

numbers into integers by GAs. Figure 2.24(g) is the transformed image from Figure 2.24(f) by DCT. Finally, the exact embedded watermark is obtained by extracting the bits from the same position of embedding the watermark, as shown in Figure 2.24(h).

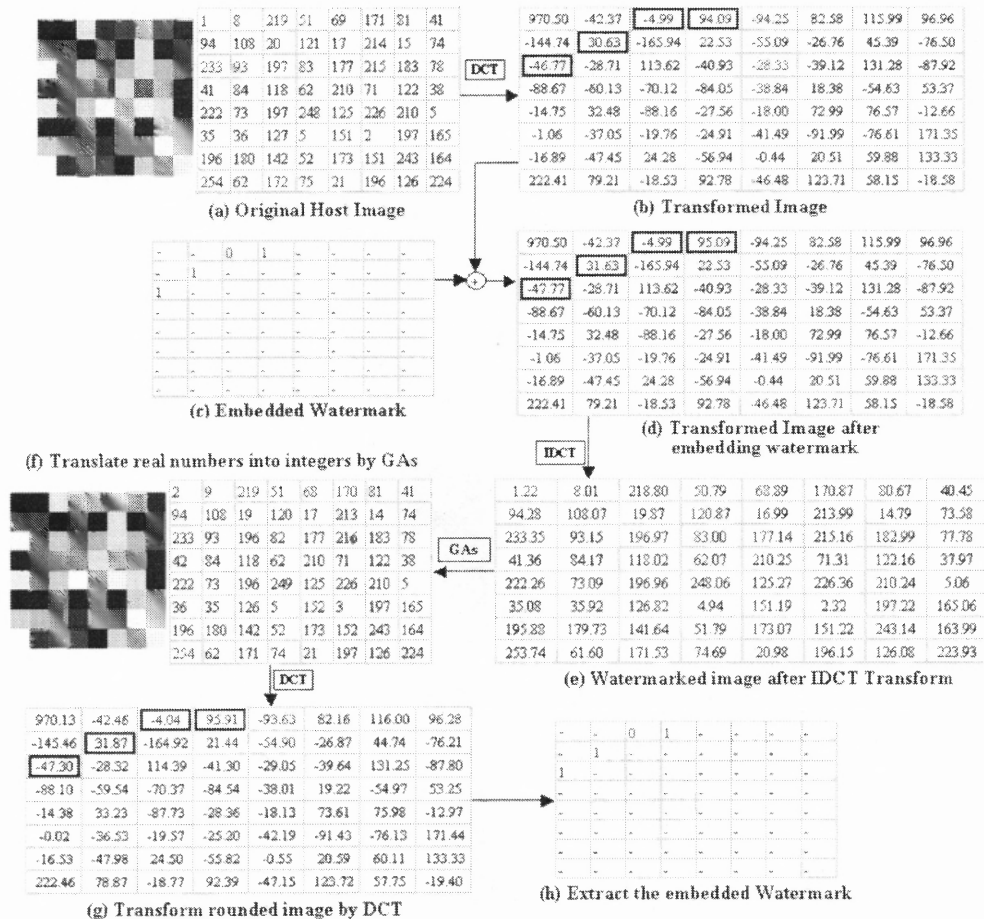


Figure 2.24 The Restoration by Genetic Algorithms.

#### 2.4.4 The Further Enhancement of GA-based Algorithm

In Figure 2.24, although numerous suitable solutions can be obtained by GAs to correct the errors, many of them are useless if considering the factor that the changes in the original image should be as small as possible. Therefore, the improved method for minimizing the changes is developed. In order to achieve the purpose of limited changes,





Figure 2.25 shows the result that not only corrects the errors, but also minimizes the differences between original and translated images. Figure 2.25(a) is the original host image, an  $8 \times 8$  gray-level image, in the spatial domain and Figure 2.25(b) is the transformed image of Figure 2.25(a) by DCT. Figures 2.25(c) and (e) are the translated results of Figure 2.24(d) by using Round and GAs, respectively. Figures 2.25(d) and (f) are the transformed images of Figures 2.25(c) and (e) by DCT, respectively. Figures 2.25(g) and (h) are the extracted data from Figure 2.25(d) and Figure 2.25(f), respectively. By comparing Figures 2.25(a) and (c), the only one change, from 41 to 40, causes the error that the expected data is not correct. After the best solution is found as shown in Figure 2.25(e), an additional change, from 17 to 16, corrects the errors.

In Figures 2.26(a) and (b), the embedded watermark is changed by the translating procedure from real numbers to integers. Therefore, the extracted data are the same as shown in Figures 2.25(c) and 2.26(c). In Figures 2.26(b) and (d), the two differences, (118, 119) and (126,125), correct the three errors of embedded data from 0000 to 1011.

As mentioned in Section 2.2.11, two fitness functions are adopted in the GA-based watermarking. Figure 2.27 is the example for illustrating the results by different strategies. Figure 2.27(a) is the original host image, the Lena image, of size  $256 \times 256$ . Figure 2.27(b) is the transformed image of Figure 2.27(a) by DCT. Figure 2.27(d) is the original binary watermark. Figure 2.27(d) is embedded into the coefficients of Figure 2.27(b). In order to obtain the IDCT image, as shown in Figure 2.27(c), the rounding and Gas approaches are adopted, respectively. Figure 2.27(e) is the extracted watermark from a round-based IDCT image. It is difficult to recognize the embedded watermark. However, if the real numbers is translated into integers by GAs, most of the

errors can be corrected, as shown in Figures 2.27(f) and (g). Figure 2.27(f) is the extracted watermark based on the GA-based watermarking which emphasizes on the second fitness function for higher PSNR. Similarly, Figure 2.27(g) is obtained by emphasizing on the first fitness function. Table 2.4 shows the different qualities of watermarked image “Lena” and extracted watermark based on different situations of translating policies: rounding, first , and second fitness functions.

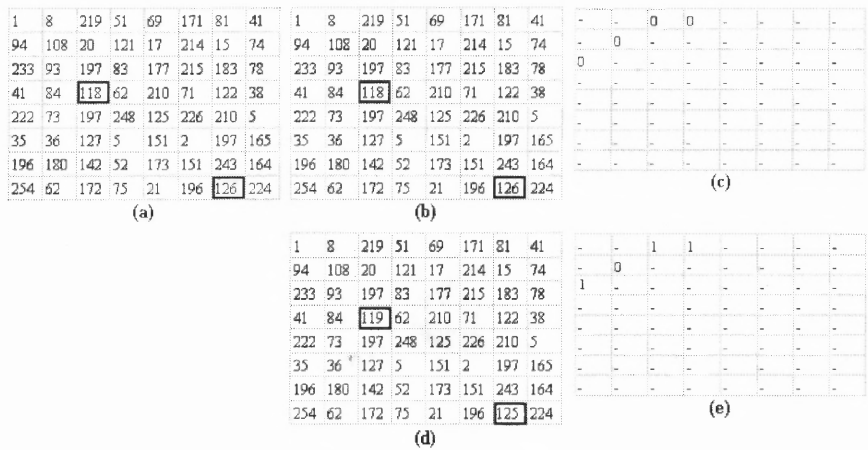


Figure 2.26 Solution in embedding 1011 by GAs.

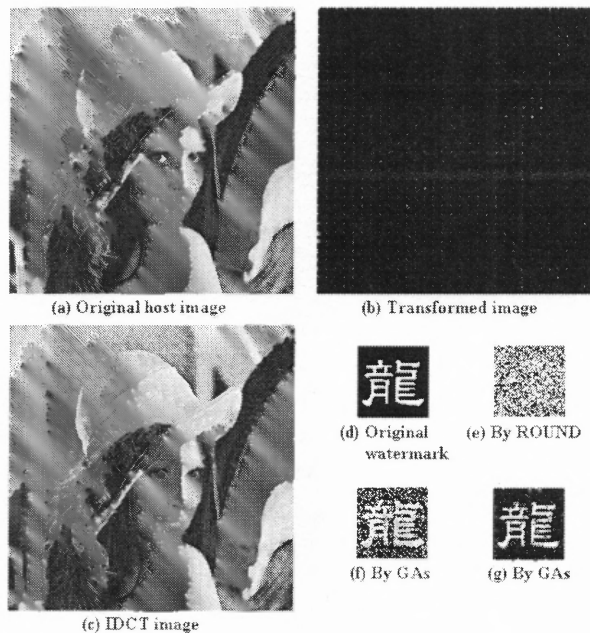


Figure 2.27 The example when embedding watermark into a real image.

**Table 2.4** Comparisons Between Round and GAs

	<b>By Round</b>	<b>By First GAs</b>	<b>By Second GAs</b>
PSNR	64.24	57.66	60.48
NC	0.545	0.951	0.845

The GA-based algorithm will stop if it satisfies the ending condition. That is, two threshold values are defined for the two fitness functions. If the fitness value is less than the threshold value, the algorithm will stop. Otherwise, the algorithm will continually run until it exceeds the maximum iteration setting. In the experiments, for an  $8 \times 8$  gray-level image, there are 64 pixels and  $2^{64} \cong 1.6 * 10^{19}$  possible permutations of the chromosome. By the GA-based watermarking, the average iterations when adopting the first and second fitness functions are 4000 and 28000, respectively. There are 10 chromosomes being considered in each iteration and the types of crossovers and mutations used. That is, one-point and two-point crossovers and mutations.

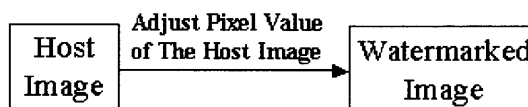
### 2.5 GA-Based Fragile Watermarking Technique

Several fragile watermarking techniques cannot detect vector quantization (VQ) attacks; the block-based approach, for example. Celik et al. [16] proposed a hierarchical watermarking approach to break the VQ attacks. In order to improve the hierarchical watermarking approach, a new fragile watermarking technique is presented based on genetic algorithms (GAs) to embed watermarks into the frequency domain of a host image. Different from existing schemes, the adjustment in the host image by GAs will achieve the embedding of fragile watermarks. The technique can provide a fundamental platform for other fragile watermarking techniques. Furthermore, the embedding of watermarks into frequency domain enhances the security in fragile watermarking.

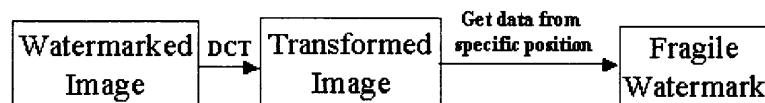
### 2.5.1 The Improved Scheme of the GA-Based Watermarking

As aforementioned, since existing block-based approaches perform the embedding of watermarks into LSB of the spatial domain, the watermarks can be extracted by some sophisticated calculations. Therefore, the embedding of watermarks into the frequency domain is a better way to enhance the security. On the other hand, in order to break the VQ counterfeiting attack, the hierarchical watermarking approach [16] is adopted in this section. That is, the payload block is obtained based on the method presented in Section 2.2.4 and embed it into the frequency domain of the host image.

In Section 2.4, the GAs-based watermarking is developed to correct the rounding errors. In this section, an improved scheme for embedding watermarks into the frequency domain of a host image is developed. The new scheme not only reduces the cost for obtaining the solution, but also offers more applications in watermarking. The main idea is to adjust pixels in the host image based on genetic algorithms, and make sure the extracted data from specific positions in the frequency domain of the host image are the same as the watermark. Figures 2.28 and 2.29 show the encoding and decoding procedures, respectively. The improved GA-based algorithm for embedding watermarks is presented below.



**Figure 2.28** The encoding procedure of the improved scheme.



**Figure 2.29** The decoding procedure of the improved scheme.

**The Improved GA-Based Algorithm:**

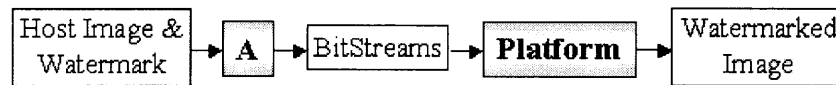
1. Define the fitness function, number of genes, size of population, crossover rate, and mutation rate.
2. Generate the first generation by random selection.
3. Adjust pixel values based on each chromosome of the generation.
4. Evaluate the fitness value for each corresponding chromosome.
5. Obtain the better chromosomes.
6. Recombine new chromosomes by crossover.
7. Recombine new chromosomes by mutation.
8. Repeat steps 3 to 8 until a predefined condition is satisfied or a constant number of iterations is reached.

**2.5.2 The Platform for the Existing Fragile Watermarking Technique**

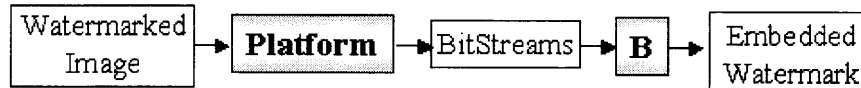
Basically, most existing fragile watermarking techniques can be considered as combining both the image and watermarks to generate a bit-stream by adopting cryptography techniques. The bit-stream is then embedded into LSB of the spatial domain. To improve the security, the GA-based algorithm can be used as a platform to embed the bit-stream into the frequency domain of a host image. Therefore, no matter what frequency-domain approaches are adopted, e.g., DCT, DWT or DFT, the GA-based algorithm can correct their rounding errors.

Figure 2.30 shows the flowchart of the watermarking insertion procedure by combining the GA-based watermarking with other fragile watermarking techniques. All strategies for encrypting watermarks can be categorized into Part A of Figure 2.31. For example, in Wong's algorithm, MD5, XOR, and RSA belong to Part A. After generating

the bit-stream, the GA-based watermarking is used to embed the bit-stream into the frequency domain.



**Figure 2.30** The insertion procedure of the GA-based watermarking platform.



**Figure 2.31** The extraction procedure of the GA-based watermarking platform.

The watermarking extraction procedure by combining the GA-based watermarking algorithm with other fragile watermarking techniques is shown in Figure 2.31. The bit-stream is extracted from specific positions of coefficients of the transformed image. Then, the embedded watermarks are reconstructed by the decryption strategies in Part B of Figure 2.31.

### 2.5.3 Experimental Results

Figure 2.32 shows an example of the GA-based algorithm. Figures 2.32(a) and (b) are the original image and the bit-stream, respectively. Using the embedded order shown in Figure 2.32(c), Figure 2.32(d) can be obtained by separating the bit-stream into 12 parts. Furthermore, a second fragile watermark is designed as shown in Figure 2.32(e) for higher security. The purpose is to adjust the pixel values in Figure 2.32(a) to obtain its frequency domain, so that the bit-stream and the second fragile watermark can be correctly extracted from specific positions. Here, the bit-stream and the second fragile watermark are extracted from bits 3, 4, 5 and bit 1, respectively. Figure 2.32(f) shows the

result of the adjusted image. The bit-stream and the second fragile watermark can be extracted from coefficients of the frequency domain of the watermarked image, as shown in Figure 2.32(g).

In general, the computational time is closely related to the amounts of the required embedded data. That is, the more the embedded data have, the more the computational time is. For example, it takes about 4 minutes on a Pentium III/866MHz and 256 megabytes RAM for obtaining the results in Figure 2.32 since there are 36 digits of a bitstream and 16 digits of the fragile watermark. Note that in the literature, usually only 4 digits of a bitstream and none of fragile watermark were used. Table 2.5 shows the bit-stream extracted from the specific positions of the coefficients.

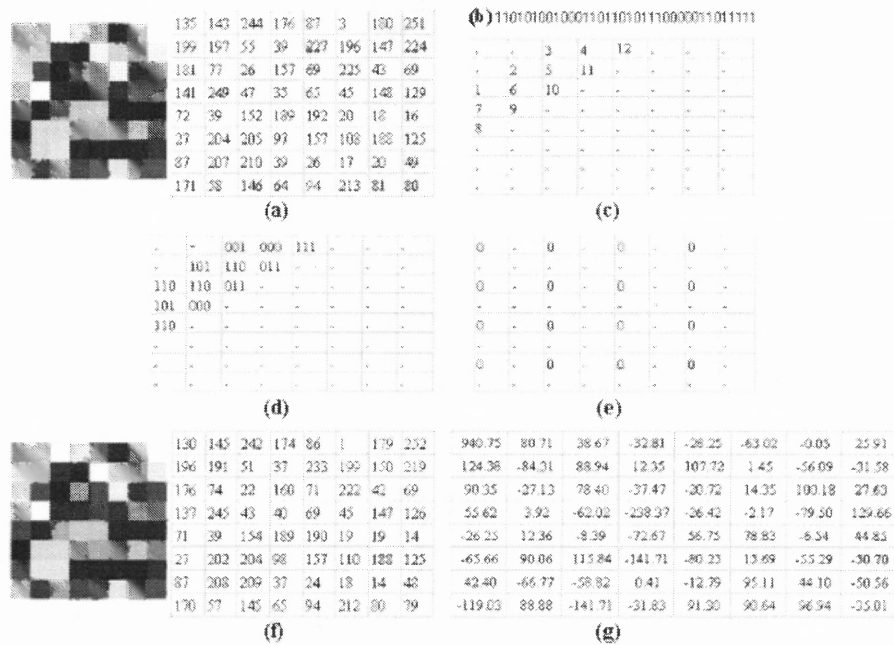
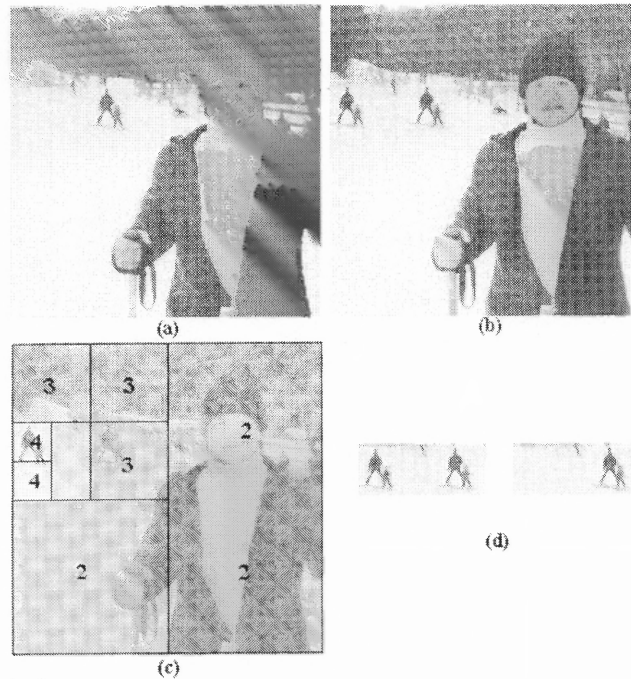


Figure 2.32 The example of the improved scheme of the GA-based watermarking.

Table 2.5 Bit-Stream Extracted from Specific Positions of the Binary Form

Decimal	Binary	Decimal	Binary	Decimal	Binary
90	01011010	88	01011000	3	00000011
84	01010100	27	00011011	78	01001110
38	00100110	55	00110111	12	00001100
32	00100000	26	00011010	28	00011100





**Figure 2.33** An example of the improved genetic algorithms.

Figure 2.33 shows an example of a real image. During embedding, the original image of size  $320 \times 320$  has been decomposed into a 4-level hierarchy of  $8 \times 8$  blocks at the lowest level. The size of each block in the lowest level is  $40 \times 40$ . After obtaining the payload block based on the method presented in Section 2.2.4, the 2-D payload block is translated into a 1-D bit stream and embed the bit stream into the frequency domain of the host image. Figures 2.33(a) and (b) are the original watermarked image and its counterfeit collage image. During the verification procedure, the modification areas are detected as shown in the rectangles labeled “4” in Figure 2.33(c). Figure 2.33(d) shows the comparison between the original image and the counterfeit collage image. Note that, by adopting Wong’s scheme, the embedded fragile watermark is unable to detect the malicious modification in Figure 2.33(b).

## 2.6 GA-Based Robust Watermarking for Medical Image

A Region Of Interest (ROI) of a medical image is an area including important information and must be stored without any distortion. In order to achieve optimal compression as well as satisfactory visualization of medical images, the ROI is compressed by lossless compression, and the rest by lossy compression. Furthermore, security is an important issue in web-based medical information system. Watermarking skill is often used for protecting medical images. In this section, a robust technique is presented by embedding the watermark of signature information or textual data around the ROI of a medical image based on genetic algorithms. A fragile watermark is adopted to detect any unauthorized modification. The embedding of watermark in the frequency domain is more difficult to be pirated than in spatial domain.

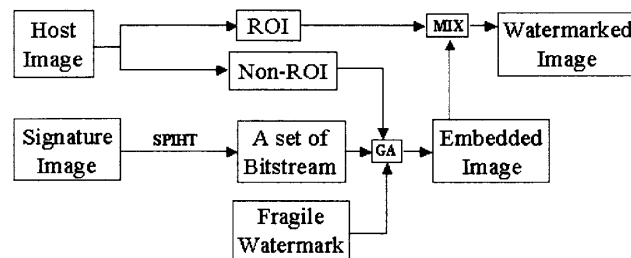
### 2.6.1 The Proposed Algorithm

In order to achieve higher compression rate without distorting the important data in a medical image, the lossless compression is performed to the ROI of a medical image and the lossy compression to the rest. For the purpose of protecting medical images and maintaining their integrity, the information watermark is embedded (signature image or textual data) and fragile watermark into the frequency domain surrounding the ROI part. Meanwhile, the embedded information watermark is pre-processed into a bitstream depending on different types of watermark.

### 2.6.1.1 The Signature Image.

#### A. Encoding Procedure

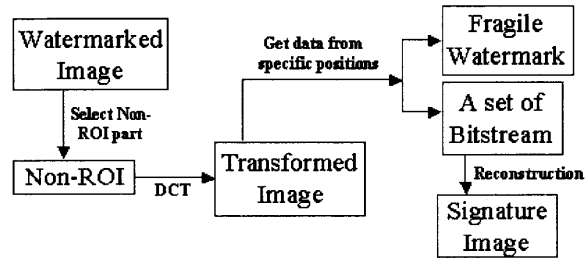
Figure 2.34 shows the encoding procedure when the watermark is a signature image. The host image (medical image) is separated into two parts, ROI and non-ROI. The signature image and fragile watermark are embedded into the non-ROI part by genetic algorithms. Note that the signature image is pre-processed by SPIHT in order for its reconstruction to perceptual satisfaction. Last, the watermarked image is obtained by combining the embedded non-ROI and ROI.



**Figure 2.34** The encoding procedure of embedding a signature image.

#### B. Decoding Procedure

Figure 2.35 shows the decoding procedure. The non-ROI part is selected from the watermarked image, and is transformed by DCT. Last, the fragile watermark and a set of bitstreams are obtained by extracting data from the specific positions of the coefficients in the frequency domain. It is possible to decide the integrity of the medical image by checking the fragile watermark, and obtain the signature image by reconstructing the bitstream using SPIHT.

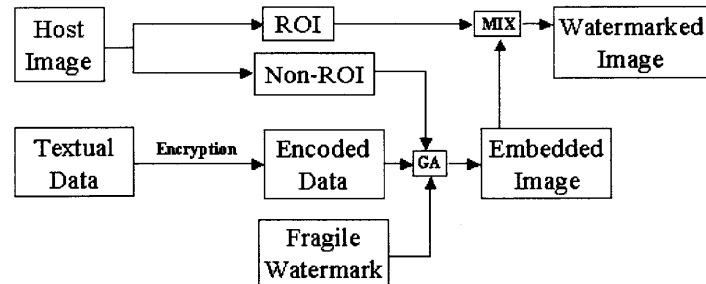


**Figure 2.35** The decoding procedure of embedding a signature image.

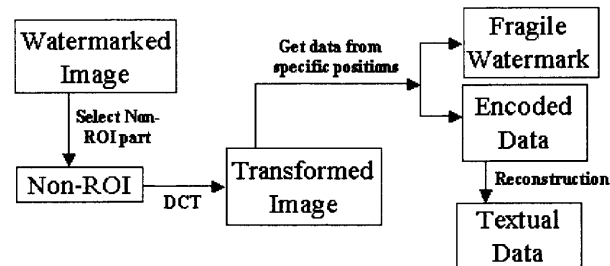
### 2.6.1.2 The Textual Data.

#### A. Encoding and Decoding Procedures

Figure 2.36 shows the encoding procedure when the watermark is textual data. The procedure is similar to the signature image encoding except the encryption of textual data is used. Its decoding procedure is shown in Figure 2.37.



**Figure 2.36** The encoding procedure of embedding textual data.



**Figure 2.37** The decoding procedure of embedding textual data.

## B. The Encryption Scheme in Textual Data

There are many techniques for encrypting textual data. Generally, the main idea is translating the textual data of plain text into secret codes of cipher text.

An easy way to achieve encryption is bit shifting. That is, the character is considered as a byte value and shift its bit position. For example, the byte value of a letter “F” is “01000110” which can change it to be “d” as “01100100” by shifting the left-most 4 bits to the right side.

An encryption method by taking the logarithm of *ASCII* codes was developed in (Acharya et al., 2001). Their encryption algorithm can be mathematically stated as

$$T_e = (\log(T_o * 2) * 100) - 300$$

where  $T_e$  denotes the encrypted text and  $T_o$  denotes the ASCII code of the original text. The decrypted text can be obtained by

$$T_o = \exp\{(T_e + 300)/100 - \log 2\}$$

Note that, the encrypted information ( $T_e$ ) is stored as an integer.

### 2.6.2 The Algorithm for the Proposed Technique

Let  $H$  be the original host image with size  $K \times K$ , which is separated into  $H^{ROI}$  (ROI) and  $H^{NROI}$  (non-ROI) with sizes of  $N \times M$  and  $K \times K - N \times M$ , respectively. Let  $S$  and  $T$  denote a signature image with size of  $W \times W$  and textual data, respectively.  $S^B$  is a bitstream obtained by compressing  $S$  in SPIHT compression or encoding  $T_o$  by the encryption technique.  $W^F$  is the fragile watermark.  $H^{WROI}$  is obtained by adjusting the pixel values of  $H^{NROI}$  in which  $S^B$  and  $W^F$  can be extracted from some specific

positions of coefficients of the frequency domain in  $H^{WROI}$ .  $H^{Final}$  is the final image by combining  $H^{ROI}$  and  $H^{WROI}$ .

**Algorithm:**

1. Separate the host image,  $H$ , into  $H^{ROI}$  and  $H^{NROI}$ .

$H = \{ h(i, j) , 0 \leq i, j < K \}$ , where  $h(i, j) \in \{0, 1, 2, \dots, 2^L - 1\}$  and  $L$  is the number of bits used in the gray level of pixels.

$H^{ROI} = \{ h^{ROI}(i, j) , 0 \leq i < N , 0 \leq j < M \}$ , where  $h^{ROI}(i, j) \in \{0, 1, 2, \dots, 2^L - 1\}$

$H^{NROI} = \{ h^{NROI}(i, j) \}$ , where  $h^{NROI}(i, j) \in \{0, 1, 2, \dots, 2^L - 1\}$

2. For Signature image:

Transform  $S$  to  $S^B$  by *SPIHT*.

$S = \{ s(i, j) , 0 \leq i, j < W \}$ , where  $s(i, j) \in \{0, 1, 2, \dots, 2^L - 1\}$ .

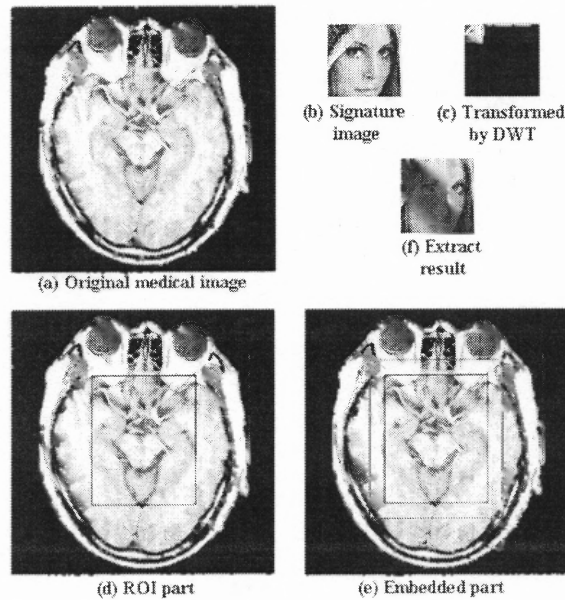
$S^B$  is the bitstream consisting of bits 0 and 1.

For Textual Data:

Encode  $T$  to  $S^B$  by the encryption technique.

3. Adjust  $H^{NROI}$  by genetic algorithms to obtain  $H^{WROI}$  in which  $S^B$  and  $W^F$  can be extracted from the coefficients of its frequency domain.
4. Combine  $H^{ROI}$  and  $H^{WROI}$  to obtain the final image  $H^{Final}$ .

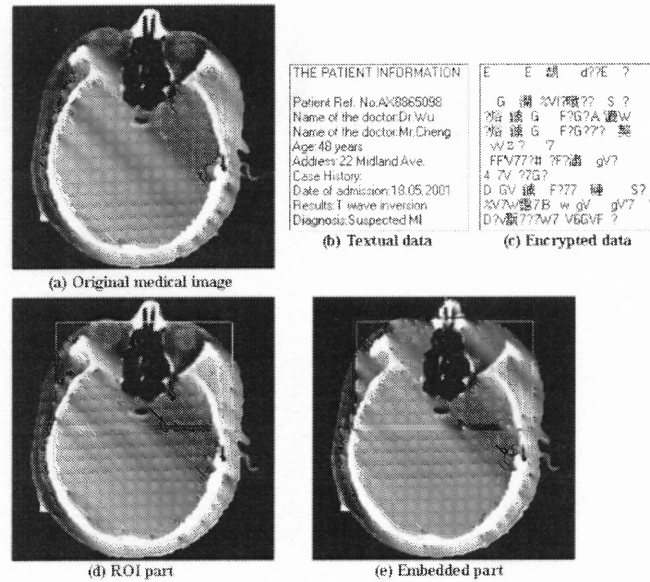
Note that not only the information image, but also the fragile watermark can be embedded into the host image. If there exist more than one regular ROIs, the following information can be recorded for each ROI: the top-left corner coordinates, the width, and the height. The watermark image can be placed in a group of  $8 \times 8$  blocks that are extracted from the outer layer of each ROI. For an irregular polygon or circular shape of ROI, the starting and ending positions of each row in ROI have to be recorded in order to extract the data from the outer layer of its boundary.



**Figure 2.38** The example of embedding signature image.

### 2.6.3 Experimental Results

Figure 2.38 shows the example of embedding a signature image into a MRI brain image. Figures 2.38(a) and (b) show the original medical image with size of  $230 \times 230$  and the signature image with size of  $64 \times 64$ , respectively. Figure 2.38(c) is the transformed image by DWT. The ROI part is marked as a rectangle with size of  $91 \times 112$ , as shown in Figure 2.38(d). Figure 2.38(c) can be encoded by SPIHT into a set of bitstreams which is embedded around the ROI part. In Figure 2.38(e), the area between two rectangles,  $91 \times 112$  and  $117 \times 138$ , is the clipped watermarked area. The signature image is extracted and the reconstructed result is shown in Figure 2.38(f). Table 2.6 shows PSNR of the original and reconstructed signature images, and of the non-watermarked and watermarked parts.



**Figure 2.39** The example of embedding textual data.

Figure 2.39 shows the example of embedding textual data into a CT brain image. Figures 2.39(a) and (b) show the original medical image with size of  $260 \times 260$  and the textual data, respectively. Figure 2.39(c) is the encrypted data. The ROI part is marked as a rectangle with size of  $179 \times 109$ , as shown in Figure 2.39(d). A set of bitstreams are obtained by shifting the right-most 4 bits to the left side. In Figure 2.39(e), the area between two rectangles is the clipped watermarked area. Note that, the original and extracted textual data are exactly the same.

**Table 2.6** PSNR of Signature and of Medical Images

	<b>Original and reconstructed signature images</b>	<b>Non-watermarked and watermarked medical images</b>
<b>PSNR</b>	24.08	38.28

The proposed techniques are focused on the following two goals: first, the embedded watermarks should be as robust as possible and can be used to detect any unauthorized modification; second, the compression rate of an image should be as high as



possible. Furthermore, the embedded watermarks will be disturbed when real numbers are converted into integers. Genetic algorithms are adopted to be the best way to achieve these goals.

## 2.7 The Adjusted-Purpose Watermarking Technique

Selection of a suitable watermarking technique is not easy since there exist many different watermarking techniques for variant types of watermarks. In this Section, a novel adjusted-purpose digital watermarking technique is presented to simplify the selection and to integrate different watermarking techniques. The quantity factor (QF) is used to affect the embedded watermarks to become fragile, semi-fragile or robust watermarks. The varying sized transform window (VSTW) is designed to determine whether the embedded strategy should be in spatial or frequency domains.

### 2.7.1 The Strategies for Adjusting VSTW and QF

The size of VSTW can determine whether the spatial or frequency domain is adopted. For example, if the size is  $1 \times 1$ , it is equivalent to the spatial domain approach. If the size is  $2 \times 2$ ,  $4 \times 4$ , or  $8 \times 8$ , the original image is separated using the size of sub-blocks accordingly, and perform DCT on each sub-block. It belongs to the frequency domain approach. Note that, the size of VSTW could be arbitrary, not necessarily a square of power of 2.

In principle, fragile watermarks are embedded into LSB, but robust watermarks are into higher significant bits. In order to integrate both watermarks, the QFs are developed based on the spread spectrum. It is obviously that shifting the binary bits of  $x$  to left (denoted as  $shl$ ) or right (denoted as  $shr$ ) by  $y$  bits is equivalent to multiplying or

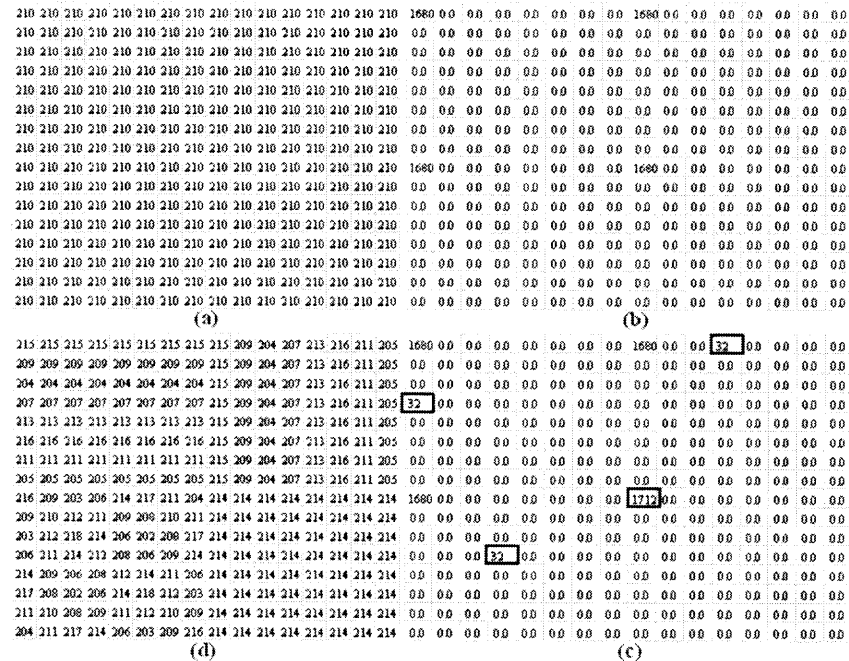
dividing  $x$  by  $2^y$ . The position for embedding the watermark can be determined based on the QF as shown in Table 2.7. For example, if the watermark is embedded into the 4<sup>th</sup> bit (i.e., QF=8) of a pixel, the pixel value are divided by 8, and then replace the resulting LSB by the watermark. Finally, the watermarked value can be obtained by multiplying the replaced result by 8.

**Table 2.7** Relation Between QF and Embedded Positions

<b>Int.</b>	<b>Binary</b>	<b>Embedded Watermark</b>	<b>Embedded Position</b>	<b>QF</b>	<b>After Embedding</b>	<b>New Int.</b>
41	0010100 <b>I</b>	0	1 (LSB)	1	0010100 <b>0</b>	40
43	0010 <b>I</b> 011	0	4	8	0010 <b>0</b> 011	35
66	010000 <b>I</b> 0	0	2	2	010000 <b>0</b> 0	64
110	0110 <b>I</b> 110	0	4	8	0110 <b>0</b> 110	102
210	110 <b>I</b> 0010	0	5	16	110 <b>0</b> 0010	194

Cox et al. [18] proposed the spread spectrum watermarking and indicated that the watermarks should not be embedded into insignificant regions of the image or its spectrum since many signal and geometric processes can affect these components. For robustness, they suggested that the watermarks should be embedded into the regions with large magnitudes in coefficients of a transformed image. For example, there are a few large magnitudes in the coefficients as shown in Figure 2.40. Figure 2.40(a) shows the original image of size  $16 \times 16$  in which all pixels are 210. Figure 2.40(b) is the transformed image by DCT using the  $8 \times 8$  VSTW. There are four significant regions with the large magnitude. Figure 2.40(c) shows the results after embedding the watermarks into the sixth bit of four different locations as marked in a bold box. Note that, the watermarks are not embedded into the four significant regions. After performing the IDCT on Figure 2.40(c), the watermarked image is obtained as shown in Figure 2.40(d). It is obviously that the original image will have a huge degradation because the

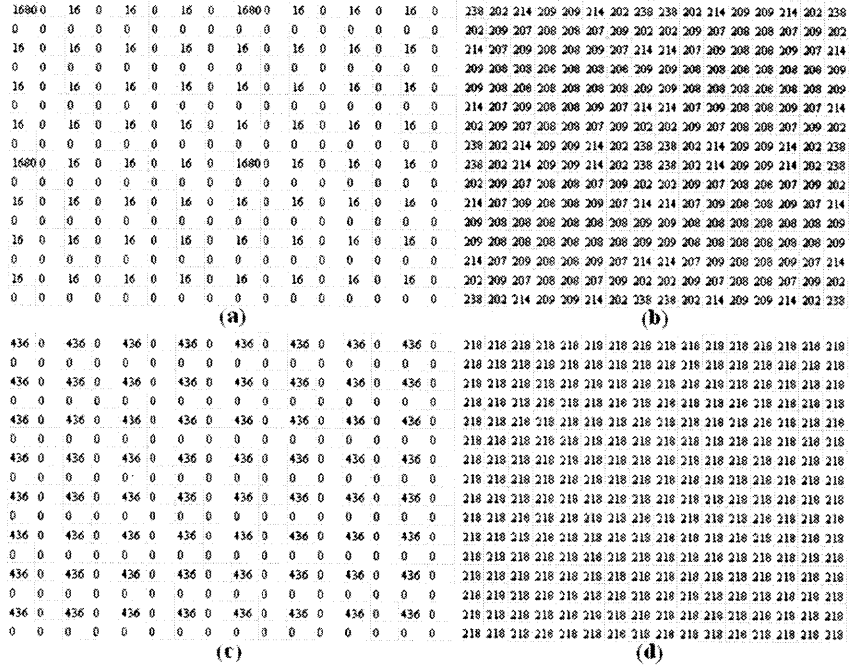
watermarks are embedded into non-significant regions. Therefore, the size of embedded watermarks is quite limited.



**Figure 2.40** The example of embedding watermarks into different spectral regions.

Figure 2.41 shows an example of the AP watermarking for enlarging the capacity of the watermarks. In order to achieve this, a small sized VSTW is selected. The watermark used is the 64 bits of all 1's. Considering robustness, the watermark is embedded into the higher significant bit (e.g. 5<sup>th</sup> bit). Given the size of VSTW being  $8 \times 8$ , Figure 2.41(a) is obtained by embedding the watermark into Figure 2.40(b). After performing the IDCT on Figure 2.41(a), the watermarked image is obtained as shown in Figure 2.41(b) which is largely distorted. It is obviously that the performance is not good since too much data is embedded. However, if the  $2 \times 2$  VSTW is selected, Figure 2.41(c) can be obtained, that is the result after embedding the watermark into the frequency domain of the original image. Similarly, after performing the IDCT on Figure 2.41(c), the

watermarked image can be obtained as shown in Figure 2.41(d). Therefore, by the VSTW, it is possible to determine the size of each sub-image to enlarge the capacity of the watermarks.



**Figure 2.41** The example of enlarging the capacity of watermark.

By adjusting the VSTW and QF, the AP watermarking can be equivalent to the existing watermarking techniques. For example, by setting VSTW to be  $N \times N$  and each element in QF to be a large number, the AP watermarking is the same as the spread spectrum watermarking developed by Cox et al. [18]. To simulate the block-based fragile watermarking by Wong [104], the VSTW is set to be  $1 \times 1$  and each element in QF to be “1.” The general rules for determining VSTW and QF are shown in Table 2.8.

**Table 2.8** General Rules for Determining VSTW and QF

Purpose	VSTW	QF
Robust and Spatial	$1 \times 1$	Larger than “1”
Robust and Frequency	Larger than $1 \times 1$	Larger than “1”
Fragile and Spatial	$1 \times 1$	Less than or Equal to “1”
Fragile and Frequency	Larger than $1 \times 1$	Less than or Equal to “1”

### 2.7.2 The Proposed Adjusted-Purpose Watermarking Technique

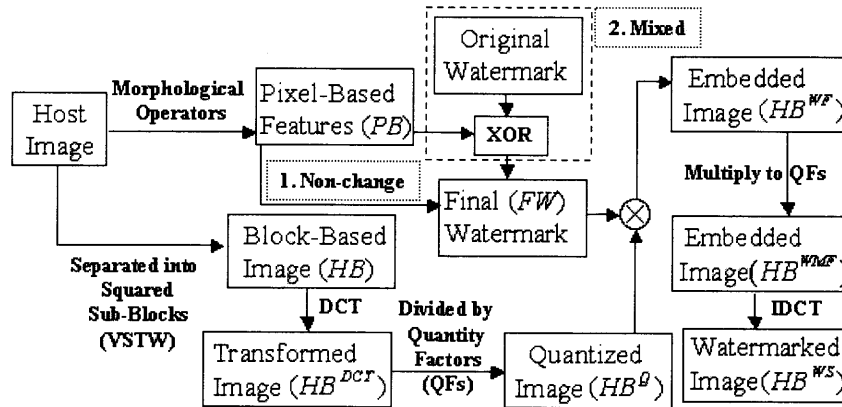
Let  $H$  be a gray-level host image of size  $N \times N$ , and  $W$  be a binary watermark image of size  $M \times M$ . Let  $HB$  be the block-based image obtained by dividing  $H$  into non-overlapping blocks of size  $n \times n$ , and  $HB^{DCT}$  be the transformed image by DCT. Let  $Q$  be the quantity factor of the same size as  $HB$ , and  $HB^Q$  be the image obtained by dividing  $HB^{DCT}$  by  $Q$ . Let  $PB$  be the pixel-based features extracted from  $H$ , and  $FW$  be the final watermark by applying an XOR operator of  $PB$  and  $W$ . Let  $\otimes$  denote the operator that substitutes bits of a watermark for bits of a host image using the Least Significant Bit (LSB) modification. Note that, the LSB modification tends to exploit the bit-plane, such that the modification does not cause significantly different visual perception. Let  $HB^{WF}$  be the watermarked image, where  $HB^Q$  and  $FW$  are combined using LSB modification. Let  $HB^{WMF}$  be the watermarked image obtained by multiplying  $HB^{WF}$  by  $Q$ . Let  $HB^{WS}$  be the watermarked image obtained by converting  $HB^{WMF}$  back to the spatial domain by Inverse Discrete Cosine Transform (IDCT).

The algorithm for the encoding procedure of the AP watermarking technique is presented below, and its flowchart is shown in Figure 2.42.

#### The Encoding Procedure of the AP Watermarking Algorithm:

1. Obtain  $PB$  from  $H$  based on morphological operators.
2. Case 1: Without an additional watermark: Set  $FW = PB$ .  
Case 2: With a watermark: Obtain  $FW$  by applying the XOR operator of  $PB$  and  $W$ .
3. Determine  $n$  and obtain  $HB$  by splitting  $H$  into non-overlapping sub-watermarks of size  $n \times n$ .

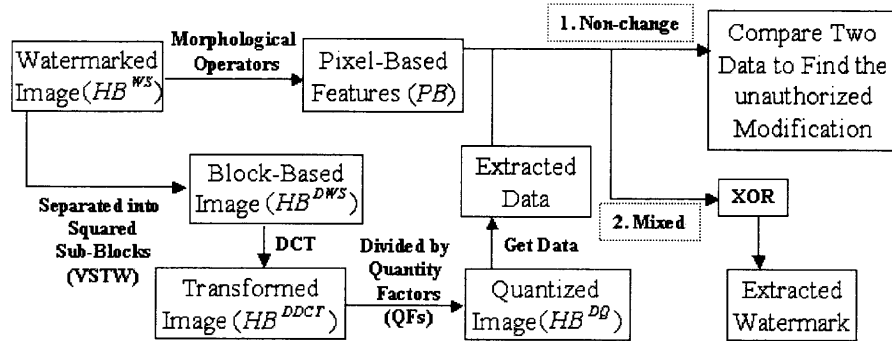
4. Transform  $HB$  by DCT to obtain  $HB^{DCT}$ .
5. Obtain  $HB^Q$  by dividing  $HB^{DCT}$  by  $Q$ .
6. Insert  $FW$  into coefficients of  $HB^Q$  by LSB modification to obtain  $HB^{WF}$ .
7. Obtain  $HB^{WMF}$  by multiplying  $HB^{WF}$  by  $Q$ .
8. Obtain  $HB^{WS}$  by converting  $HB^{WMF}$  back to the spatial domain by IDCT.



**Figure 2.42** The encoding procedure of the AP watermarking.

Let  $HB^{DWS}$  be the block-based image by dividing  $HB^{WS}$  into non-overlapping blocks of size  $n \times n$ . Let  $HB^{DDCT}$  be the transformed image of  $HB^{DWS}$  by DCT, and  $HB^{DQ}$  be the image obtained by dividing  $HB^{DDCT}$  by  $Q$ . Let  $HB^{LSB}$  be the binary image obtained by extracting the LSB value of each pixel in  $HB^{DQ}$ . Let  $FW^D$  be the extracted watermark by applying an XOR operator of  $PB$  and  $HB^{DQ}$ .

The algorithm for the decoding procedure of the AP watermarking technique is presented below, and its flowchart is shown in Figure 2.43.



**Figure 2.43** The decoding procedure of the AP watermarking.

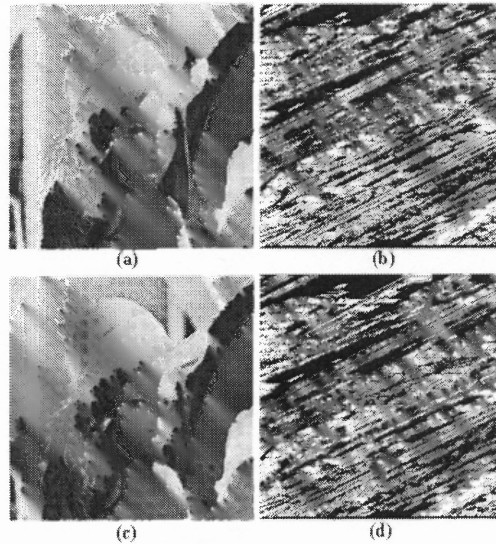
### The Decoding Procedure of the AP Watermarking Algorithm:

1. Obtain  $PB$  from  $HB^{WS}$  based on morphological operators.
2. Determine  $n$  and obtain  $HB^{DWS}$  by splitting  $HB^{WS}$  into non-overlapping sub-watermarks of size  $n \times n$ .
3. Transform  $HB^{DWS}$  by DCT to obtain  $HB^{DDCT}$ .
4. Obtain  $HB^{DQ}$  by dividing  $HB^{DDCT}$  by  $Q$ .
5. Obtain  $HB^{LSB}$  by extracting the LSB value of each pixel in  $HB^{DQ}$ .
6. Case 1: Without an additional watermark: Set  $FW^D = HB^{LSB}$ .  
Case 2: With a watermark  $W$ : Obtain  $FW^D$  by applying the XOR operator of  $PB$  and  $HB^{DQ}$ .

### 2.7.3 Experimental Results

Figures 2.44(a) and (b) show an original image and its pixel-based features, respectively. Figures 2.44(c) and (d) show the image after JPEG with 20% quality level and its pixel-based features, respectively. By comparing Figures 2.44(b) and (d), the pixel-based features almost remain the same even after low-quality JPEG compression. Table 2.9 provides the quantitative measures. Note that, the size of the structuring element is  $3 \times 3$

and  $(T_1, T_1)$  indicates the threshold values adopted in step 4 of the algorithm. Table 2.10 provides the analysis based on different sizes of the structuring element.



**Figure 2.44** The example of the pixel-based features of an image.

**Table 2.9** Quantitative Measures for Pixel-Based Features

JPEG Quality	PSNR	$T_1=0.1$	$T_1=0.15$	$T_1=0.2$	$T_1=0.25$	$T_1=0.3$	$T_1=0.35$
		$T_2=0.9$	$T_2=0.85$	$T_2=0.8$	$T_2=0.75$	$T_2=0.7$	$T_2=0.65$
90 %	48.98	97.67%	97.31%	94.14%	97.05%	96.78%	95.73%
80 %	44.92	96.44%	95.98%	95.78%	95.67%	95.04%	93.21%
70 %	41.99	95.88%	95.09%	94.84%	94.65%	93.77%	91.84%
60 %	39.18	95.45%	94.91%	94.69%	94.20%	93.50%	91.31%
50 %	37.68	94.64%	94.04%	93.67%	93.37%	92.33%	90.20%
40 %	35.81	94.15%	93.48%	93.08%	92.49%	92.12%	90.06%
30 %	32.33	92.22%	91.24%	90.73%	90.42%	89.62%	86.44%
20 %	30.57	91.45%	90.38%	89.93%	89.38%	88.47%	86.78%
10 %	24.51	74.11%	74.08%	73.96%	74.15%	75.09%	78.73%



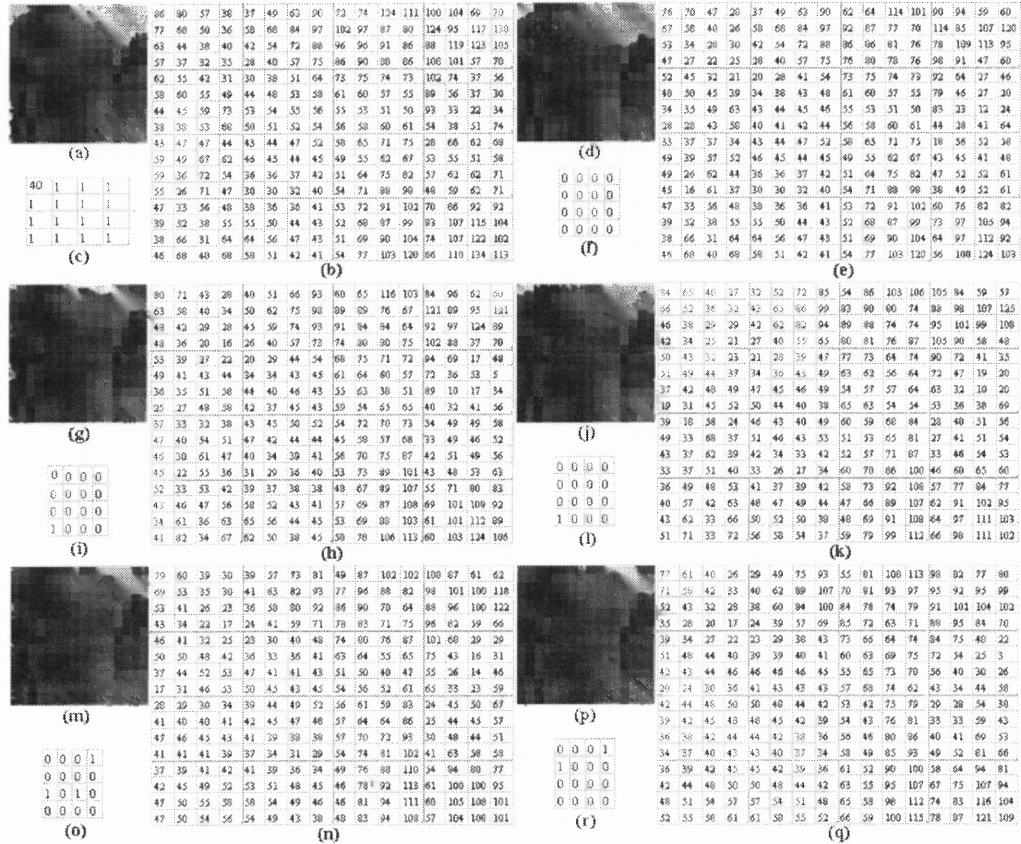
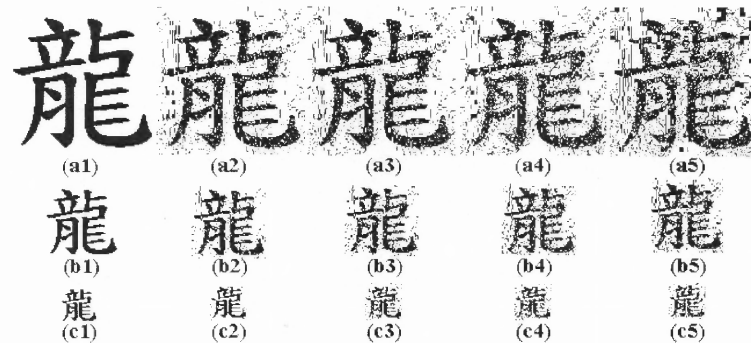


Figure 2.45 The example of the AP watermarking technique.

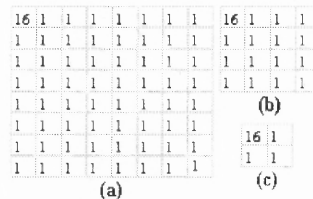
Table 2.10 Pixel-Based Features Based on Sizes of Structuring Elements

JPEG Quality	SE = 5×5		SE = 7×7	
	$T_1=0.1$	$T_1=0.25$	$T_1=0.1$	$T_1=0.25$
	$T_2=0.9$	$T_2=0.75$	$T_2=0.9$	$T_2=0.75$
90 %	98.75%	98.27%	99.10%	98.68%
80 %	98.15%	97.27%	98.56%	97.98%
70 %	97.38%	96.05%	97.71%	96.67%
60 %	97.16%	95.87%	96.65%	96.35%
50 %	96.30%	94.60%	96.52%	95.55%
40 %	95.92%	94.43%	96.38%	95.06%
30 %	94.32%	91.73%	95.04%	93.18%
20 %	93.90%	91.76%	94.37%	92.83%
10 %	76.89%	76.78%	79.30%	79.18%

Figures 2.45(a) and (b) show an image and its data of size  $16 \times 16$ . The size of VSTW adopted is  $4 \times 4$  in this example. Figure 2.45(c) shows the QF corresponding to each sub-image. Figure 2.45(f) is the original watermark. Note that the watermark is embedded into the DC component of coefficients in the transformed sub-images. That is, each sub-image is inserted one bit of watermark. Figures 2.45(d) and (e) show the watermarked image and its data. The JPEG compression is adopted as the attacking method. Pairs of Figures 2.45 ((g),(h)), ((j),(k)), ((m),(n)), and ((p),(q)) are the results of JPEG compression at four quality levels 80%, 60%, 40%, and 20%, respectively. Figures 2.45(i), (l), (o), and (r) are the corresponding extracted watermarks.



**Figure 2.46** The example of extracting embedded watermarks after JPEG compression.



**Figure 2.47** The QFs for their corresponding VSTW's.

Figure 2.46 shows an example of the AP watermarking algorithm for extracting the embedded watermarks after different JPEG compression qualities. Figures 2.46(a1), (b1) and (c1) are the original watermarks of size  $128 \times 128$ ,  $64 \times 64$  and  $32 \times 32$ , respectively. Sizes of VSTW's used for embedding those watermarks are  $2 \times 2$ ,  $4 \times 4$ , and

$8 \times 8$ , and their corresponding QFs are shown in Figures 2.47(a), (b) and (c). After embedding watermarks into the host image, four JPEG compression qualities are used such as 80%, 60%, 40%, and 20% as the attackers. Figures 2.46(a2), (b2), and (c2) are the extracted watermarks after 80% JPEG compression. Similarly, Figures 2.46((a3),(b3),(c3)), ((a4),(b4),(c4)), and ((a5),(b5),(c5)) are obtained after 60%, 40%, and 20% JPEG compressions, respectively. Figures 2.45 and 2.46 demonstrate that the AP watermarking can resist the JPEG compression attack.

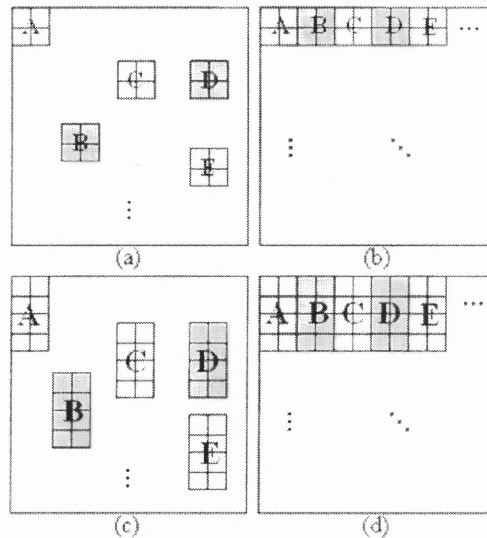
## 2.8 The Robust High-Capacity Watermarking Technique

Zhao et al. [110] presented a robust wavelet-domain watermarking algorithm based on the chaotic map. They divide an image into a set of  $8 \times 8$  blocks with labels. After the order is mixed by a chaotic map, the first 256 blocks are selected for embedding watermarks. Miller et al. [57] proposed a robust high-capacity watermarking algorithm by informed coding and embedding. However, they can embed only 1,380 bits of information in an image of size  $240 \times 368$ , i.e., a capacity of 0.015625 bits/pixel.

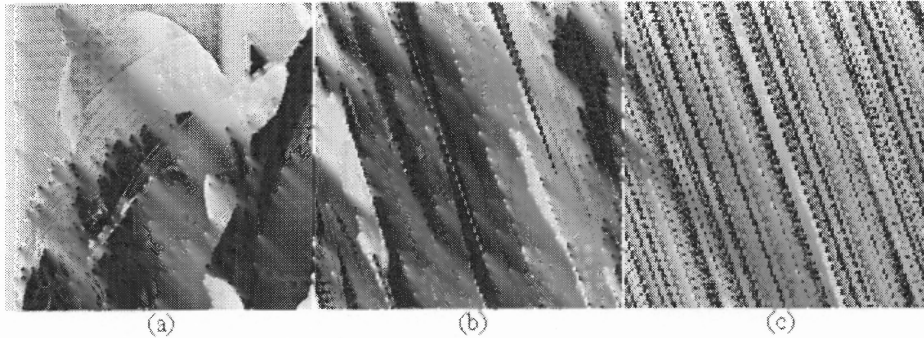
In order to enlarge the watermark capacity, the idea of breaking the local spatial similarity of an image is considered, so it can generate more significant coefficients of the transformed image. the chaotic map is utilized to relocate pixels. Unfortunately, the traditional pixel-based chaotic map [100, 110] cannot be adopted in the robust high-capacity watermarking algorithm because the reference register cannot be obtained. The reference register is created to indicate the locations for embedding watermarks and will be introduced in Section 2.8.2. Therefore, a new block-based chaotic map is developed to achieve the goals.

### 2.8.1 The Block-Based Chaotic Map

The traditional chaotic map can relocate the pixel locations. It cannot be used in the algorithm because the reference register cannot be obtained. In addition, to be more efficient, the block-based chaotic map is developed to relocate the pixel locations based on “block” unit (i.e., a set of connected pixels) instead of “pixel” unit. Figures 2.48(a) and (b) show a diagram and its relocated result based on the block size of  $2 \times 2$ . Figures 2.48 (c) and (d) show a diagram and its relocated result based on the block size of  $2 \times 4$ . In general, the bigger the block size is, the larger the local similarity is. An example by setting the block size of  $2 \times 2$  and  $l = 2$  is shown in Figure 2.49, where (a) is the Lena image, (b) its relocated image, and (c) the resulting image in the next iteration.



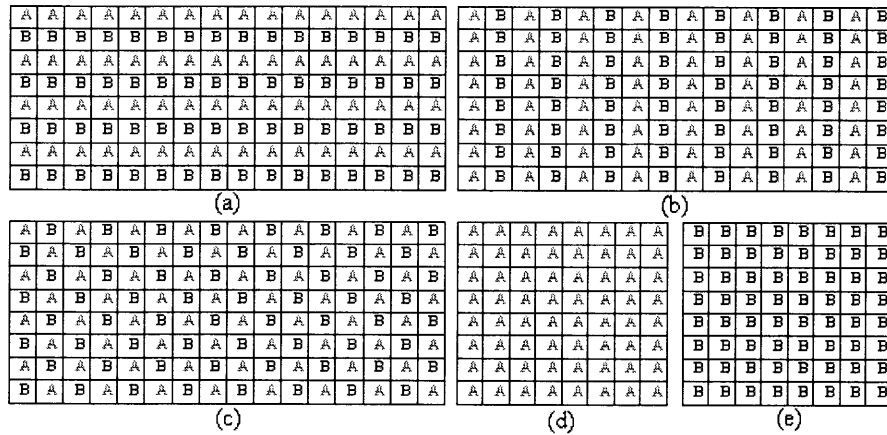
**Figure 2.48** An example of the block-based relocation.



**Figure 2.49** An example of performing the block-based chaotic map.

### 2.8.2 The Reference Register

The intersection-based pixels collection (IBPC) intends to label an image using two symbols alternatively and then collect the two sub-images with the same symbol as illustrated in Figure 2.50. Figures 2.50(a), (b), and (c) show three different approaches to collect the pixels with same label horizontally, vertically, and diagonally, respectively. The two sub-images formed are shown in Figures 2.50(d) and (e). Note that the pair of images obtained has local spatial similarity, even after transformation or attacks. From experiments, the diagonal IBPC shows better similarity in the RHC watermarking algorithm. Therefore, a pair of  $8 \times 8$  coefficients are generated from the  $16 \times 8$  image by using the IBPC approach followed by the DCT. Afterwards, one is used as the reference register for indicating significant DCT coefficients, and the other is used as the container for embedding watermarks.



**Figure 2.50** An example shows how the IBPC works.

An example of demonstrating the similarity of two sub-images obtained by the IBPC is shown in Figure 2.51. Figure 2.51(a) shows the Lena image where a small block of pixels are cropped in Figure 2.51(b). Figures 2.51(c) and (d) show the pair of sub-images obtained from Figure 2.51(b) by the IBPC. Figures 2.51(e) and (f) are their respective DCT images. Figures 2.51(g) and (h) show the results after dividing Figures 2.51(e) and (f) respectively by the quantization table with the *quality factor* ( $QF$ ) of 50. Note that the  $QF$  will be used in Equation (2.2), and one example of the quantization table in JPEG is given in Figure 2.52. It is obviously that Figures 2.51(g) and (h) are similar. Therefore, either one can be utilized as the reference register to indicate the significant coefficients for embedding watermarks. The significant coefficients have the values larger than a pre-defined threshold  $RR_{Th}$ .

Let  $QTable(i, j)$  denote the a quantization table, where  $0 \leq i, j \leq 7$ . The new quantization table  $NewTable(i, j)$  is obtained by

$$NewTable(i, j) = \begin{cases} QTable(i, j) \times \frac{50}{QF} & \text{if } QF < 50 \\ QTable(i, j) \times (2 - 0.02 \times QF) & \text{otherwise} \end{cases} \quad (2.2)$$

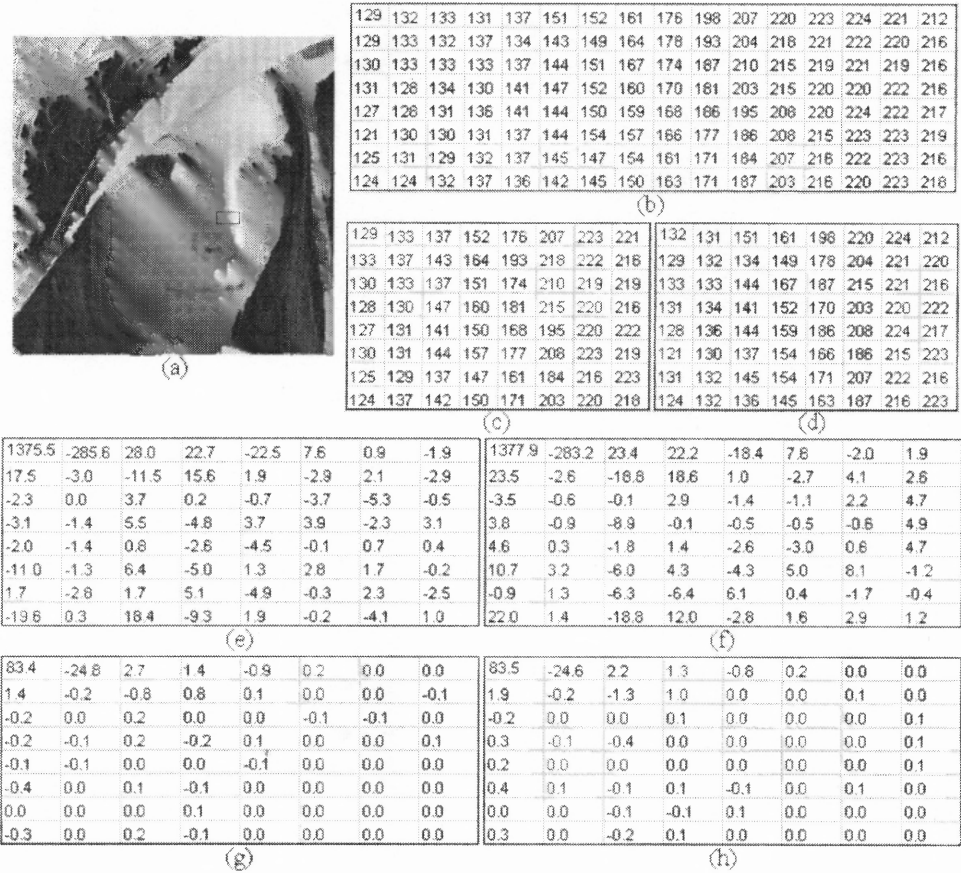


Figure 2.51 The similarity illustration of two sub-images obtained by IBPC.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Figure 2.52 The quantization table of JPEG.

### 2.8.3 The Algorithm of the Robust High-Capacity Watermarking

The *robust high-capacity* (RHC) watermarking algorithm contains two main components: the *block-based chaotic map* (BBCM) and the *reference register* (RR). As aforementioned, to enhance robustness of image watermarking, the messages are embedded into the significant coefficients of the transformed image. The local spatial similarity in most images hinders the amount of significant coefficients and lowers the watermarking capacity. Therefore, in order to enlarge the capacity, the BBCM is created to relocate pixels and the RR is established using the *intersection-based pixels collection* (IBPC) to indicate the locations for embedding watermarks. The BBCM and RR will be described in Sections III and IV. In this section, the watermark embedding will be first presented followed by the watermark extracting procedures.

Following symbols are used to clearly explain the watermark embedding procedures:

$H$ : The gray-level host image of size  $n \times m$ .

$H_{(i,j)}^{16 \times 8}$ : The  $(i, j)$ -th sub-image of size  $16 \times 8$  of  $H$ , where  $1 \leq i \leq \lfloor n/16 \rfloor$  and  $1 \leq j \leq \lfloor m/8 \rfloor$ .

$HA_{(i,j)}^{8 \times 8}$  and  $HB_{(i,j)}^{8 \times 8}$ : A pair of images obtained from  $H_{(i,j)}^{16 \times 8}$  by the IBPC.

$DA_{(i,j)}^{8 \times 8}$  and  $DB_{(i,j)}^{8 \times 8}$ : The transformed images of applying the DCT on  $HA_{(i,j)}^{8 \times 8}$  and  $HB_{(i,j)}^{8 \times 8}$ , respectively.

$QA_{(i,j)}^{8 \times 8}$  and  $QB_{(i,j)}^{8 \times 8}$ : The resulting images of dividing  $DA_{(i,j)}^{8 \times 8}$  and  $DB_{(i,j)}^{8 \times 8}$  by the quantization table, respectively.

$E_{(i,j)}^{8 \times 8}$ : The watermarked image of  $DA_{(i,j)}^{8 \times 8}$  using  $QA_{(i,j)}^{8 \times 8}$  and  $QB_{(i,j)}^{8 \times 8}$ .



$M_{(i,j)}^{8 \times 8}$ : The image after multiplying  $E_{(i,j)}^{8 \times 8}$  by the quantization table.

$I_{(i,j)}^{8 \times 8}$ : The image obtained by the *Inverse Discrete Cosine Transformation* (IDCT) of

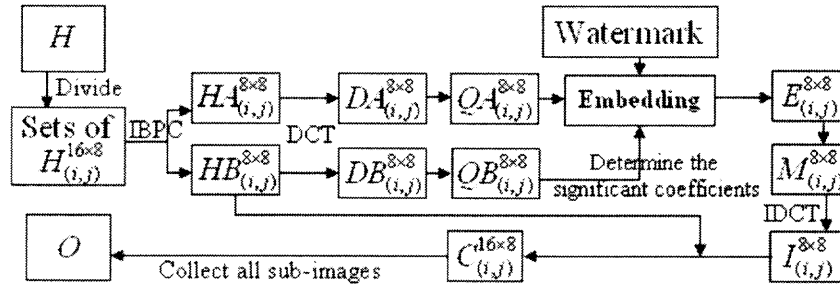
$$M_{(i,j)}^{8 \times 8}.$$

$C_{(i,j)}^{16 \times 8}$ : The watermarked sub-image obtained by combining  $I_{(i,j)}^{8 \times 8}$  and  $HB_{(i,j)}^{8 \times 8}$  using the

IBPC.

$O$ : The output watermarked image obtained by collecting all the sub-images of  $C_{(i,j)}^{16 \times 8}$ .

The overall embedding procedures are presented as follows and the flowchart in Figure 2.53.



**Figure 2.53** The embedding procedure of the RHC watermarking.

### The Embedding Procedures of the RHC Watermarking:

1. Divide an input image  $H$  into a set of sub-images,  $H_{(i,j)}^{16 \times 8}$ , of size  $16 \times 8$ .
2. Build  $HA_{(i,j)}^{8 \times 8}$  and  $HB_{(i,j)}^{8 \times 8}$  from each sub-image  $H_{(i,j)}^{16 \times 8}$  by the IBPC.
3. Obtain  $DA_{(i,j)}^{8 \times 8}$  and  $DB_{(i,j)}^{8 \times 8}$  from  $HA_{(i,j)}^{8 \times 8}$  and  $HB_{(i,j)}^{8 \times 8}$  by the DCT, respectively.
4. Obtain  $QA_{(i,j)}^{8 \times 8}$  and  $QB_{(i,j)}^{8 \times 8}$  from dividing  $DA_{(i,j)}^{8 \times 8}$  and  $DB_{(i,j)}^{8 \times 8}$  by the JPEG quantization table, respectively.

5. Determine the proper positions of significant coefficients in  $QB_{(i,j)}^{8 \times 8}$  and embed watermarks into the corresponding positions in  $QA_{(i,j)}^{8 \times 8}$  to obtain  $E_{(i,j)}^{8 \times 8}$ . The detailed embedding strategy will be described later.
6. Obtain  $M_{(i,j)}^{8 \times 8}$  from multiplying  $E_{(i,j)}^{8 \times 8}$  by the JPEG quantization table.
7. Obtain  $I_{(i,j)}^{8 \times 8}$  by applying IDCT on  $M_{(i,j)}^{8 \times 8}$ .
8. Reconstruct  $C_{(i,j)}^{16 \times 8}$  by combining  $I_{(i,j)}^{8 \times 8}$  and  $HB_{(i,j)}^{8 \times 8}$ .
9. Obtain the output watermarked image  $O$  by collecting all the  $C_{(i,j)}^{16 \times 8}$ 's.

After receiving the watermarked image  $O$ , the hidden message can be extracted by the extracting procedures below. Again, the following symbols are introduced for clearly explaining the watermark extracting procedures:

$O_{(i,j)}^{16 \times 8}$ : The  $(i, j)$ -th sub-image of size  $16 \times 8$  of  $O$ .

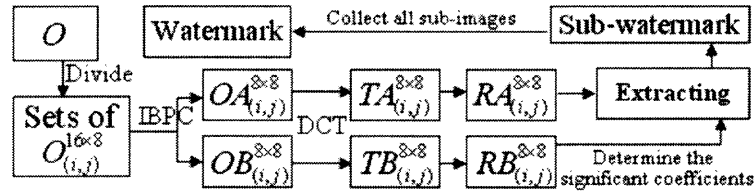
$OA_{(i,j)}^{8 \times 8}$  and  $OB_{(i,j)}^{8 \times 8}$ : A pair of extracted images from  $O_{(i,j)}^{16 \times 8}$  using the IBPC.

$TA_{(i,j)}^{8 \times 8}$  and  $TB_{(i,j)}^{8 \times 8}$ : The transformed image of applying the DCT on  $OA_{(i,j)}^{8 \times 8}$  and  $OB_{(i,j)}^{8 \times 8}$ ,

respectively.

$RA_{(i,j)}^{8 \times 8}$  and  $RB_{(i,j)}^{8 \times 8}$ : The resulting images of dividing  $TA_{(i,j)}^{8 \times 8}$  and  $TB_{(i,j)}^{8 \times 8}$  by the quantization table, respectively.

The watermark extracting procedures are presented as follows and the flowchart in Figure 2.54.



**Figure 2.54** The extracting procedures in the RHC watermarking algorithm.

### The Extracting Procedures of the RHC Watermarking Algorithm:

1. Divide the watermarked image  $O$  into a set of sub-images,  $O_{(i,j)}^{16 \times 8}$ , of size  $16 \times 8$ .
2. Build  $OA_{(i,j)}^{8 \times 8}$  and  $OB_{(i,j)}^{8 \times 8}$  from each sub-image  $O_{(i,j)}^{16 \times 8}$  by the IBPC.
3. Obtain  $TA_{(i,j)}^{8 \times 8}$  and  $TB_{(i,j)}^{8 \times 8}$  from  $OA_{(i,j)}^{8 \times 8}$  and  $OB_{(i,j)}^{8 \times 8}$  by the DCT, respectively.
4. Obtain  $RA_{(i,j)}^{8 \times 8}$  and  $RB_{(i,j)}^{8 \times 8}$  from dividing  $TA_{(i,j)}^{8 \times 8}$  and  $TB_{(i,j)}^{8 \times 8}$  by the JPEG quantization table, respectively.
5. Determine the proper positions of significant coefficients in  $RB_{(i,j)}^{8 \times 8}$  and extract sub-watermark from the corresponding positions in  $RA_{(i,j)}^{8 \times 8}$ . The detailed extracting strategy will be described later.
6. Collect all the sub-watermarks to obtain the watermark.

Note that, for the embedding strategy, each pair of  $QA_{(i,j)}^{8 \times 8}$  (container) and  $QB_{(i,j)}^{8 \times 8}$  (reference register) are obtained first. After the significant coefficients are determined by the reference register, the watermarks are embedded into the corresponding positions of the cover-coefficients by adding the values as in Equation (2.3). Let  $V_{(k,l)}^C$  and  $V_{(k,l)}^R$  denote the values of  $QA_{(i,j)}^{8 \times 8}$  and  $QB_{(i,j)}^{8 \times 8}$ , respectively. Let  $S_{(k,l)}^C$  be the result after embedding the message.

$$S_{(k,l)}^C = \begin{cases} V_{(k,l)}^C + \alpha V_{(k,l)}^R & \text{if } V_{(k,l)}^R \geq RR_{Th} \\ V_{(k,l)}^C & \text{otherwise} \end{cases}, \quad (2.3)$$

where  $0 \leq k, l \leq 7$  and  $\alpha > 0$ . Note that, the bigger the  $\alpha$  is, the higher the robustness is. The  $8 \times 8$   $S_{(k,l)}^C$ 's are collected to form the corresponding  $E_{(i,j)}^{8 \times 8}$ . The embedding strategy is presented below.

### Watermark Embedding Strategy:

1. Determine the embedding location by checking significant coefficients of  $QB_{(i,j)}^{8 \times 8}$ .
2. If the embedding message is "1" and  $V_{(k,l)}^R \geq RR_{Th}$ ,  $S_{(k,l)}^C = V_{(k,l)}^C + \alpha V_{(k,l)}^R$  ;  
otherwise,  $S_{(k,l)}^C = V_{(k,l)}^C$ .

For the extracting strategy, each pair of  $RA_{(i,j)}^{8 \times 8}$  (container) and  $RB_{(i,j)}^{8 \times 8}$  (reference register) is obtained first. After determining the embedded positions of the reference register, the watermark can be extracted by Equation (2.4) and the watermarked coefficients can be used to construct the original ones by Equation (2.5). Let  $W_{(k,l)}^C$  and  $W_{(k,l)}^R$  denote the values of  $RA_{(i,j)}^{8 \times 8}$  and  $RB_{(i,j)}^{8 \times 8}$ , respectively. Let  $F_{(k,l)}^C$  be the result after the additional amount is removed by Equation (2.5), and  $w$  be the embedded message. The following equations can be derived.

$$w = \begin{cases} 1 & \text{if } (W_{(k,l)}^C - W_{(k,l)}^R) \geq \frac{\alpha}{2} V_{(k,l)}^R \\ 0 & \text{elsewhere} \end{cases} \quad (2.4)$$

$$F_{(k,l)}^C = \begin{cases} W_{(k,l)}^C - \alpha W_{(k,l)}^R & \text{if } w = 1 \\ W_{(k,l)}^C & \text{otherwise} \end{cases} \quad (2.5)$$

The embedding strategy of the RHC watermarking is presented below.

### Watermark Extracting Strategy:

1. Determine the extracting location by checking the significant coefficients of  $RB_{(i,j)}^{8 \times 8}$ .
2. Obtain  $W_{(k,l)}^C$  and  $W_{(k,l)}^R$ .
3. Obtain the embedded message using Equation (2.4).
4. If the embedded message is 1,  $F_{(k,l)}^C = W_{(k,l)}^C - \alpha W_{(k,l)}^R$ ;  
otherwise,  $F_{(k,l)}^C = W_{(k,l)}^C$ .

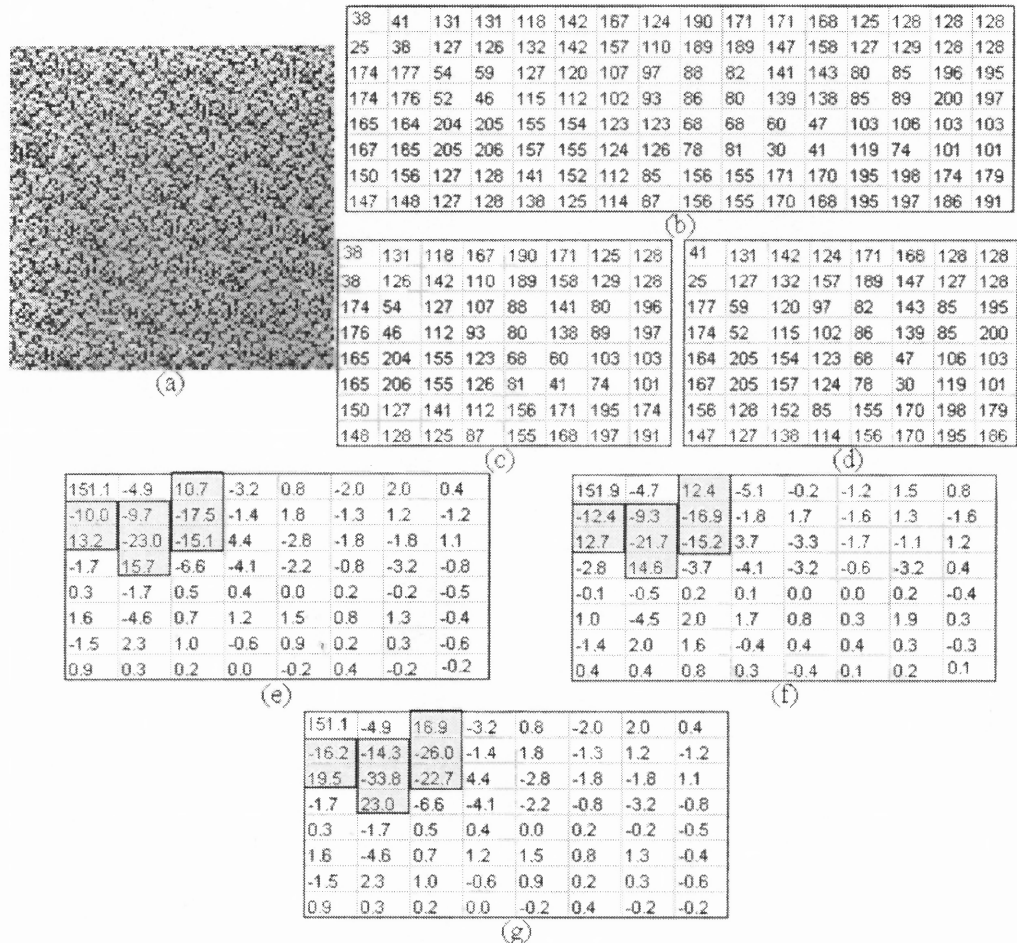
### 2.8.4 Experimental Results

In this section, an example is shown to illustrate that the watermark capacity can be enlarged by the block-based chaotic map (BBCM) and the intersection-based pixels collection (IBPC) approaches. The experimental results are obtained based on 200 images and comparisons with the “iciens” algorithm by Miller et al. [57].

#### A. Capacity Enlargement

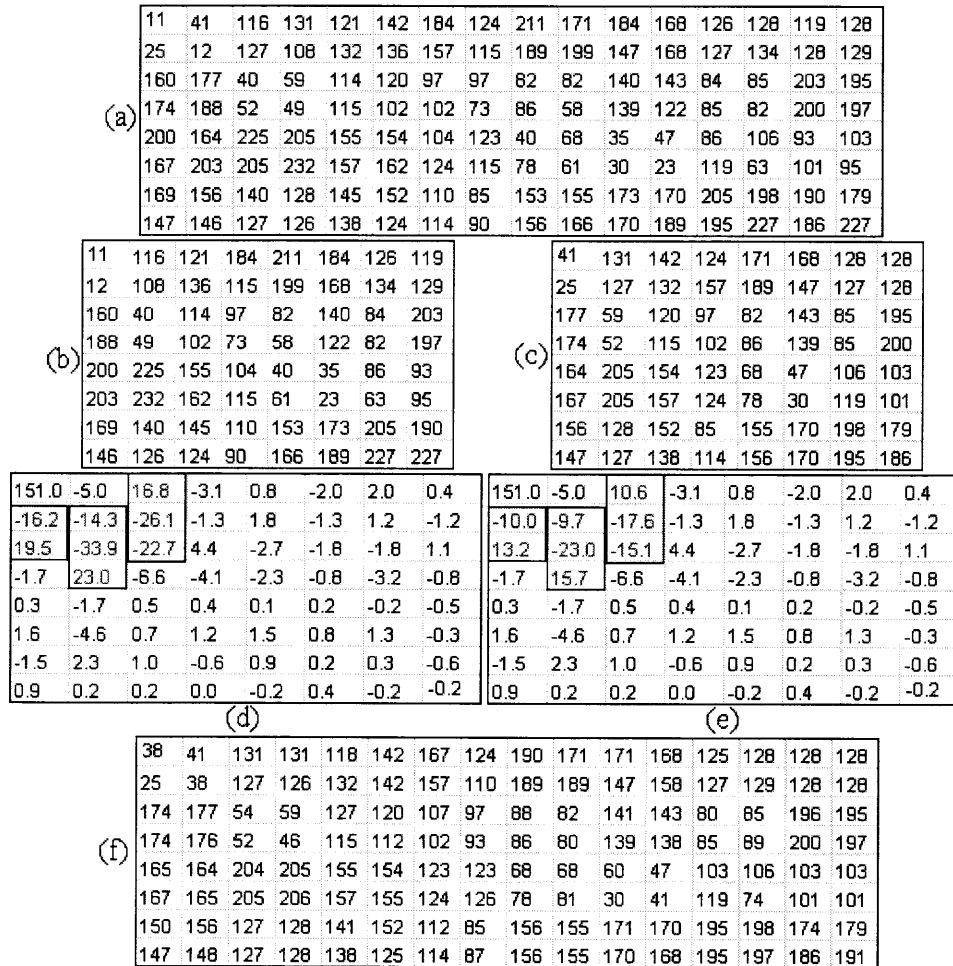
An example of the embedding strategy is given in Figure 2.55. Figure 2.55(a) is the relocated image of Figure 2.49(a) after performing 7 iterations of relocation using the BBCM in Equation (2.1), where the block size is  $2 \times 2$  and  $l = 2$ . After obtaining Figure 2.55(b) from the small box in Figure 2.55(a), Figures 2.55(c) and (d) are generated by the IBPC. Figures 2.55(e) and (f) are respectively obtained from Figures 2.55(c) and (d) by the DCT followed by a division of the quantization table. Note that, Figure 2.55(e) is considered as the container for watermark embedding, and Figure 2.55(f) is the reference register for indicating the embedding positions as colored in gray. Since the number of embedding positions is 8, the watermark is set to be “11111111.” Figure 2.55(g) shows

the result after embedding watermarks into these calculated positions by Equation (2.3), where  $\alpha = 0.5$  and  $RR_{T_h} = 9$ .



**Figure 2.55** An example of the embedding strategy.

Figure 2.56 shows an example of the extracting strategy. Figure 2.56(a) is the watermarked sub-image, and Figures 2.56(b) and (c) are extracted from (a) by the IBPC. Note that, since Figure 2.56(c) is exactly the same as Figure 2.56(d), the reference register is the same as Figure 2.56(f). Therefore, after obtaining Figure 2.56(d) from Figure 2.56(b) by the DCT, the watermark “11111111” can be extracted from the coefficients, and generate Figure 2.56(e). Figure 2.56(f) is the reconstructed result. By observing Figures 2.56(f) and 2.55(b), it is amazing that both of them are the same..

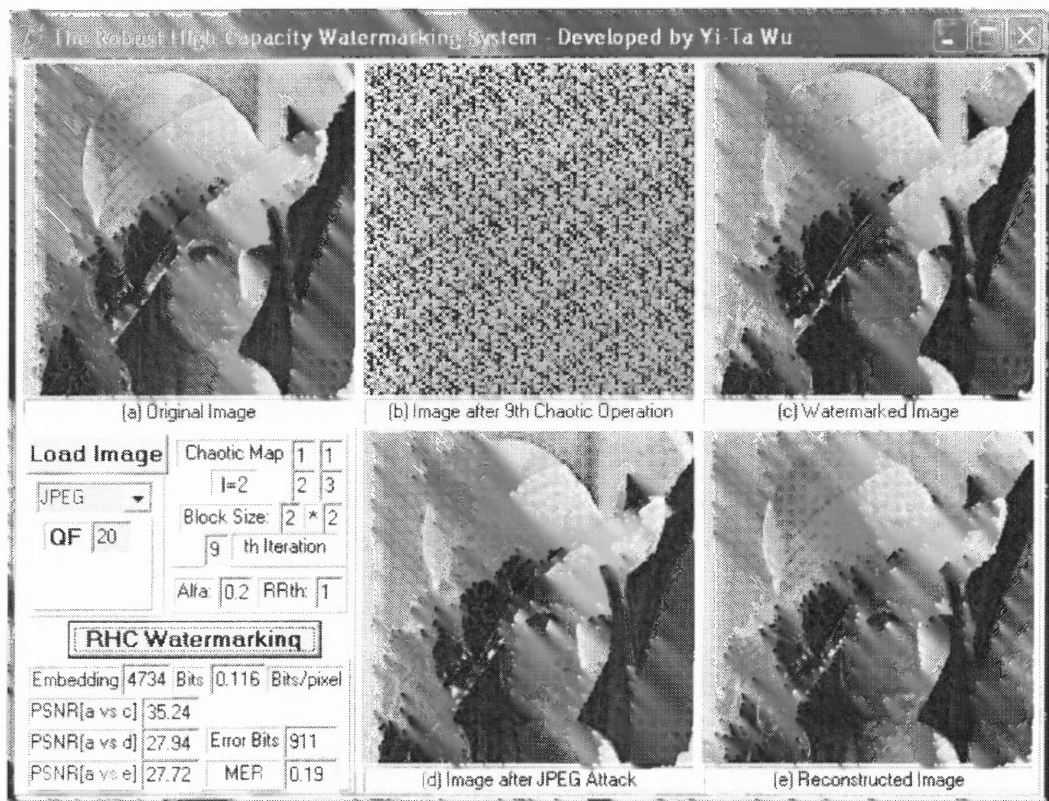


**Figure 2.56** An example of the extracting strategy.

## B. Robustness Experiments

Miller et al. [57] presented a robust high-capacity watermarking by applying informed coding and embedding. They conclude that the watermarking scheme is robust if the message error rate (MER) is lower than 20%. In order to evaluate the robustness of the RHC watermarking, 200 images are used and three kinds of attacks, JPEG compression, Gaussian noise, and low-pass filter, are utilized to the watermarked images. The resulting MERs from these attacked images are provided.

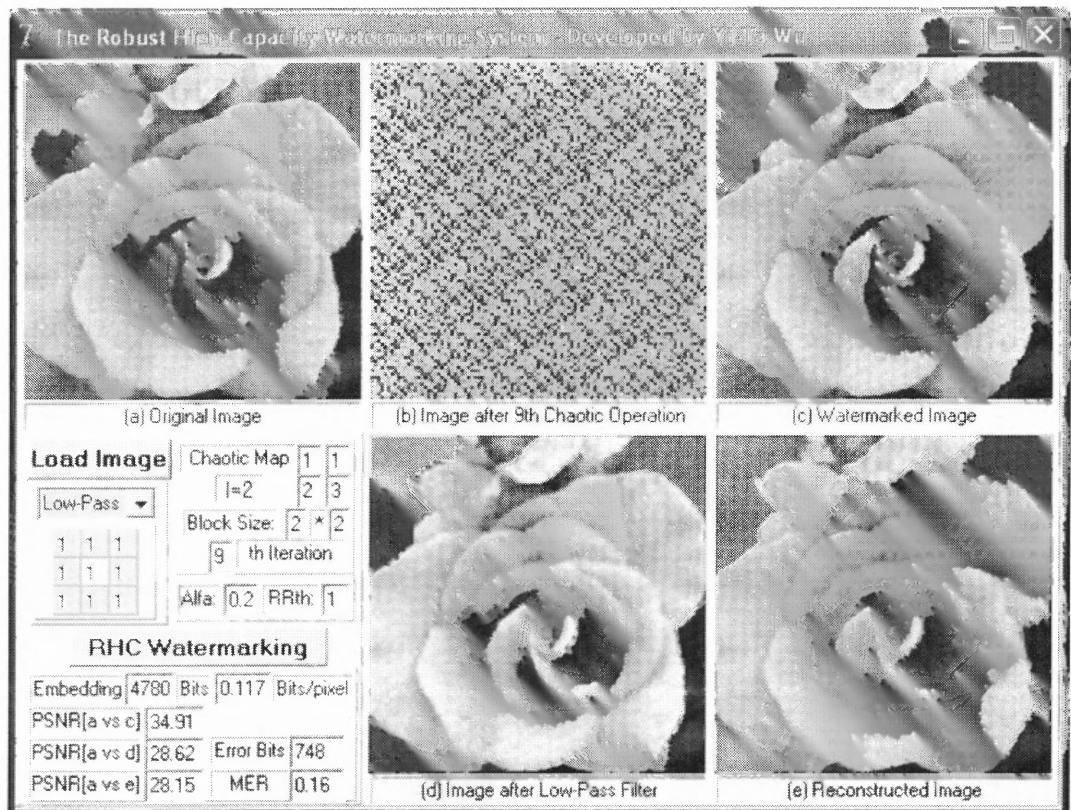
Figure 2.57 shows the robustness experiment under the JPEG compression. Figure 2.57(a) is the original Lena image. Figure 2.57(b) is obtained from Figure 2.57(a) after continuously performing 9 times of BBCM, and is used for embedding watermarks with the following parameters: the block size of  $2 \times 2$ ,  $l = 2$ ,  $\alpha = 0.2$ , and  $RR_{th} = 1$ . Figure 2.57(c) is the watermarked image obtained by continuously performing 8 times of BBCM to Figure 2.57(b). Figure 2.57(d) is the attacked image by applying JPEG compression on Figure 2.57(c) using with quality factor of 20. Figure 2.57(e) is the reconstructed image after removing the embedded watermarks. Note that, the RHC watermarking is not only robust due to that the MER is 0.19 which is less than 20%, but also contains high capacity, 0.116 bits/pixel, which is much larger than 0.015625 bits/pixel presented by Miller et al. [57].



**Figure 2.57** An example of robustness experiment using JPEG compression.



Figures 2.58 and 2.59 show the experimental results under the low-pass filter and Gaussian noise attacks, respectively. The parameters used are the same as the previous example. In Figure 2.58, a  $3 \times 3$  low-pass filter with all 1's is utilized as the attacker. The capacity and the MER are 0.117 and 16%, respectively. In Figure 2.59, a Gaussian noise with the *standard deviation* (SD) 500 and mean 0 is added to the watermarked image. The capacity and the MER are 0.124 and 23%, respectively. It is obviously that the RHC watermarking algorithm not only achieves the robustness requirement, but also maintains the high capacity of watermarks.



**Figure 2.58** An example of robustness experiment using low-pass filter.

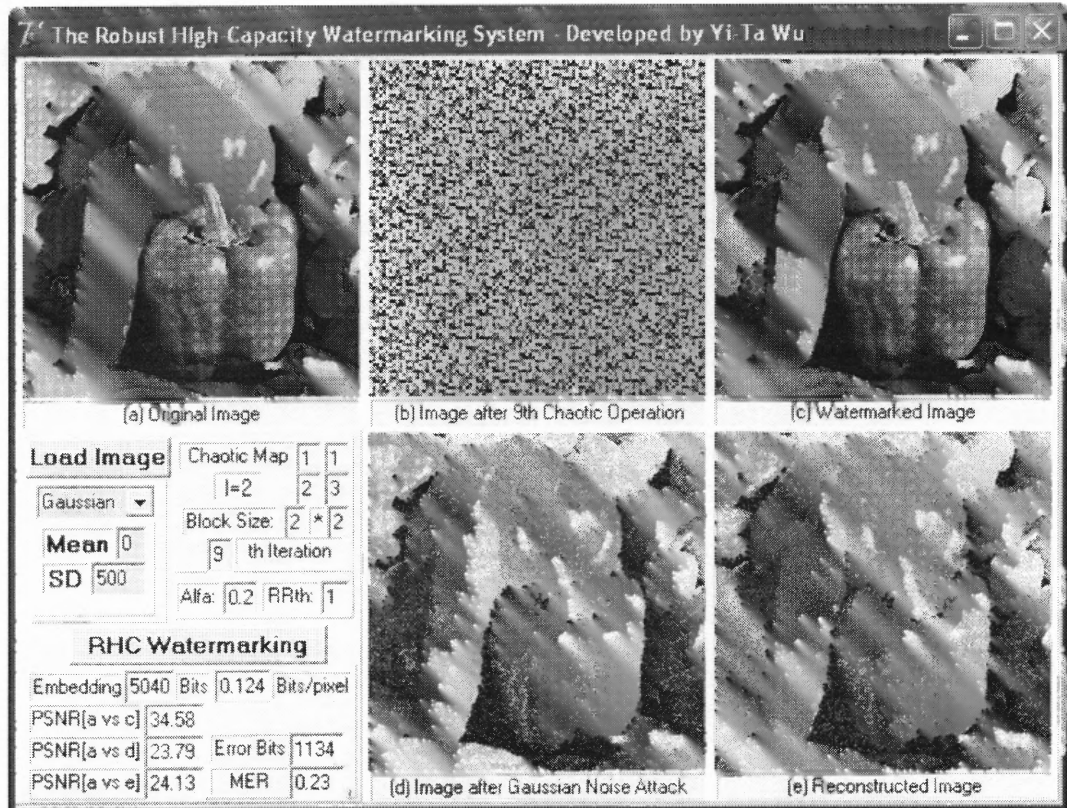


Figure 2.59 An example of robustness experiment using Gaussian noise.

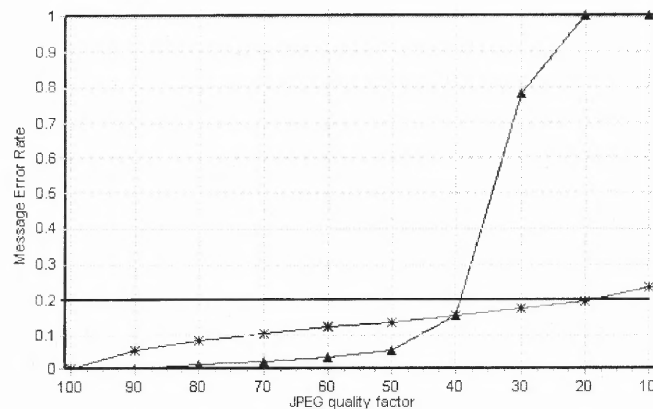
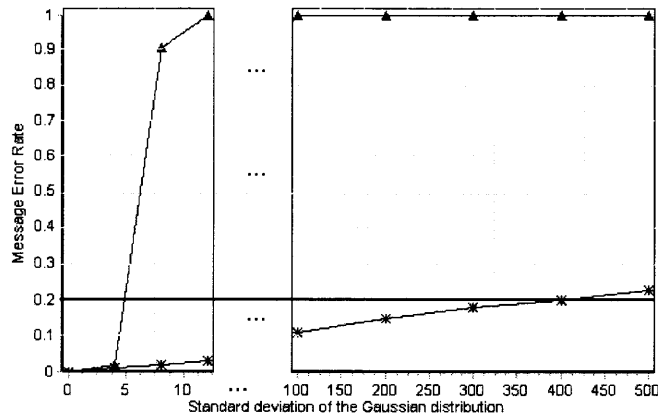


Figure 2.60 Robustness versus JPEG compression.

### C. Performance Comparisons

In this section, the performance of RHC watermarking is evaluated by comparisons with the “iciens” algorithm by Miller et al. [57]. Figure 2.60 shows the effect under the JPEG compression with different quality factors (QFs). For simplicity,

the symbol “\*” is used to represent the RHC and “▲” to represent iciens. Figure 2.61 shows the effect under Gaussian noise with mean 0 and different standard deviations (SDs). The RHC algorithm outperforms the iciens algorithm in terms of low MER. For watermarking capacity, the average capacity of the RHC watermarking is 0.1238 bits/pixel, which is much larger than 0.015625 bits/pixel obtained by Miller’s iciens algorithm.



**Figure 2.61** Robustness versus Gaussian noise.

**Table 2.11** Effect of  $RR_{Th}$

$RR_{Th}$	Capacity (bits)	Capacity (bits/pixel)	JPEG (QF = 20)		Gaussian noise(SD=500,mean=0)	
			Error bits	MER	Error bits	MER
10	810	0.0199	104	0.12	140	0.17
9	924	0.0226	120	0.13	170	0.18
8	1061	0.0260	139	0.13	195	0.18
7	1228	0.0301	163	0.13	228	0.19
6	1439	0.0353	197	0.14	271	0.19
5	1715	0.0420	243	0.14	329	0.19
4	2087	0.0511	304	0.15	403	0.19
3	2617	0.0641	404	0.15	522	0.20
2	3442	0.0844	577	0.18	712	0.21
1	5050	0.1238	968	0.19	1116	0.22

The capacity of the RHC watermarking is affected by the value of  $RR_{th}$  which is the threshold to determine the significant coefficients. The smaller the  $RR_{th}$  is, the higher the capacity is. Table 2.11 shows the effect of  $RR_{th}$  under the JPEG compression with QF=20 and the Gaussian noise with SD=500 and mean=0.

## 2.9 Genetic Algorithm Based Methodology for Breaking the Steganalytic Systems

For steganographic systems, the fundamental requirement is that the stego-object should be perceptually indistinguishable to the degree that it does not raise suspicion. In other words, the hidden information introduces only slight modification to the cover-object. Most passive warden distinguishes the stego-images by analyzing their statistic features.

In general, the steganalytic systems can be categorized into two classes: spatial-domain steganalytic system (SDSS) and frequency-domain steganalytic system (FDSS). The SDSS [3, 102] is adopted for checking the lossless compressed images by analyzing the spatial-domain statistic features. For the lossy compression images such as JPEG, the FDSS [29, 31] is used to analyze the frequency-domain statistic features. Westfeld and Pfitzmann [102] presented two SDSSs based on visual and chi-square attacks. The visual attack uses utilizes human eyes to inspect stego-images by checking their lower bit-planes. The chi-square attack can automatically detect the specific characteristic generated by the least-significant-bit (LSB) steganographic technique. Avcibas et al. [3] proposed image quality measure (IQM) which is based on a hypothesis that the steganographic systems leave statistic evidences that can be exploited for detection using IQM and multivariate regression analysis. Fridrich et al. [31] presented a FDSS for detecting the JPEG stego-images by analyzing their discrete cosine

transformation (DCT) with cropped images. They are utilized to test the GA-based methodology for breaking the steganalytic systems.

Since the steganalytic system analyzes certain statistic features of an image, the idea of developing a robust steganographic system is to generate the stego-image by avoiding changing the statistic features of the cover-image. In literature, several papers have presented the algorithms for steganographic and steganalytic systems. Very few papers have discussed the algorithms for breaking the steganalytic systems. Recently, Chu et al. [17] presented a DCT-based steganographic system by utilizing the similarities of DCT coefficients between the adjacent image blocks where the embedding distortion is spread. Their algorithm can allow random selection of DCT coefficients in order to maintain key statistic features. However, the drawback of their approach is that the capacity of the embedded message is limited, i.e., only 2 bits for an  $8 \times 8$  DCT block.

In this dissertation, a new method for breaking steganalytic systems is developed. The emphasis is shifted from traditionally avoiding the change of statistic features to artificially counterfeiting the statistic features. The algorithm is based on the methodology: in order to manipulate the statistic features for breaking the inspection of steganalytic systems, the GA-based approach is adopted to counterfeit several stego-images (candidates) until one of them can break the inspection of steganalytic systems.

Let  $\xi$  denote a chromosome and  $C$  denote a cover-image. In order to embed messages into DCT-based coefficients and avoid the detection of steganalytic systems, a fitness function is developed to evaluate the following two terms:

- (1) *Analysis*( $\xi, C$ ) - The analysis function evaluates the difference between the cover-image and the stego-image in order to maintain the statistic features. It is

related to the type of the steganalytic systems used, and will be explained Sections 2.9.1 and 2.9.2.

- (2)  $BER(\xi, C)$  - The *bit error rate* (BER) sums up the bit differences between the embedded and the extracted messages. It is defined as

$$BER(\xi, C) = \frac{1}{|Message^H|} \sum_{i=0}^{all\ pixels} |Message_i^H - Message_i^E|, \quad (2.6)$$

where  $Message^H$  and  $Message^E$  denote respectively the embedded and the extracted binary messages, and  $|Message^H|$  is the length of the message. For example, if  $Message^H = 11111$  and  $Message^E = 10101$ , then  $BER(\xi, C) = 0.4$ .

A linear combination of the analysis and the bit error rate is utilized to be the fitness function as

$$Evaluation(\xi, C) = \alpha_1 \times Analysis(\xi, C) + \alpha_2 \times BER(\xi, C), \quad (2.7)$$

where  $\alpha_1$  and  $\alpha_2$  denote weights. The weights can be adjusted according to the user's demand on the degree of distortion to the stego-image or the extracted message.

### 2.9.1 Generating the Stego-image on the Visual Steganalytic System (VSS)

Westfeld and Pfitzmann [102] presented a visual steganalytic system (VSS) that uses an assignment function of color replacement, called the visual filter, to efficiently detect a stego-image by translating a gray-scale image into binary. Therefore, in order to break the VSS, the two results,  $VF^C$  and  $VF^S$ , of applying the visual filter on the cover- and the stego-images respectively should be as identical as possible. The VSS was originally designed to detect the GIF format images by reassigning the color in the color palette of an image. The extension of their method is made to detect the BMP format images as well by setting the odd- and even-numbered gray scales to black and white, respectively.

The  $Analysis(\xi, C)$  to VSS indicating the sum of difference between  $LSB^C$  and  $LSB^S$  is defined as

$$Analysis(\xi, C) = \frac{1}{|C|} \sum_{i=0}^{all\ pixels} (VF_i^C \oplus VF_i^S), \quad (2.8)$$

where  $\oplus$  denotes the Exclusive-OR (XOR) operator. The algorithm is described below.

### **The Algorithm for Generating a Stego-image on VSS:**

1. Divide a cover-image into a set of cover-images of size  $8 \times 8$ .
2. For each  $8 \times 8$  cover-image, a stego-image is generated based on the GA to perform the embedding procedure as well as to ensure that  $LSB^C$  and  $LSB^S$  are as identical as possible.
3. Combine all the  $8 \times 8$  stego-images together to form a complete stego-image.

### **2.9.2 Generating the Stego-image on the IQM-based Steganalytic System**

Avcibas et al. [3] proposed a steganalytic system by analyzing the image quality measures (IQMs) [4] of the cover- and the stego-images. The IQM-SDSS consists of two phases: training and testing. In the training phase, the IQM is calculated between an image and its filtered image using a low-pass filter based on the Gaussian kernel. Suppose there are  $N$  images and  $q$  IQMs in the training set. Let  $x_{ij}$  denote the score in the  $i$ -th image and the  $j$ -th IQM, where  $1 \leq i \leq N$ ,  $1 \leq j \leq q$ . Let  $y_i$  be the value of  $-1$  or  $1$  indicating the cover- or stego-image, respectively. All the images can be represented in the training set as

$$\begin{aligned}
y_1 &= \beta_1 x_{11} + \beta_2 x_{12} + \cdots + \beta_q x_{1q} + \varepsilon_1 \\
y_2 &= \beta_1 x_{21} + \beta_2 x_{22} + \cdots + \beta_q x_{2q} + \varepsilon_2 \\
&\vdots \\
y_N &= \beta_1 x_{N1} + \beta_2 x_{N2} + \cdots + \beta_q x_{Nq} + \varepsilon_N
\end{aligned} \tag{2.9}$$

where  $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_N$  denote random errors in the linear regression model [68]. Then the linear predictor  $\beta = [\beta_1, \beta_2, \dots, \beta_q]$  can be obtained from all the training images.

In the testing phase, the  $q$  IQMs is used to compute  $y_i$  to determine whether it is a stego-image. If  $y_i$  is positive, the test image is a stego-image; otherwise, it is a cover-image.

The IQM-SDSS is first trained in order to obtain the linear predictor, i.e.,  $\beta = [\beta_1, \beta_2, \dots, \beta_q]$ , from the database. Then,  $\beta$  is used to generate the stego-image by the GA-based algorithm, so that it can pass through the inspection of IQM-SDSS. Note that the GA procedure is not used in the training phase. The *Analysis*( $\xi, C$ ) to the IQM-SDSS is defined as

$$\text{Analysis}(\xi, C) = \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q. \tag{2.10}$$

The GA-based algorithm for generating a stego-image to break the IQM-SDSS is presented below.

### **The Algorithm for Generating a Stego-image on IQM-SDSS:**

1. Divide a cover-image into a set of cover-images of size  $8 \times 8$ .
2. Adjust the pixel values in each  $8 \times 8$  cover-image based on the GA to embed messages into the frequency domain and ensure that the stego-image can pass through the inspection of the IQM-SDSS. The procedures for generating the stego-image are presented next after this algorithm.



3. Combine all the  $8 \times 8$  embedded-images together to form a complete embedded image.
4. Test the embedded-image on IQM-SDSS. If it passes, it is the desired stego-image; otherwise, repeat steps 2 to 4.

**The Procedures for Generating an  $8 \times 8$  Embedded-image on IQM-SDSS:**

1. Define the fitness function, the number of genes, the size of population, the crossover rate, the critical value, and the mutation rate.
2. Generate the first generation by a random selection.
3. Generate an  $8 \times 8$  embedded-image based on each chromosome.
4. Evaluate the fitness value for each chromosome by analyzing the  $8 \times 8$  embedded-image.
5. Obtain the better chromosome based on the fitness value.
6. Recombine new chromosomes by crossover.
7. Recombine new chromosomes by mutation.
8. Repeat steps 3 to 8 until a predefined condition is satisfied or a constant number of iteration is reached.

**2.9.3 The GA-based Breaking Algorithm on FDSS**

Fridrich et al. [31] presented a steganalytic system for detecting JPEG stego-images based on the assumption that the histogram distributions of some specific AC DCT coefficients of a cover-image and its cropped image should be similar. Note that, in the DCT coefficients, only the zero frequency (0,0) is the DC component, and the remaining frequencies are the AC components. Let  $h_{kl}(d)$  and  $\bar{h}_{kl}(d)$  respectively denote the total number of AC DCT coefficients in the  $8 \times 8$  cover and its corresponding  $8 \times 8$  cropped images with the absolute value equal to  $d$  at location  $(k, l)$ , where  $0 \leq k, l \leq 7$ .

Note that, the  $8 \times 8$  cropped images, defined in Fridrich et al. [31], are obtained using the same way as the  $8 \times 8$  cover images with a horizontal shift by 4 pixels.

The probability,  $\rho_{kl}$ , of the modification of a non-zero AC coefficient at  $(k, l)$  can be obtained by

$$\rho_{kl} = \frac{\bar{h}_{kl}(1)[h_{kl}(0) - \bar{h}_{kl}(0)] + [h_{kl}(1) - \bar{h}_{kl}(1)][\bar{h}_{kl}(2) - \bar{h}_{kl}(1)]}{[\bar{h}_{kl}(1)]^2 + [\bar{h}_{kl}(2) - \bar{h}_{kl}(1)]^2} \quad (2.11)$$

Note that, the final value of the parameter  $\rho$  is calculated as an average over the selected low-frequency DCT coefficients  $(k, l) \in \{(1,2), (2,1), (2,2)\}$ , and only 0, 1, and 2 are considered when checking the coefficient values of the specific frequencies between a cover and its corresponding cropped image.

The GA-based breaking algorithm on the JFDSS is intended to minimize the differences between the two histograms of a stego-image and its cropped-image. It is presented below.

**Algorithm for Generating a Stego-image on JFDSS:**

1. Compress a cover-image by JPEG and divide it into a set of small cover-images of size  $8 \times 8$ . Each is performed by DCT.
2. Embed the messages into the specific DCT coefficients and decompress the embedded image by IDCT.
3. A  $12 \times 8$  working window is selected and generate an  $8 \times 8$  cropped-image for each  $8 \times 8$  embedded-image.
4. Determine the overlapping area between each  $8 \times 8$  embedded-image and its cropped-image.

5. Adjust the overlapping pixel values by making that the coefficients of some specific frequencies  $(k,l)$  of the stego-image and its cropped-image are as identical as possible and the embedded messages are not altered.
6. Repeat steps 3 to 6 until all the  $8 \times 8$  embedded-images are generated.

Let  $Coef^{Stego}$  and  $Coef^{Crop}$  denote the coefficients of each  $8 \times 8$  stego-image and its cropped-image. The  $Analysis(\xi, C)$  to the JFDSS is defined as

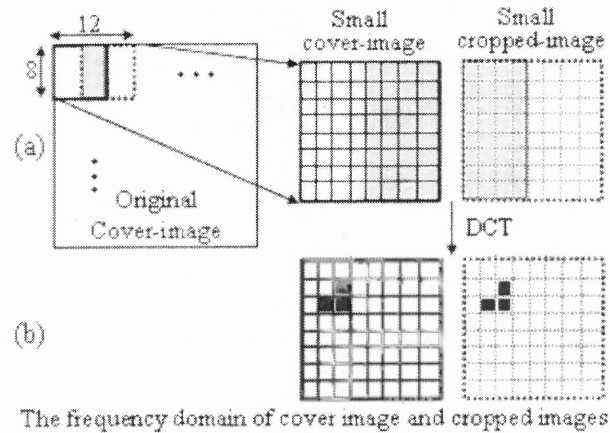
$$Analysis(\xi, C) = \frac{1}{|C|} \sum_{i=0}^{all\ pixels} (Coef_i^{Stego} \otimes Coef_i^{Crop}), \quad (2.12)$$

where  $\otimes$  denotes the operator defined by

$$\begin{cases} Coef_i^{Stego} \otimes Coef_i^{Crop} = 1 & \text{if } (Coef_i^{Stego} = 0 \text{ and } Coef_i^{Crop} \neq 0) \text{ or} \\ & (Coef_i^{Stego} = 1 \text{ and } Coef_i^{Crop} \neq 1) \text{ or} \\ & (Coef_i^{Stego} = 2 \text{ and } Coef_i^{Crop} \neq 2) \text{ or} \\ & (Coef_i^{Stego} \neq 0,1,2 \text{ and } Coef_i^{Crop} = 0,1,2) \\ Coef_i^{Stego} \otimes Coef_i^{Crop} = 0 & \text{otherwise} \end{cases} \quad (2.13)$$

Note that, 0, 1, and 2 denote the values in the specific frequencies obtained by dividing the quantization table. Only the values of the desired frequencies 0, 1, 2 are considered, or some values in Equation (2.13) because of the strategy of JFDSS in Equation (2.11).

Figure 2.62 shows an example of the GA-based algorithm on the JFDSS. In Figure 2.62(a), a  $12 \times 8$  working window is selected for each  $8 \times 8$  stego-image, and generate its  $8 \times 8$  cropped-image. Note that, the shaded pixels indicate their overlapping area, and the three black boxes in Figure 2.62(b) are the desired locations.



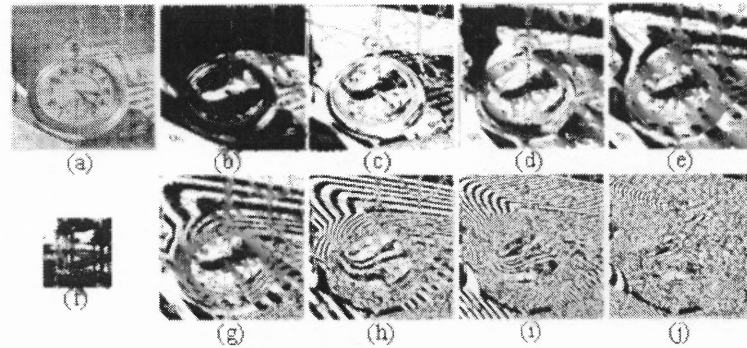
**Figure 2.62** An example of the GA-based algorithm on the JFDSS.

### 2.9.4 Experimental Results

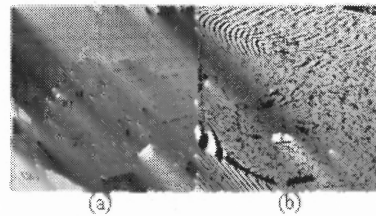
In this section, the experimental results are provided to show that the GA-based steganographic system can successfully break the inspection of steganalytic systems. For testing the algorithm, a database of 200 gray-scale images of size  $256 \times 256$  are used. All the images were originally stored in the BMP format.

#### A. The GA-based Breaking Algorithm on VSS

The algorithm is tested on VSS. Figures 2.63(a) and (f) show a stego-image and a message-image of sizes  $256 \times 256$  and  $64 \times 64$ , respectively. 4 bits of watermarks are embedded into the  $8 \times 8$  DCT coefficients on frequencies (0,2), (1,1), (2,0), and (3,0) for avoiding distortion. Note that, the stego-image in Figure 2.63(a) is generated by embedding Figure 2.63(f) into the DCT coefficients of the cover-image using the GA-based algorithm. Figures 2.63(b) to (j) respectively display the bit-planes from 7 to 0. Figure 2.64 shows the stego-image and its visual filtered result. It is difficult to determine that Figures 2.64(a) is a stego-image.

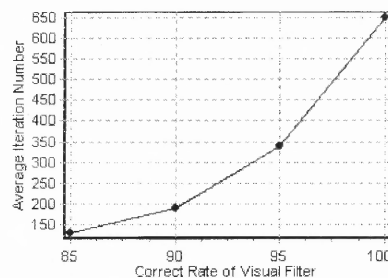


**Figure 2.63** A stego-image generated by the GA-based algorithm and its 8 bit-planes.



**Figure 2.64** A stego-image and its visual filtered result.

Figure 2.65 shows the relationship of the average iteration for adjusting an  $8 \times 8$  cover-image vs. the correct rate of the visual filter. The correct rate is the percentage of similarity between the transformed results of the cover- and the stego-images using the visual filter. Note that, the BERs in Figure 6 are all 100%.



**Figure 2.65** The relationship between the average iteration and the correct rate.

## B. The GA-based Breaking Algorithm on IQM-SDSS

Three stego-images are generated as the training samples for each cover-image by Photoshop plug-in Digimarc [64], Cox's technique [18], and S-tools [12]. Therefore, there are totally 800 images of size  $256 \times 256$ , including 200 cover-images and 600

stego-images. The embedded message size were 1/10, 1/24, and 1/40 of the cover image size for Digimarc, Cox's technique, and S-tools, respectively. Note that the IQM-SDSS [3] can detect the stego-images containing the message size of 1/100 of the cover image. the following four training strategies are developed to obtain the linear predictors as:

- (A) Train all the images in the database to obtain the linear predictor  $\beta^A$ .
- (B) Train 100 cover-images and 100 stego-images to obtain the linear predictor  $\beta^B$ , in which the stego-images are obtained by Cox's technique.
- (C) Train 100 cover-images and 100 stego-images to obtain the linear predictor  $\beta^C$ , in which the stego-images are obtained by Photoshop plug-in Digimarc.
- (D) Train 100 cover-images and 100 stego-images to obtain the linear predictor  $\beta^D$ , in which the stego-images are obtained by S-tools.

In the testing phase, 50 stego-images are generated for each linear predictor. Therefore, 4 types of stego-images  $SI_A$ ,  $SI_B$ ,  $SI_C$ , and  $SI_D$  will be obtained corresponding to  $\beta^A$ ,  $\beta^B$ ,  $\beta^C$ , and  $\beta^D$ , respectively. It is obvious that all the  $SI_A$  images will pass through the inspection of IQM-SDSS with the linear predictor  $\beta^A$ , but may fail with other linear predictors; similarly for the  $SI_B$ ,  $SI_C$ , and  $SI_D$  as well.

**Table 2.12** Experimental Results of GA-based Algorithm on IQM-SDSS

		$\beta^A$	$\beta^B$	$\beta^C$	$\beta^D$	Average
50 $SI_A$ images	FN		94 %	94 %	100 %	96 %
50 $SI_B$ images	FN	84 %		84%	80 %	83 %
50 $SI_C$ images	FN	82 %	86 %		88 %	85 %
50 $SI_D$ images	FN	82 %	86 %	82 %		83 %
50 cover images	FP	6 %	10 %	16 %	16 %	12 %

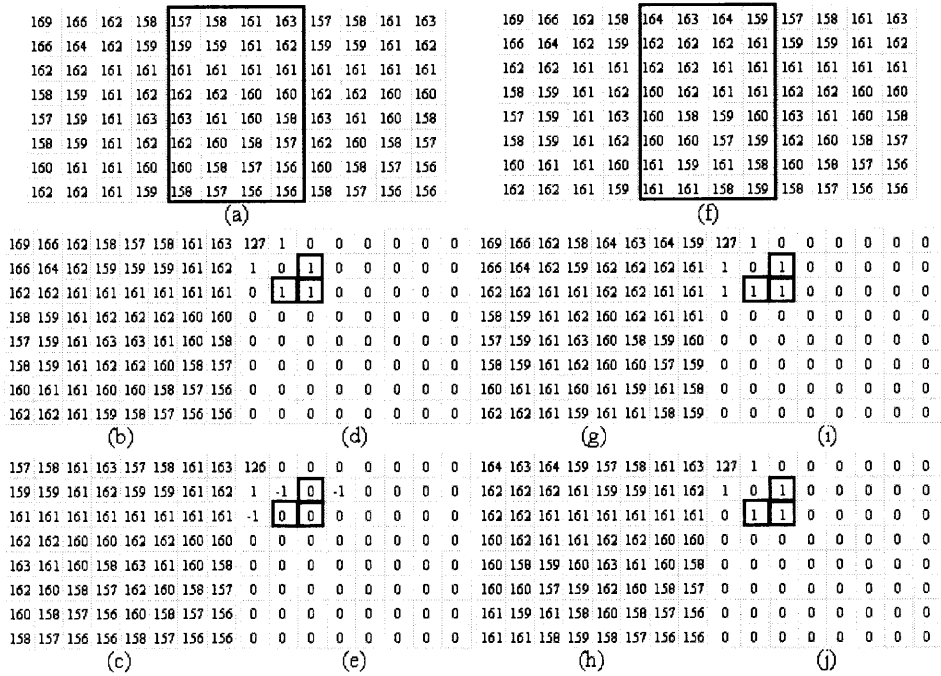
Table 2.12 shows the false negative (Type I Error) and false positive (Type II Error) rates of testing each type of stego- and cover-images under different linear predictors. For example, the false negative (FN) rate of testing 50  $SI_A$  stego-images under  $\beta^B$  is 94% indicating that the  $\beta^B$ -IQMSS falsely decides that 94% of the stego-images belong to the cover-image. On the other hand, the false positive (FP) rate of testing 50 cover-images under  $\beta^B$  is 10% indicating that the  $\beta^B$ -IQMSS decides that 10% of cover-images belong to the stego-images. Note that the cover-images used in the testing phase are different to those used in the training phase.

### C. The GA-based Breaking Algorithm on JFDSS

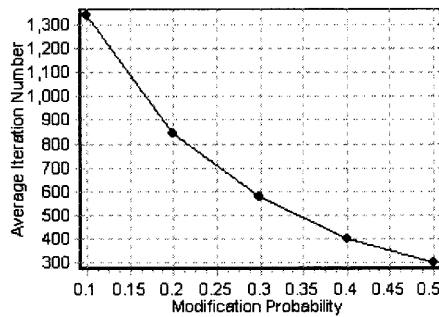
Figure 2.66 shows an example of adjusting an  $8 \times 8$  embedded-image to obtain an  $8 \times 8$  stego-image for breaking the JEDSS. Figure 2.66(a) shows a  $12 \times 8$  working window, where the enclosed is the overlapping area. Figures 2.66(b) and (c) show the original  $8 \times 8$  embedded- and cropped-images, respectively. The value “1” is embedded on (1,2), (2,1), and (2,2) by compressing Figure 2.66(b) using JPEG under 70% compression quality to obtain Figure 2.66(d). Note that, the top-left pixel is (0,0) and the messages can be embedded into any frequency of the transformed domain, so that the embedding capacity could be sufficiently high. Due to that the JFDSS only checks frequencies (1,2), (2,1), and (2,2), an example of embedding 3 bits into these 3 frequencies is provided. Similarly, Figure 2.66(e) is obtained by compressing Figure 2.66(c) using JPEG under the same compression quality. By evaluating the frequencies (1,2), (2,1), and (2,2), the JFDSS can determine whether the embedded-image is a stego-image. Therefore, in order to break the JFDSS, the stego-image is obtained as in Figure 2.66(f). Figures 2.66(g) and (h) respectively show the new embedded- and cropped-images. Similarly, Figures 2.66(i) and

(j) are obtained by compressing Figures 2.66(g) and (h) respectively using JPEG under 70% compression quality. Therefore, the JFDSS cannot distinguish from the frequencies (1,2), (2,1), and (2,2).

Figure 2.67 shows the relationship between the modification probability and the iterations required for adjusting an  $8 \times 8$  embedded-image. Note that a 3-bit message is embedded into frequencies (1,2), (2,1), and (2,2) in each  $8 \times 8$  embedded-image.



**Figure 2.66** An example of adjusting an  $8 \times 8$  embedded-image.



**Figure 2.67** The relationship between the modification probability and the iterations.



## **CHAPTER 3**

### **MORPHOLOGICAL PROCESSING**

Mathematical morphology has been widely used for many applications in image processing and analysis. Most image processing architectures adapted to morphological operations use structuring elements of a limited size. Therefore, difficulties arise dramatically when dealing with a large sized structuring element. In this chapter, a few efficient techniques are presented to decompose arbitrary shapes of big binary structuring elements by applying genetic algorithms, and to decompose arbitrary values of big grayscale structuring elements into dilations or a maximum selection of smaller structuring components. The advantages of the proposed algorithms are not only decomposing arbitrary structuring elements, but also simplifying the computation. Furthermore, the decomposition is suited for a parallel pipelined architecture. The techniques will allow full freedom for users to design any kind and any size of binary and grayscale morphological structuring elements.

#### **3.1 Introduction**

The computation of grayscale dilation and erosion can be implemented very efficiently by using the architecture of threshold decomposition of grayscale morphology into binary morphology [80]. In practical applications, dilation and erosion pairs are used in sequence, either dilation of an image followed by erosion of the dilated result or vice versa. In either way, the result of iteratively applied dilations and erosions is an

elimination of specific image details smaller than the structuring element without the global geometric distortion of unsuppressed features.

Most image processing architectures adapted to morphological operations use structuring elements of a limited size. Implementation becomes difficult when dealing with a large sized structuring element. Hence, the techniques for efficiently decomposing a large structuring element into combined structures of small components are extremely important. Another advantage is the reduction of calculation. For example, if a structuring element of size  $(p+q-1) \times (p+q-1)$  is decomposed into a dilation of two smaller structuring components of sizes  $p \times p$  and  $q \times q$ , the number of calculations required in morphological operations will be reduced from  $(p+q-1)^2$  to  $p^2 + q^2$ . Therefore, the computational time will be reduced dramatically when  $p$  and  $q$  are large.

An optimization algorithm for decomposing binary morphological structuring elements was developed by Zhuang and Haralick [112]. Their algorithm can find the decomposition in which each small structuring element consists of at most two points if such a decomposition exists. Xu [107] presented an algorithm for decomposing convex morphological structuring elements and proved that all convex polygon-shaped morphological structuring elements are decomposable. Kanungo and Haralick [46-48] developed an algorithm for the decomposition of convex structuring elements in a constant time. Park and Chin [62] proposed an algorithm for decomposing limited types of binary structuring elements based on checking the contour of original large structuring elements. If the contour satisfies the defined rules, the decomposition can be achieved. Hashimoto and Barrera [37] showed that some structuring elements, defined as indecomposable in Park and Chin's algorithm, can be decomposable. Hashimoto et. al

[38] presented a method to check whether a structuring element is decomposable or not. If it is decomposable, their algorithm can find a minimum number of subsets. Anelli, Broggi, and Destri [2] developed a technique for decomposing arbitrarily shaped binary structuring elements based on genetic algorithms. However, the random selection of initial population spends more time in processing and their decomposed solutions can be further improved.

For decomposition of grayscale structuring elements, Gader [33] developed an algorithm for decomposing grayscale structuring elements with rectangular support into horizontal and vertical structuring elements. A strategy for decomposing certain types, such as *linear-sloped*, *convex*, and *concave*, of grayscale morphological structuring elements into dilations or maximum selections of small components was explored by Shih and Mitchell [81]. Decomposition of geometric-shaped structuring elements using morphological transformations was developed by Shih and Wu [85]. Sussner and Ritter [95, 96] proposed the decomposition of grayscale morphological templates using the rank method. Sussner, Pardalos, and Ritter [94] proved that the morphological template decomposition problem is an NP-complete problem. A technique using the Top and Umbra operations to find the decomposition of an arbitrary grayscale structuring element was developed by Camps, Kanungo, and Haralick [14].

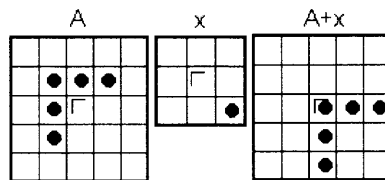
This chapter is organized as follows. Section 3.2 presents the notations and definitions used in the morphological processing. In Section 3.3, the decomposition techniques are presented for arbitrary binary structuring elements. In Section 3.4, the decomposition techniques for arbitrary grayscale structure elements are developed. Experimental results are provided in Section 3.5.

### 3.2 Notations and Definitions in the Mathematical Morphology

The basic binary and grayscale mathematical morphology operations are presented in Sections 3.2.1 and 3.2.2, respectively.

#### 3.2.1 Binary Mathematical Morphology

The theoretical foundation of binary mathematical morphology is set theory. In binary images, those points in the set are called foreground the others in the complement set are called background. The binary mathematical morphology depends extensively on translation and reflection operation as shown below: Figures 3.1 and 3.2 show the example of the translation and reflection, respectively.



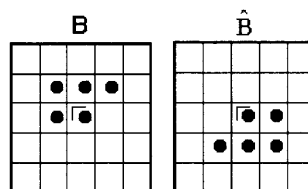
**Figure 3.1** An example of translation.

**Translation:** Given an image  $A$ , the translation of  $A$  by a point  $x$ , denoted by  $A_x$ , is defined by

$$A + x = \{c \mid c = a + x \text{ for } a \in A\} \quad (3.1)$$

**Reflection:** Given an image  $B$ , the reflection of  $B$ , denoted by  $\hat{B}$ , is defined by

$$\hat{B} = \{w \mid w = -b \text{ for } b \in B\} \quad (3.2)$$



**Figure 3.2** An example of reflection.

**3.2.1.1 Binary Dilation and Erosion.** Dilation of a binary image  $A$  by structuring element  $S$ , denoted by  $A \oplus_b S$ , is defined as

$$A \oplus_b S = \{z \mid (\hat{S})_z \cap A \neq \emptyset\} \quad (3.3)$$

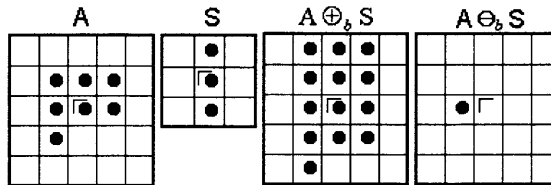
From Equation (3.3), dilation is equivalent to a union of translates of the original image with respect to the structuring element

$$A \oplus_b S = \bigcup_{s \in S} A_s \quad (3.3)$$

Erosion of a binary image  $A$  by structuring element  $S$ , denoted by  $A \ominus_b S$ , is defined as

$$A \ominus_b S = \{z \mid (\hat{S})_z \subseteq A\} \quad (3.4)$$

Figure 3.3 shows an example of the binary dilation and erosion.



**Figure 3.3** An example of binary dilation and erosion.

**3.2.1.2 Binary Opening and Closing.** Opening of a binary image  $A$  by structuring element  $S$ , denoted by  $A \circ_b S$ , is defined as

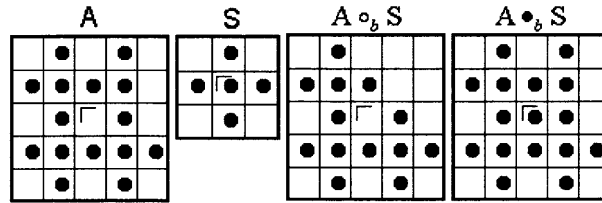
$$A \circ_b S = (A \ominus_b S) \oplus_b S \quad (3.5)$$

Therefore, the opening  $A$  by  $S$  is the erosion of  $A$  by  $S$ , followed by a dilation of the result by  $S$ .

Closing of a binary image  $A$  by structuring element  $S$ , denoted by  $A \bullet_b S$ , is defined as

$$A \bullet_b S = (A \oplus_b S) \ominus_b S \quad (3.6)$$

Therefore, the closing  $A$  by  $S$  is the dilation of  $A$  by  $S$ , followed by an erosion of the result by  $S$ . Figure 3.4 shows an example of the binary opening and closing.



**Figure 3.4** An example of binary opening and closing.

### 3.2.2 Grayscale Mathematical Morphology

A 2D grayscale image can be considered as a set of points in 3D space  $(x, y, f(x, y))$  where  $x$  and  $y$  indicate the row and column coordinates, and  $f(x, y)$  is the grayscale value of the image. By applying the umbra transform  $U$ , a 2D grayscale image can be transformed as a 3D binary image. Therefore, grayscale morphological operations can be regarded as the extension of binary morphological operations to 3D space.

**Umbra Transform:** Given a signal  $f$ , the umbra transform of  $f$ , denoted as  $U[f]$ , is defined as

$$U[f] = \{(x, y, v) \mid x, y \in D_f \text{ and } v \leq f(x, y)\} \quad (3.7)$$

**3.2.2.1 Grayscale Dilation and Erosion.** In grayscale processing, two elementary morphological operations, dilation and erosion, are similar to the convolution operator, except addition/subtraction are substituted for multiplication and maximum/minimum for summation. Let  $f: F \rightarrow E$  and  $k: K \rightarrow E$  denote the grayscale image and the grayscale structuring element, respectively. The symbols,  $E$ ,  $F$  and  $K$  represent the Euclidean space, bounded domain for  $f$ , and bounded domain for  $k$ , respectively. Grayscale dilation of  $f$  by  $k$  is given by

$$(f \oplus_g k)(x) = \max_{x-z \in F, z \in K} \{f(x-z) + k(z)\} \quad (3.8)$$

Grayscale erosion of  $f$  by  $k$  is given by

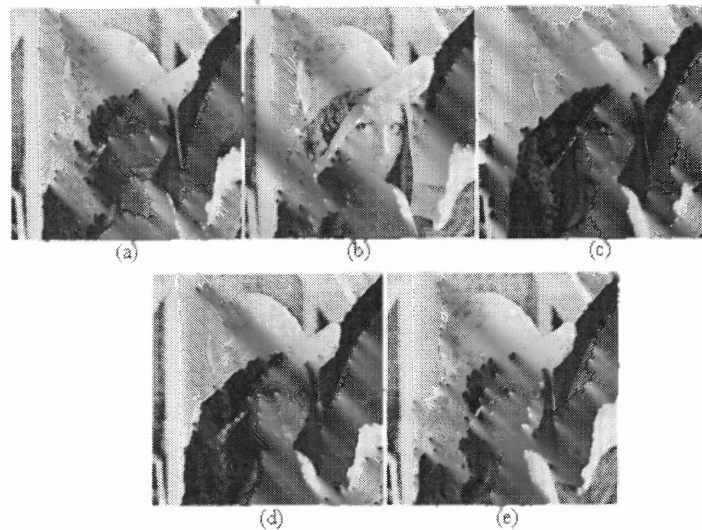
$$(f \ominus_b k)(x) = \min_{x+z \in F, z \in K} \{f(x+z) - k(z)\} \quad (3.9)$$

**3.2.2.2 Grayscale Opening and Closing.** Grayscale opening of an image  $A$  by structuring element  $S$ , denoted by  $A \circ_g S$ , is defined as

$$A \circ_g S = (A \ominus_g S) \oplus_b S \quad (3.10)$$

Closing of a binary image  $A$  by structuring element  $S$ , denoted by  $A \bullet_g S$ , is defined as

$$A \bullet_g S = (A \oplus_g S) \ominus_g S \quad (3.11)$$



**Figure 3.5** An example of grayscale morphological operations.

Figure 3.5 shows an example of grayscale morphological operations. Figure 3.5(a) is the original image. Figures 3.5(b), (c), (d) and (e) are the results after morphological grayscale dilation, erosion, opening and closing, respectively. Note that, the structuring element is size of  $3 \times 3$  with all 1's.

### 3.3 Decomposition of Binary Morphological Structuring Elements

#### 3.3.1 Overview

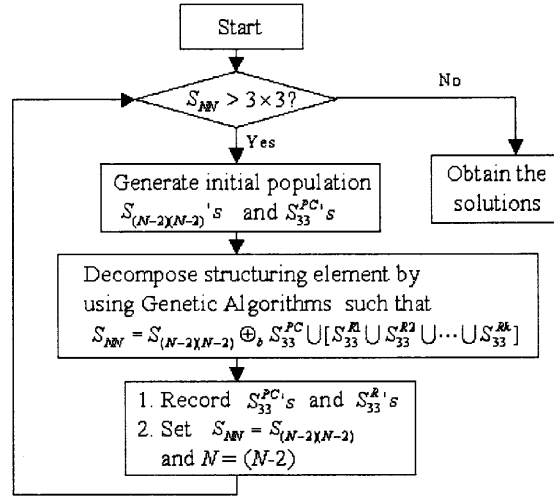
In order to decompose arbitrarily shaped binary structuring elements efficiently, the 13 prime factors from [61] are adopted to generate the initial population and recursively apply genetic algorithms in size reduction. For simplicity, only the odd-sized structuring elements are presented for decomposition, i.e., all even-sized structuring elements are changed into odd-sized ones by adding one additional column and row.

Let  $S_{size}$  denotes a structuring element hereafter in this chapter. Note that, the subscript represents the size of the structuring element. For example,  $S_{N \times N}$  denotes a structuring element of size  $N \times N$ . For simplicity,  $S_{N \times N}$  is replaced by  $S_{NN}$ . Suppose that there exists a structuring element  $S_{NN}$ , where  $N$  is an odd number and  $N \geq 5$ . Two factors,  $S_{(N-2)(N-2)}$  and  $S_{33}$  can be obtained in the first iteration if  $S_{NN} = S_{(N-2)(N-2)} \oplus S_{33}$  or three factors,  $S_{NN}^*$ , with the third size of  $N \times N$  by a union operator if  $S_{NN} = S_{(N-2)(N-2)} \oplus S_{33} \cup S_{NN}^*$ . Note that the dilation receives higher priority in computation than the union and  $S_{NN}^*$  can be decomposed into union of factors of  $3 \times 3$ . After the first decomposition, the same scheme is performed to continuously decompose  $S_{(N-2)(N-2)}$ . The procedure will not be terminated until every factor of the decomposed structuring elements is in the same size of  $3 \times 3$ .

Let  $S_{NN}$  be the original structuring element. By applying genetic algorithms, one  $S_{(N-2)(N-2)}$ , one  $S_{33}^{PC}$ , and several factors  $S_{33}^R$  can be obtained in the first iteration. Then,  $S_{(N-2)(N-2)}$  will be set to be the original structuring element and continually



decompose it until all factors are the same size of  $3 \times 3$ . The procedures are presented below and the flowchart is shown in Figure 3.6. Note that, each  $S_{33}^{PC}$  is obtained from the 13 prime factors in [61] and each  $S_{(N-2)(N-2)}$  is generated by adopting an erosion of  $S_{NN}$  by  $S_{33}^{PC}$ .



**Figure 3.6** The flowchart of recursively decomposing structuring elements.

**Algorithm:**

1. Generate the initial population,  $S_{(N-2)(N-2)}$  and  $S_{33}^{PC}$ , by using the 13 prime factors.

$$S_{NN} = \{ s_{NN}(i, j), 0 \leq i, j < N \}, \text{ where } s_{NN}(i, j) \in \{0, 1\}.$$

$$S_{(N-2)(N-2)} = \{ s_{(N-2)(N-2)}(i, j), 0 \leq i, j < N - 2 \}, \text{ where } s_{(N-2)(N-2)}(i, j) \in \{0, 1\}.$$

$$S_{33}^{PC} = \{ s_{33}^{PC}(i, j), 0 \leq i, j < 3 \}, \text{ where } s_{33}^{PC}(i, j) \in \{0, 1\}.$$

2. After calculating the fitness value of the population, the decomposition,  $S_{(N-2)(N-2)}$ ,

$S_{33}^{PC}$  and some  $S_{33}^R$ 's can be obtained in which

$$S_{NN} = S_{(N-2)(N-2)} \oplus_b S_{33}^{PC} \cup [S_{33}^{R1} \cup S_{33}^{R2} \cup \dots \cup S_{33}^{Rk}].$$

$$S_{33}^{Rl} = \{ s_{33}^{Rl}(i, j), 0 \leq i, j < 3, l \geq 1 \}, \text{ where } s_{33}^{Rl}(i, j) \in \{0, 1\}.$$

3. Record  $S_{33}^{PC}$  and  $S_{33}^R$ 's, set  $S_{NN}$  to be  $S_{(N-2)(N-2)}$  and “ $N$ ” to be “ $N-2$ ”, and further decompose  $S_{NN}$  by repeating steps 2 to 3 until all factors are in the same size of  $3 \times 3$ .

4. Obtain the decomposition  $S_{NN} = \bigcup_{i=1}^m B_i$ , where

$$B_i = SE_{33}^{PC-1} \oplus_b SE_{33}^{PC-2} \oplus_b \dots \oplus_b SE_{33}^{PC-k} \oplus_b SE_{33}^R, \quad k \geq 0, \text{ and } m \text{ and } k \text{ are integers.}$$

Note that, in step 4, the decomposition can be continuously reduced into the form of unions of dilations of smaller structuring elements due to the different properties of unions and intersections discussed in [25], namely

$$S^1 \oplus_b (S^2 \cup S^3) = (S^1 \oplus_b S^2) \cup (S^1 \oplus_b S^3) \quad (3.12)$$

$$S^1 \oplus_b (S^4 \cap S^5) \subseteq (S^1 \oplus_b S^4) \cup (S^1 \oplus_b S^5) \quad (3.13)$$

### 3.3.2 An Example

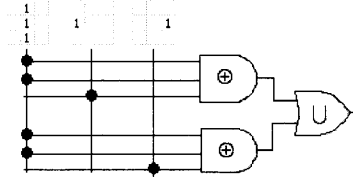
If a big structuring element  $S$  is decomposed into  $S^1 \oplus S^2 \oplus \dots \oplus S^k$ , the dilation and erosion will become

$$\text{Dilation: } A \oplus S = A \oplus (S^1 \oplus S^2 \oplus \dots \oplus S^k) = (\dots((A \oplus S^1) \oplus S^2) \oplus \dots) \oplus S^k \quad (3.14)$$

$$\text{Erosion: } A \ominus S = A \ominus (S^1 \oplus S^2 \oplus \dots \oplus S^k) = (\dots((A \ominus S^1) \ominus S^2) \ominus \dots) \ominus S^k \quad (3.15)$$

Note that Equations (3.12) and (3.13) can be applied to both binary and grayscale morphology. Suppose that the sizes of  $A$  and  $S$  are  $N \times N$  and  $M \times M$ , respectively. The time complexity for the dilation or erosion operator is  $N^2 \times M^2$ . For a parallel pipelined architecture, the structuring element  $S$  is decomposed into another form. That is, a set operator, union, is adopted to enhance the performance. An example of the decomposition is shown below.

$$\begin{aligned}
S &= \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \oplus_b \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \left( \begin{bmatrix} 1 \\ 1 \end{bmatrix} \oplus_b (1 \ 1) \right) \oplus_b \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
&= \left[ \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \oplus_b (1 \ 1 \ 1) \right] \oplus_b \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \\
S &= \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \oplus_b \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \oplus_b 1 \cup \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \oplus_b \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \oplus_b 1
\end{aligned}$$



**Figure 3.7** The parallel pipelined architecture.

In order to be suited for a parallel pipelined architecture, the above decomposition will be continuously decomposed as follows, and its parallel pipelined architecture is shown in Figure 3.7.

### 3.3.3 Decomposition Using Genetic Algorithms

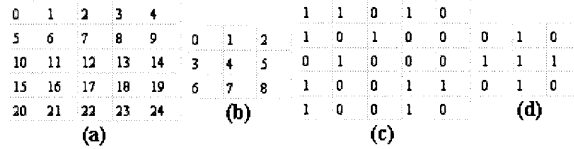
In order to apply GAs in the decomposition, the dynamic varying chromosomes are adopted. That is, the length of chromosomes gradually decreases as the size of structuring elements decreases. Each chromosome consists of two sub-chromosomes. One,  $\xi^F$ , is fixed size and the other,  $\xi^V$ , is variable size. A two-dimensional (2-D) structuring element is converted into a one-dimensional (1-D) string of chromosome. For example, let a structuring element be the size of  $7 \times 7$ . A chromosome  $\xi$  consisting  $\xi^V$  and  $\xi^F$  will be chosen. Note that,  $\xi^V$  and  $\xi^F$  consist of 25 ( $5 \times 5$ ) and 9 ( $3 \times 3$ ) genes, respectively.

$$\xi = \xi^V \parallel \xi^F, \quad \xi^V = g_0 g_1 \cdots g_{24}, \text{ and } \xi^F = g_0 g_1 \cdots g_8$$

where “ $\parallel$ ” indicates concatenation.

Figures 3.8(a) and (b) illustrate the numbering scheme location for  $\xi^V$  ( $g_i(0 \leq i \leq 24)$ ) and  $\xi^F$  ( $g_i(0 \leq i \leq 8)$ ) and their corresponding structuring element is shown in Figures 3.8(c) and (d). The 1-D string of chromosome  $\xi^V$  and  $\xi^F$  are represented as

$$\xi^V = 1101010100010001001110010 \text{ and } \xi^F = 010111010, \text{ respectively.}$$



**Figure 3.8** The positions correspond to the genes.

**3.3.3.1 The Basic Operators and Fitness Function.** For general-purpose serial

systems, the computational complexity of a dilation of two structuring elements is based on the number of elements of the two structuring elements [2]. Therefore, in order to make the decomposition not only work well on the parallel pipelined systems, but also on the serial systems, two fitness functions are used. If a structuring element  $S$  is decomposed into three parts,  $S = S' \oplus S^F \cup S^R$ , the first fitness function can be defined as

$$Evaluation\_1(\xi) = \sum_{i=0}^{all\ pixels} (S_i^\alpha + S_i^F + S_i^R),$$

where  $\xi$  is a chromosome, and  $S'$  and  $S^F$  are the decomposed structuring elements.  $S^R$  is the difference set between  $S$  and  $S' \oplus S^F$ . In fact, the decomposed structuring elements are the same to the sub-chromosomes. That is,  $S' = \xi^V$  and  $S^F = \xi^F$ . Note that, the If  $S = S' \oplus S^F$ , all pixels in  $S^R$  are 0. In other words,  $Evaluation\_1(\xi)$

counts all non-zero pixels of the decomposed structuring elements. The sum of non-zero pixels is aimed to be minimized.

$$Evaluation\_2(\xi) = \sum_{i=0}^{all\ pixels} (S_i - S_i^{New}),$$

where  $S^{New}$  is the dilation of the two decomposed structuring elements, i.e.,  $S' \oplus S^F$ .

In other words,  $Evaluation\_2(\xi)$  counts the difference between the original  $S$  and the new  $S' \oplus F$ . The difference is aimed to be minimized. Two threshold values are selected for each fitness function to determine whether  $\xi$  is good or not. Those good  $\xi$ 's will be adopted for further operations such as reproduction, crossover, and mutation.

### **Reproduction:**

$$Reproduction(\xi) = \{\xi_i, \xi_i \in \Psi, Evaluation(\xi_i) \leq \Omega\},$$

where  $\Omega$  is used to sieve chromosomes and  $\Psi$  is the population. That is, the excellent chromosomes will be reproduced from population because of their high qualities.

### **Crossover:**

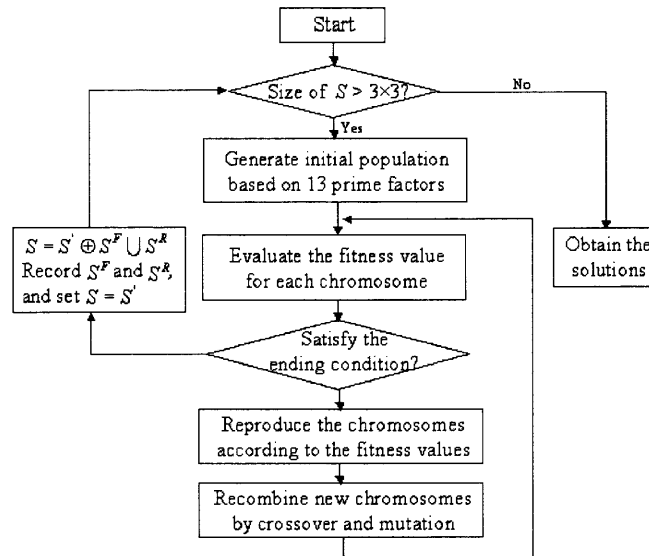
$$Crossover(\xi) = \{\xi_i \varpi \xi_j, \xi_i, \xi_j \in \Psi\},$$

where  $\varpi$  denotes the operation that recombines  $\xi$  by exchanging genes from their parents,  $\xi_i$  and  $\xi_j$ . The most often-used crossovers are one-point, two-point, and multi-point crossovers. Note that, the crossover is only applied to sub-chromosomes. That is, only sub-chromosomes with the same sizes are considered as candidates for performing crossover.

**Mutation:**

$$Mutation(\xi) = \{\xi_i \circ j, \text{ where } 0 \leq j \leq Max\_Length(\xi_i), \xi_i \in \Psi \}$$

where  $\circ$  is the operation by randomly selecting chromosome  $\xi_i$  from  $\Psi$  and randomly changing bits from  $\xi_i$ . There also exist one-point, two-point, and multi-point mutations.



**Figure 3.9** Basic steps in finding the best solutions.

**3.3.3.2 The Proposed Algorithm.** Shih and Wu [85] discovered that some structuring elements, especially those with convex shaped, can be decomposed into dilations of smaller structuring components. Park and Chin [61] developed some rules and 13 prime factors for efficiently checking and obtaining the decomposition. Unfortunately, there exist some types of structuring elements cannot be decomposed by their algorithms. In order to decompose arbitrary-shaped structuring elements, the union operator is adopted. The 13 prime factors are used for obtaining the first population to enhance the performance for those structuring elements which are decomposable without adopting the

union operator. Following is the proposed algorithm and its flowchart is shown in Figure 3.9.

**Algorithm:**

1. Check the size of the input structuring element. If the size is smaller than  $3 \times 3$ , go to step 11.
2. Generate the initial population using the 13 prime factors in [61].
3. Evaluate the fitness value for each chromosome.
4. Check the ending condition whether the value of the fitness function is less than a threshold or the number of iterations is reached. If it is satisfied, go to step 9.
5. Obtain the better chromosomes.
6. Recombine new chromosomes by using crossover.
7. Recombine new chromosomes by using mutation.
8. Go to step 3.
9. Record the decomposed structuring elements  $S^F$  and  $S^R$ , and set  $S = S'$ .
10. Go to step 1.
11. Output all of the decomposed structuring elements  $S^F$ 's and  $S^R$ 's.

### 3.4 Decomposition of Grayscale Morphological Structuring Elements

The structuring element decomposition properties described in this section are suited for binary (with two values, “0” indicating background and “1” indicating foreground) and gray-scale images.

**Property 3.1:** If a structuring element  $k$  can be decomposed into sequential dilations of smaller structuring components, then a dilation (erosion, respectively) of an image  $f$  by  $k$  is equal to successively applied dilations (erosions, respectively) of the previous stage output by these structuring components. That is, if  $k = k_1 \oplus k_2 \oplus \dots \oplus k_n$ , then

$$f \oplus k = (\dots((f \oplus k_1) \oplus k_2) \oplus \dots) \oplus k_n \text{ and } f \ominus k = (\dots((f \ominus k_1) \ominus k_2) \ominus \dots) \ominus k_n \quad (3.16)$$

**Property 3.2:** If a structuring element  $k$  can be decomposed into a maximum selection over several structuring components, then a dilation (erosion, respectively) of an image  $f$  by  $k$  is equal to finding a maximum (minimum, respectively) over each applied dilation (erosion, respectively) of the image by these structuring components. That is, if  $k = \max(k_1, k_2, \dots, k_n)$ , then

$$\begin{aligned} f \oplus k &= \max(f \oplus k_1, f \oplus k_2, \dots, f \oplus k_n) \text{ and} \\ f \ominus k &= \min(f \ominus k_1, f \ominus k_2, \dots, f \ominus k_n) \end{aligned} \quad (3.17)$$

**Property 3.3:** Let  $f_{(x)}$  denote an image  $f$  being shifted by  $x$ . The dilation and erosion satisfy the translation invariance property as

$$f_{(x)} \oplus k = (f \oplus k)_{(x)} \text{ and } f_{(x)} \ominus k = (f \ominus k)_{(x)} \quad (3.18)$$

**Property 3.4:** Let  $u$  denote a constant. The dilation and erosion satisfy

$$f \oplus k = (f(x) - u) \oplus (k(z) + u) \text{ and } f \ominus k = (f(x) - u) \ominus (k(z) - u) \quad (3.19)$$



### 3.4.1 Decomposition of 1-D Arbitrary Grayscale Structuring Elements

The decomposition of certain types of structuring elements such as linearly-sloped, eccentrically convex, and eccentrically concave, was developed by Shih and Mitchell [12]. Furthermore, the decomposition of certain geometric-shaped structuring elements such as sine, cosine, semiellipse, parabola, semihyperbola and Gaussian, was presented by Shih and Wu [13]. In this section, a general decomposition strategy is introduced to decompose 1-D arbitrary gray-scale structuring elements.

Let the origin of the structuring element be placed at the leftmost point. The decomposition of a structuring element of odd size  $n$  can be expressed as

$$[a_1, a_2, a_3, \dots, a_n] = [b_1^1, b_2^1, b_3^1] \oplus [b_1^2, b_2^2, b_3^2] \oplus \dots \oplus [b_1^{(n-1)/2}, b_2^{(n-1)/2}, b_3^{(n-1)/2}], \quad (3.20)$$

and the decomposition of a structuring element of even size  $m$  can be expressed as

$$[a_1, a_2, a_3, \dots, a_m] = [c_1, c_2] \oplus [b_1^1, b_2^1, b_3^1] \oplus [b_1^2, b_2^2, b_3^2] \oplus \dots \oplus [b_1^{(m-2)/2}, b_2^{(m-2)/2}, b_3^{(m-2)/2}] \quad (3.21)$$

where  $a_i$ ,  $b_j^k$  and  $c_i$  denote factors in the structuring elements. Note that,  $b_j^k$  indicates the  $j$ -th factor in the  $k$ -th structuring element, where  $j$  ( $1 \leq j \leq 3$ ) and  $k$  ( $1 \leq k \leq (n-1)/2$ ) are integers. For example, a structuring element of size 7 will be decomposed into dilations of three small structuring elements of size 3, and  $b_2^1$  will be the second factor in the first small structuring element.

**Proposition 3.1:** If a structuring element satisfies the following two conditions, it can be decomposed by Equations (3.20) and (3.21). Let  $S_N = [a_1, a_2, a_3, \dots, a_c, \dots, a_N]$ , where  $a_c$  denotes an apex element, and  $N$  denotes an odd number. The two conditions are:

(a)  $a_1 \leq a_2 \leq a_3 \leq \dots \leq a_c$  and  $a_c \geq a_{c+1} \geq \dots \geq a_N$

(b)  $a_2 - a_1 \geq a_3 - a_2 \geq \dots \geq a_c - a_{c-1}$  and  $a_c - a_{c+1} \leq a_{c+1} - a_{c+2} \leq \dots \leq a_{N-1} - a_N$ .

**Proof:** Let  $S_N$  be decomposed into  $S_{N-2}$  and  $S_3$ , where  $S_{N-2} = [a_1, a_2, a_3, \dots, a_{N-2}]$  obtained by duplicating the first  $N-2$  elements in  $S_N$ , and  $S_3 = [c_1, c_2, c_3]$  is computed using  $c_1 = 0$ ,  $c_2 = a_{N-1} - a_{N-2}$  and  $c_3 = a_N - a_{N-2}$ . The proof will be considered that by choosing  $S_{N-2}$  and  $S_3$  in this way, the equality of  $S_N = S_{N-2} \oplus S_3$  holds. Let  $a_{y-2}$ ,  $a_{y-1}$ , and  $a_y$  denote any three adjacent elements in  $S_{N-2}$ . By looking at the location of  $a_y$  in calculating  $S_{N-2} \oplus S_3$  as shown in Figure 3.10, the three values are obtained:  $a_y$ ,  $(a_{N-1} - a_{N-2}) + a_{y-1}$ , and  $(a_N - a_{N-2}) + a_{y-2}$ . It is necessary to prove that  $a_y$  is the maximum value. Since  $0$ ,  $(a_{N-1} - a_{N-2}) + (a_{y-1} - a_y)$ , and  $(a_N - a_{N-2}) + (a_{y-2} - a_y)$  can be obtained by subtracting  $a_y$  from the three values, the proof can be made that  $(a_{N-1} - a_{N-2}) + (a_{y-1} - a_y) \leq 0$  and  $(a_N - a_{N-2}) + (a_{y-2} - a_y) \leq 0$  in the following three cases:

(1) If  $a_{y-2}$  is on the right hand side of  $a_c$  (i.e.,  $a_{N-2}$  is on the right hand side of  $a_c$ ):

From condition (a), it is obviously that  $a_c \geq a_{y-2} \geq a_{y-1} \geq a_y \geq \dots \geq a_{N-2} \geq a_{N-1} \geq a_N$ .

Combining with condition (b),  $a_{y-1} - a_y \leq a_{N-2} - a_{N-1}$  and  $a_{y-2} - a_{y-1} \leq a_{N-1} - a_N$

can be obtained. The summation of the above two inequalities gives  $a_{y-2} - a_y \leq$

$a_{N-2} - a_N$ . Therefore, the conclusion can be made that  $(a_{N-1} - a_{N-2}) + (a_{y-1} - a_y) \leq 0$

and  $(a_N - a_{N-2}) + (a_{y-2} - a_y) \leq 0$ .

(2) If  $a_y$  and  $a_N$  are both on the left hand side of  $a_c$ :

$a_{y-1} - a_y \leq 0$  and  $a_{y-2} - a_y \leq 0$  can be obtained because

$a_1 \leq \dots \leq a_{y-2} \leq a_{y-1} \leq a_y \leq a_c$ .  $a_N - a_{N-1} \leq a_y - a_{y-1}$  and  $a_N - a_{N-2} \leq a_y - a_{y-2}$  can

be obtained because  $a_2 - a_1 \geq a_3 - a_2 \geq \dots \geq a_c - a_{c-1}$  and

$a_1 \leq \dots \leq a_{y-2} \leq a_{y-1} \leq a_y \leq \dots \leq a_{N-2} \leq a_{N-1} \leq a_N \leq a_c$ . Therefore, the conclusion can

be made that  $(a_{N-1} - a_{N-2}) + (a_{y-1} - a_y) \leq 0$  and  $(a_N - a_{N-2}) + (a_{y-2} - a_y) \leq 0$ .

(3) If  $a_y$  is on the left hand side of  $a_c$ , but  $a_{N-2}$  is on the right hand side of  $a_c$ :

$a_{N-1} - a_{N-2} \leq 0$  and  $a_N - a_{N-2} \leq 0$  can be obtained because  $a_c \geq a_{c+1} \geq \dots \geq a_N$ .

$a_{y-1} - a_y \leq 0$  and  $a_{y-2} - a_y \leq 0$  is obtained because  $a_1 \leq \dots \leq a_{y-2} \leq a_{y-1} \leq a_y \leq a_c$ .

Therefore, the conclusion can be made that  $(a_{N-1} - a_{N-2}) + (a_{y-1} - a_y) \leq 0$  and

$(a_N - a_{N-2}) + (a_{y-2} - a_y) \leq 0$ .

$$\begin{aligned}
 & \mathcal{S}_3 = \left[ \begin{array}{c|c|c} C_3 & & C_2 \\ \hline a_N - a_{N-2} & & a_{N-1} - a_{N-2} \\ \hline a_{y-2} & & a_{y-1} \\ \hline C_1 & & \\ \hline 0 & & a_y, \dots, a_{N-2} \end{array} \right] \\
 \text{(a) In element } a_y : \mathcal{S}_{N-2} &= [ a_1, a_2, \dots, \begin{array}{|c|} \hline a_{y-2} \\ \hline \end{array}, \begin{array}{|c|} \hline a_{y-1} \\ \hline \end{array}, \begin{array}{|c|} \hline 0 \\ \hline \end{array}, \dots, a_{N-2} ] \\
 \text{(b) Dilation Result : } \max & \left\{ \begin{array}{l} a_{y-2} + (a_N - a_{N-2}), \\ a_{y-1} + (a_{N-1} - a_{N-2}), \\ a_y + 0 \end{array} \right\} = \max \left\{ \begin{array}{l} (a_{y-2} - a_y) + (a_N - a_{N-2}), \\ (a_{y-1} - a_y) + (a_{N-1} - a_{N-2}), \\ 0 \end{array} \right\}
 \end{aligned}$$

**Figure 3.10** Dilation of a structuring element verifying proposition 1.

The procedure of continuously decomposing  $\mathcal{S}_{N-2}$  will not be terminated until the decomposed size is 3. Note that the two conditions listed in Proposition 1 are named as *convex eccentrically decreasing* in [13]. If a structuring element belongs to this type, it can be decomposed by Equations (3.20) and (3.21). However, the decomposition strategy aimed for arbitrary gray-scale structuring elements is different from the one in [13] which is designed for certain geometric-shaped structuring elements.

For the arbitrary gray-scale structuring elements, the proposed algorithm can decompose the structuring element into the formats as shown in Equations (3.22) and (3.23), where  $n$  and  $m$  denote odd and even numbers respectively, and “ $\times$ ” denotes *don't*

care. That is, the assumptions (a) and (b) of proposition 3.1 are embedded in the more general case dealing with arbitrary gray-scale structuring elements. By applying the proposed algorithm for decomposing arbitrary gray-scale structuring elements, if a structuring element satisfies assumptions (a) and (b), then the dilated result of the two decomposed components will be the same as the original structuring element, so the maximum operation is not needed. Therefore, it is equivalent to equations 3-20 and 3-21 in this case. Note that “ $\times$ ” is equivalent to “ $-\infty$ ” in mathematical morphology.

$$[a_1, a_2, a_3, \dots, a_n] = \tag{3.22}$$

$$\max \left\{ \begin{array}{l} ([b_1^{(1,1)}, b_2^{(1,1)}, b_3^{(1,1)}] \oplus [b_1^{(1,2)}, b_2^{(1,2)}, b_3^{(1,2)}] \oplus \dots \oplus [b_1^{(1,(n-1)/2)}, b_2^{(1,(n-1)/2)}, b_3^{(1,(n-1)/2)}])_{(0)}, \\ ([b_1^{(2,1)}, b_2^{(2,1)}, b_3^{(2,1)}] \oplus [b_1^{(2,2)}, b_2^{(2,2)}, b_3^{(2,2)}] \oplus \dots \oplus [b_1^{(2,(n-3)/2)}, b_2^{(2,(n-3)/2)}, b_3^{(2,(n-3)/2)}])_{(2)}, \\ \vdots \\ ([b_1^{((n-1)/2,1)}, b_2^{((n-1)/2,1)}, b_3^{((n-1)/2,1)}])_{(n-3)} \end{array} \right\}$$

$$[a_1, a_2, a_3, \dots, a_m] = \tag{3.23}$$

$$\max \left\{ \begin{array}{l} ([c_1, c_2] \oplus [b_1^{(1,1)}, b_2^{(1,1)}, b_3^{(1,1)}] \oplus [b_1^{(1,2)}, b_2^{(1,2)}, b_3^{(1,2)}] \oplus \dots \oplus [b_1^{(1,(m-2)/2)}, b_2^{(1,(m-2)/2)}, b_3^{(1,(m-2)/2)}])_{(0)}, \\ ([b_1^{(2,1)}, b_2^{(2,1)}, b_3^{(2,1)}] \oplus [b_1^{(2,2)}, b_2^{(2,2)}, b_3^{(2,2)}] \oplus \dots \oplus [b_1^{(2,(m-2)/2)}, b_2^{(2,(m-2)/2)}, b_3^{(2,(m-2)/2)}])_{(0)}, \\ ([b_1^{(3,1)}, b_2^{(3,1)}, b_3^{(3,1)}] \oplus [b_1^{(3,2)}, b_2^{(3,2)}, b_3^{(3,2)}] \oplus \dots \oplus [b_1^{(3,(m-4)/2)}, b_2^{(3,(m-4)/2)}, b_3^{(3,(m-4)/2)}])_{(2)}, \\ \vdots \\ ([b_1^{(m/2,1)}, b_2^{(m/2,1)}, b_3^{(m/2,1)}])_{(m-4)}, \\ ([\times, \times, e])_{(m-3)} \end{array} \right\}$$

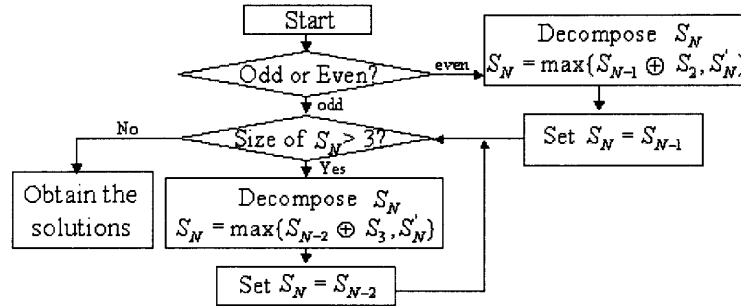
where  $e$ ,  $c_1$ ,  $c_2$ , and  $b_j^{(t,k)}$  denote the factors in the structuring elements. Note that,

$b_j^{(t,k)}$  indicates the  $j$ -th factor in the  $k$ -th structuring element of the  $t$ -th decomposition.

Let  $S_N$  denote a structuring element of size  $N$ . The proposed algorithm for the 1-D structuring element decomposition is presented below, and its flowchart is shown in Figure 3.11.

**The 1-D Decomposition Algorithm:**

1. If  $S_N$  is odd sized, go to step 5.
2. Decompose  $S_N = [a_1, a_2, a_3, \dots, a_N]$  into  $S_{N-1}$  and  $S_2$ , where  
 $S_{N-1} = [a_1, a_2, a_3, \dots, a_{N-1}]$  is obtained by duplicating the N-1 elements in  $S_N$ , and  
 $S_2 = [d_1, d_2]$  is computed using  $d_1 = 0$  and  $d_2 = a_N - a_{N-1}$ .
3. Compare  $S_N$  and  $S_{N-1} \oplus S_2$ . If they are equal,  $S_N = S_{N-1} \oplus S_2$ ; otherwise,  
 $S_N = \max\{S_{N-1} \oplus [0, \times], [\times, a_N]_{(N-2)}\}$ . Note that, unlike some existing algorithms, the proposed algorithm does not need to check the type of the structuring element before decomposing the structuring element. In fact, the result of comparing  $S_N$  and  $S_{N-1} \oplus S_2$  implies whether a structuring element is decomposable or not.
4. Set  $S_N = S_{N-1}$ .
5. If  $N \leq 3$ , go to step 9.
6. Decompose  $S_N = [a_1, a_2, a_3, \dots, a_N]$  into  $S_{N-2}$  and  $S_3$ , where  
 $S_{N-2} = [a_1, a_2, a_3, \dots, a_{N-2}]$  is obtained by duplicating the N-2 elements in  $S_N$ , and  
 $S_3 = [c_1, c_2, c_3]$  is computed using  $c_1 = 0$ ,  $c_2 = a_{N-1} - a_{N-2}$  and  $c_3 = a_N - a_{N-2}$ .
7. Compare  $S_N$  and  $S_{N-2} \oplus S_3$ . If they are equal,  $S_N = S_{N-2} \oplus S_3$ ; otherwise,  
 $S_N = \max\{S_{N-2} \oplus [0, 0, \times], [\times, a_{N-1}, a_{N-2}]_{(N-3)}\}$ .
8. Set  $S_N = S_{N-2}$ , and go to step 5.
9. Obtain the result of the decomposition. Let m and k be integers and  $1 \leq k \leq m$ . For even-sized structuring elements,  $S_N = \max\{Q_k, P_k\}$ , and for odd-sized,  
 $S_N = \max\{P_k\}$ , where  $P_k = S_3^1 \oplus S_3^2 \oplus \dots \oplus S_3^k$  and  
 $Q_k = S_2 \oplus S_3^1 \oplus S_3^2 \oplus \dots \oplus S_3^k$ .



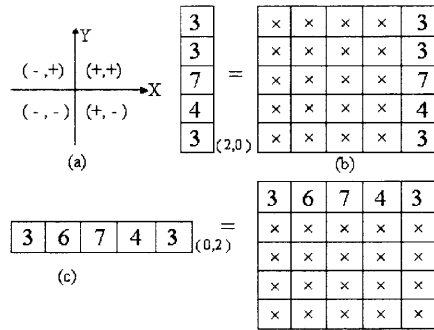
**Figure 3.11** The flowchart of the 1-D structuring element decomposition.

### 3.4.2 Decomposition of 2-D Arbitrary Grayscale Structuring Elements

The 2-D arbitrary gray-scale structuring elements can be decomposed by two methods. One method is to decompose them into a set of 1-D row or column structuring elements, which can be continuously decomposed using the aforementioned technique in Section 3.4.1. Figure 3.12 illustrates the row decomposition, where the row structuring elements are shifted by (x, y). The translation is explained in Figure 3.13, where (a) describes the coordinate system, (b) is a translation by (2, 0), and (c) is a translation by (0, 2).

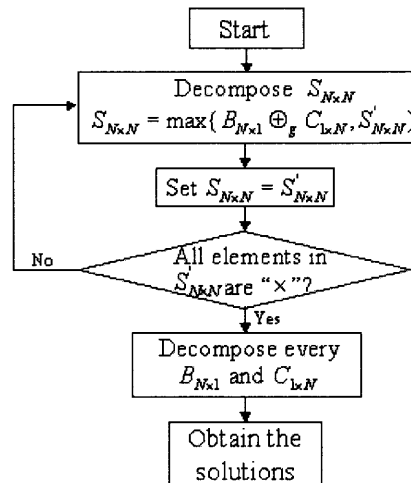
$$\begin{array}{|c|c|c|c|c|} \hline 3 & 6 & 7 & 4 & 3 \\ \hline 7 & 8 & 9 & 10 & 3 \\ \hline 8 & 9 & 11 & 10 & 7 \\ \hline 5 & 8 & 10 & 9 & 4 \\ \hline 5 & 5 & 5 & 5 & 3 \\ \hline \end{array} = \max \left\{ \begin{array}{l} \left( \begin{array}{|c|c|c|c|c|} \hline 3 & 6 & 7 & 4 & 3 \\ \hline \end{array} \right)_{(0,2)}, \left( \begin{array}{|c|c|c|c|c|} \hline 7 & 8 & 9 & 10 & 3 \\ \hline \end{array} \right)_{(0,1)}, \\ \left( \begin{array}{|c|c|c|c|c|} \hline 8 & 9 & 11 & 10 & 7 \\ \hline \end{array} \right)_{(0,0)}, \left( \begin{array}{|c|c|c|c|c|} \hline 5 & 8 & 10 & 9 & 4 \\ \hline \end{array} \right)_{(0,-1)}, \\ \left( \begin{array}{|c|c|c|c|c|} \hline 5 & 5 & 5 & 5 & 3 \\ \hline \end{array} \right)_{(0,-2)} \end{array} \right\} \\
 = \max \left\{ \begin{array}{l} \left( \begin{array}{|c|c|c|} \hline 1 & 4 & 5 \\ \hline \end{array} \oplus \begin{array}{|c|c|c|} \hline 2 & -1 & -2 \\ \hline \end{array} \right)_{(0,2)}, \left( \begin{array}{|c|c|c|} \hline 3 & 4 & 5 \\ \hline \end{array} \oplus \begin{array}{|c|c|c|} \hline 4 & 5 & -2 \\ \hline \end{array} \right)_{(0,1)}, \\ \left( \begin{array}{|c|c|c|} \hline 4 & 5 & 7 \\ \hline \end{array} \oplus \begin{array}{|c|c|c|} \hline 4 & 3 & 0 \\ \hline \end{array} \right)_{(0,0)}, \left( \begin{array}{|c|c|c|} \hline 2 & 5 & 7 \\ \hline \end{array} \oplus \begin{array}{|c|c|c|} \hline 3 & 2 & -3 \\ \hline \end{array} \right)_{(0,-1)}, \\ \left( \begin{array}{|c|c|c|} \hline 2 & 2 & 2 \\ \hline \end{array} \oplus \begin{array}{|c|c|c|} \hline 3 & 3 & 1 \\ \hline \end{array} \right)_{(0,-2)} \end{array} \right\}$$

**Figure 3.12** The row decomposition of a 2-D structuring element.



**Figure 3.13** The representation of the translation.

Another method, called the *combined row-and-column* approach, for the decomposition of 2-D gray-scale structuring elements is presented below, which iteratively decomposes a 2-D gray-scale structuring element into a dilation of 1-D row and 1-D column ones until the original structuring element can be reconstructed by the union of each dilation of pair of row and column 1-D gray-scale structuring elements. Its flowchart is shown in Figure 3.14. Let  $S_{N \times N}$  denote a 2-D structuring element of size  $N \times N$ , and let  $B_{N \times 1}$  and  $C_{1 \times N}$  denote the 1-D row and column structuring elements of size  $N$ , respectively.



**Figure 3.14** The flowchart of the combined row-and-column decomposition.

### The 2-D Combined Row-and-column Decomposition Algorithm:

1. Choose either row-major or column-major and perform the decomposition as  $S_{N \times N} = \max\{B_{N \times 1} \oplus C_{1 \times N}, S'_{N \times N}\}$ . In this dissertation, only the row-major decomposition is presented. The column-major decomposition can be derived similarly. One row, say  $B_{N \times 1}$ , is selected which is the maximum over the sum of the row elements in  $S_{N \times N}$ . The  $C_{1 \times N}$  can be obtained by selecting the row-wise minimum of the matrix generated from subtracting each row of  $S_{N \times N}$  by  $B_{N \times 1}$ . For illustration, if selecting

$$B_{3 \times 1} = [5, 7, 8] \text{ in } S_{3 \times 3} = \begin{bmatrix} 5 & 7 & 8 \\ 3 & 2 & 9 \\ 6 & 3 & 5 \end{bmatrix}, \text{ then}$$

$$C_{1 \times 3} = \min_{\text{row-wise}} \left\{ \begin{bmatrix} 5 & 7 & 8 \\ 3 & 2 & 9 \\ 6 & 3 & 5 \end{bmatrix} - \begin{bmatrix} 5 & 7 & 8 \\ 5 & 7 & 8 \\ 5 & 7 & 8 \end{bmatrix} \right\} = \begin{bmatrix} 0 \\ -5 \\ -4 \end{bmatrix}.$$

Note that, it will cause errors if the row-wise minimum is not selected. For example, if the value is randomly selected from each row of the differences as shown below

$$C_{1 \times 3} = \underset{\text{row-wise}}{\text{random}} \left\{ \begin{bmatrix} 0 & 0 & 0 \\ -2 & -5 & 1 \\ 1 & -4 & -3 \end{bmatrix} \right\} = \begin{bmatrix} 0 \\ 1 \\ -3 \end{bmatrix}. \text{ The dilation of } B_{3 \times 1} \text{ and } C_{1 \times 3} \text{ will become}$$

$$B_{3 \times 1} \oplus C_{1 \times 3} = \begin{bmatrix} 5 & 7 & 8 \\ \underline{6} & \underline{8} & 9 \\ 2 & \underline{4} & 5 \end{bmatrix}, \text{ and it is impossible to reconstruct the original structuring}$$

element because those elements with underline will be selected after the maximum operation.

2. Generate  $S'_{N \times N}$  by comparing  $S_{N \times N}$  and  $B_{N \times 1} \oplus C_{1 \times N}$ . That is,  $S'_{(i,j)}$  is set to be “x” if the corresponding values in  $S_{(i,j)}$  and  $[B_{N \times 1} \oplus C_{1 \times N}]_{(i,j)}$  are the same;



otherwise,  $S'_{(i,j)} = S_{(i,j)}$ . Note that,  $(i, j)$  indicate the  $i$ -th row and  $j$ -th column. If all the elements in  $S'_{N \times N}$  are “ $\times$ ”, go to step 4.

3. Set  $S_{N \times N} = S'_{N \times N}$ , and go to step 1.
4. Decompose  $B_{N \times 1}$  and  $C_{1 \times N}$  into small structuring elements of sizes 2 and 3 using the aforementioned algorithm in Section 3.

### 3.5 Experimental Results

#### 3.5.1 Binary

**Example 3.1:** The structuring element shown in Figure 3.15 is not decomposable by Park and Chin’s [62] definition [example 3]. Using the proposed approach, it is not only find the same factors obtained by Hashimoto and Barrera [35] as shown in Figure 3.15(a), but also obtain another better decomposition as shown in Figure 3.15(b) which is suitable for a parallel pipelined architecture. In Figure 3.15(a), it reduces from the original 20 elementary dilations to 9 elementary dilations. In Figure 3.15(b), it requires only 8 elementary dilations. Note that the origin of the structuring element is always placed at the center of the window.

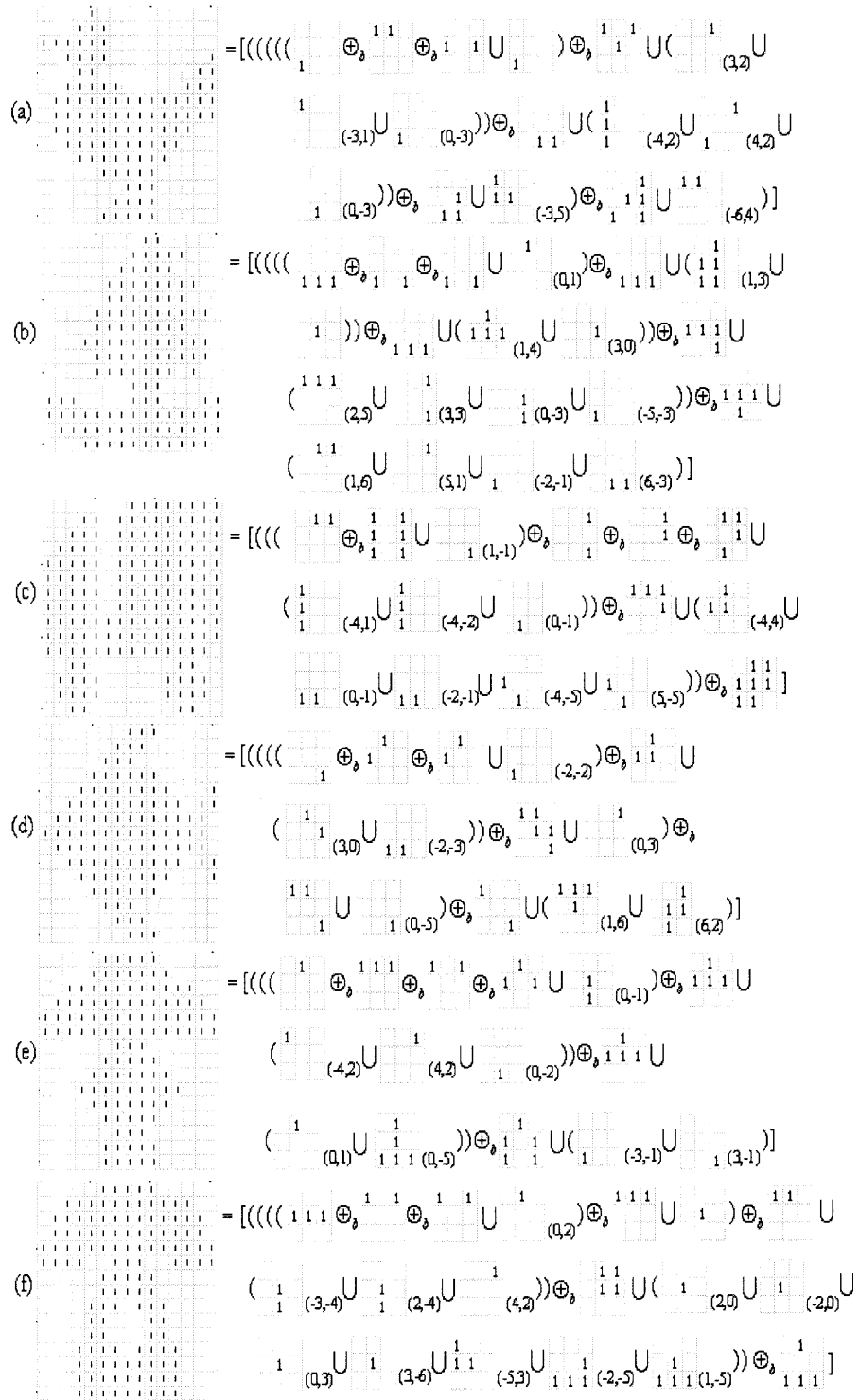
Note that in Figure 3.15(b), a structuring element is shifted by  $(-1, -2)$ . Figure 3.16(a) denotes the geometric relationship in Cartesian coordinates. Figure 3.16(b) is a  $3 \times 3$  structuring element with an extension to  $7 \times 7$  by a null shift. Figures 3.16(c) and (d) are the examples of translation by  $(2, 2)$  and  $(-2, 1)$ , respectively.



**Example 3.3:** Figure 3.18 shows an example of decomposing a large sized structuring element of size  $17 \times 17$ . In Anelli, Broggi, and Destri [2], it takes about six hours of CPU time when a  $16 \times 16$  structuring element is decomposed on the two processors HP 9000 with 128 megabytes of RAM. Using the proposed algorithm, it takes about 20 minutes of CPU time when a  $17 \times 17$  structuring element is decomposed on a single processor Pentium III/866MHz with 256 megabytes of RAM. In Figure 3.18, a  $17 \times 17$  sized structuring element with a total of 124 elementary dilations is decomposed into several  $3 \times 3$  structuring elements with a total of 34 elementary dilations and 8 logical unions.

$$\begin{aligned}
 &= [(((( \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \oplus_{(0,3)} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \oplus_{(0,4)} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \oplus_{(0,2)} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} U \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} ) \oplus_{(0,3)} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \\
 &\quad \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} U \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} ) \oplus_{(0,4)} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} U \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} ) \oplus_{(0,2)} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} U \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \\
 &\quad \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} U \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} ) \oplus_{(0,3)} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} U \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} ] \\
 &= [(((( \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \oplus_{(0,3)} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \oplus_{(0,4)} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \oplus_{(0,2)} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} U \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} ) \oplus_{(0,3)} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \\
 &\quad \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} U \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} U \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} ) \oplus_{(0,4)} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} U \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} ) \oplus_{(0,2)} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \oplus_{(0,6)} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \\
 &\quad \oplus_{(0,7)} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} U \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} U \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} ) \oplus_{(-7,-3)} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} U \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} ) \oplus_{(7,-3)} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} ]
 \end{aligned}$$

**Figure 3.18** The decomposition of a  $17 \times 17$  structuring element.



**Figure 3.19** The decomposition of arbitrarily shaped structuring elements.

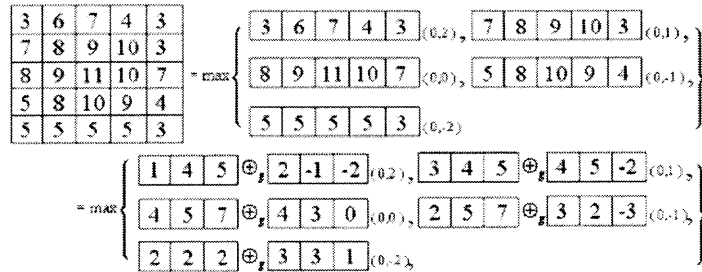
**Example 3.4:** Figure 3.19 shows that the proposed algorithm is efficient in decomposing arbitrarily shaped structuring element, such as duck, ship, car, fish, lamp and telephone. Note that Figure 3.19(f) illustrates the difficult case of a structuring element with a hole in the telephone. Table 3.1 lists the comparison of the original structuring element and its decomposition.

**Table 3.1** Comparison Between Traditional and Proposed Algorithm

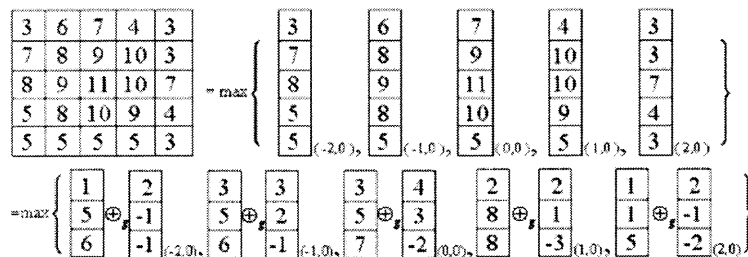
	Dark	Ship	Car	Fish	Lamp	Telephone
Original Elementary Dilation	101	124	167	111	114	126
Original Logical Union	0	0	0	0	0	0
New Elementary Dilation	31	45	45	31	33	39
New Logical Union	9	13	9	7	8	12

**3.5.2 Grayscale**

**Example 3.5:** Figures 3.20 and 3.21 show the examples of decomposing a 2-D structuring element based on row and column 1-D structuring element, respectively.

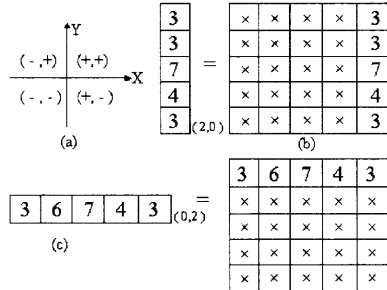


**Figure 3.20** The row decomposition of a 2-D structuring element of size 5 × 5.



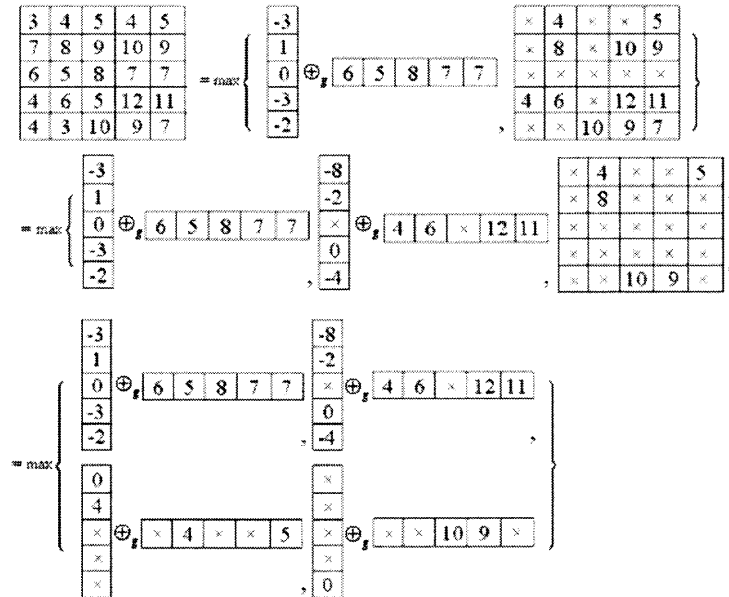
**Figure 3.21** The column decomposition of a 2-D structuring element of size 5 × 5.

Note that in Figures 3.20 and 3.21, the row or column structuring elements are shifted by  $(x, y)$ , where  $x$  and  $y$  are integers. Figure 3.22(a) denotes the geometric relationship in Cartesian coordinates. Figure 3.22(b) is the example of translation of a 1-D column structuring element by  $(2, 0)$ . Figure 3.22(c) is the example of translation of a 1-D row structuring element by  $(0, 2)$ .



**Figure 3.22** The representation of translation.

**Example 3.6:** Figure 3.23 shows the combined dilation and maximum operators for decomposing a 2-D structuring element.



**Figure 3.23** The combined dilation and maximum operators.

## CHAPTER 4

### SHORTEST PATH PLANNING

The shortest path planning problem is investigated in this chapter. The problem is given as that a car (object / robot) of any desired shape moves from a starting position to a destination in a finite space with arbitrarily shaped obstacles in it. In this dissertation, only the configuration space approach is considered by using the distance map that is made by the *Euclidean distance transformation* (EDT). The efficient algorithms to performing EDT are also developed in this dissertation. A new chain-code representation is presented to record the motion path when forward and backward movements are allowed. By placing the smooth turning-angle constraint, more realistic results can be obtained to the actual motion of cars. Meanwhile, by combining rotational mathematical morphology and distance transformation, the shortest collision-free path can be obtained. As soon as the distance map and the collision-free codes have been established off-line, the shortest paths of cars starting from any location toward the destination can be promptly obtained on-line. Experimental results show that the proposed algorithm works successfully in different conditions.

#### 4.1 Introduction

The recent advances in the fields of robotics and artificial intelligence have stimulated considerable interests in the robot motion planning and the shortest path finding problems [49]. The path planning is in general concerned with finding paths connecting different locations in an environment (e.g., a network, a graph, or a geometric space). Depending

on the specific applications, the desired paths often need to satisfy some constraints (e.g., obstacle-avoiding) and optimize certain criteria (e.g., variant distance metrics and cost functions). The problems of planning shortest paths arise in many disciplines, and in fact are one of the several most powerful tools for modeling combinatorial optimization problems.

Two approaches in shortest path planning are often used. One is via configuration space where the geometry of the robot is added to the obstacle in order to create a new free space [55]. The other approach uses the computational geometry to search the space directly without transforming the workspace to the configuration space [11, 47]. In this dissertation, only the configuration space approach is considered by using the distance map made by the Euclidean Distance Transformation.

Pei, Lai and Shih [63] proposed a MM based algorithm for *shortest path planning* (SPP). Although they achieve the two constraints such as collision-free and smooth turning-angle, the additional space and time-complexity in their algorithm appear to be not optimum. In order to reduce the time-complexity and the additional space, the *dynamically rotational mathematical morphology* (DRMM) is proposed in this chapter. In order to record the shortest path, a new chain-code representation is used to allow “16” different movements in 2D and “52” different movements in 3D. Furthermore, the backward movement is allowed and the constraint of the smooth turning-angle is placed to simulate the actual motion of cars.



This chapter is organized as follows. Section 4.2 presents the notations and definitions used in the shortest path planning problem. The algorithms for performing EDT are shown in Section 4.3. The EDT-based SPP approach and the car-like SPP problem are developed in Sections 4.4 and 4.5, respectively.

## 4.2 Notations and Definitions in the Shortest Path Planning

### 4.2.1 The Properties of a Distance Function

Let  $S$  be a set of pairs of integers. The function  $d$  mapping from  $S \times S$  to nonnegative integers is called a *distance function*, if it is

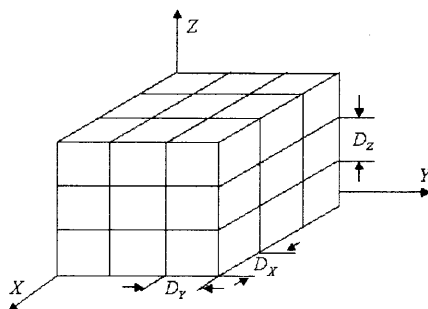
(a) *Positive Definite*: That is  $d(p, q) \geq 0$ , if and only if  $p=q$ , for all  $p, q \in S$ .

(b) *Symmetric*: That is  $d(p, q) = d(q, p)$ , for all  $p, q \in S$ .

(c) *Triangular*: That is  $d(p, r) \leq d(p, q) + d(q, r)$ , for all  $p, q, r \in S$ .

### 4.2.2 3-D Image Representation

A 3-D digitized domain can be represented as a 3-D matrix:  $T^3[x][y][z]$  (or  $T^3[x, y, z]$ ) of dimensions  $X \times Y \times Z$ , where  $x$ ,  $y$ , and  $z$  denote respectively the row, column and height coordinates, as shown in Figure 4.1. Each voxel has physical size  $D_x \times D_y \times D_z$  in physical units (e.g.  $mm$  or  $\mu m$ ). For a binary image, the values of voxels are composed of only two kinds (0 and 1). For a gray-scale image, the values of voxels are composed of 256 kinds (0, 1, 2, ..., 255). Note that the representation can be extended to images of an arbitrarily dimensional domain as shown in Figure 4.2 where an  $n$ -D digitized domain can be represented as an  $n$ -D matrix  $T^n[t_1][t_2] \cdots [t_n]$  of dimensions  $t_1 \times t_2 \times \cdots \times t_n$ .



**Figure 4.1** The 3-D image representation.

$$\begin{aligned}
 t_1 &= \{ -\infty \dots 0 \dots +\infty \} \\
 t_2 &= \{ -\infty \dots 0 \dots +\infty \} \\
 t_3 &= \{ -\infty \dots 0 \dots +\infty \} \\
 &\vdots \\
 t_n &= \{ -\infty \dots 0 \dots +\infty \}
 \end{aligned}$$

**Figure 4.2** The representation of the  $n$ -dimensional space.

### 4.2.3 Distance Transformation in 3-D Domain

Let two points be  $p = (p_1, p_2, p_3)$  and  $q = (q_1, q_2, q_3)$  in a 3-D digital image, where  $p_1, p_2, p_3, q_1, q_2$  and  $q_3$  are integers. Three frequently-used distance functions are defined as follows:

(a) *City-block distance*:  $d_{ci}(p, q) = |p_1 - q_1| + |p_2 - q_2| + |p_3 - q_3|$ .

(b) *Chessboard distance*:  $d_{ch}(p, q) = \max(|p_1 - q_1|, |p_2 - q_2|, |p_3 - q_3|)$ .

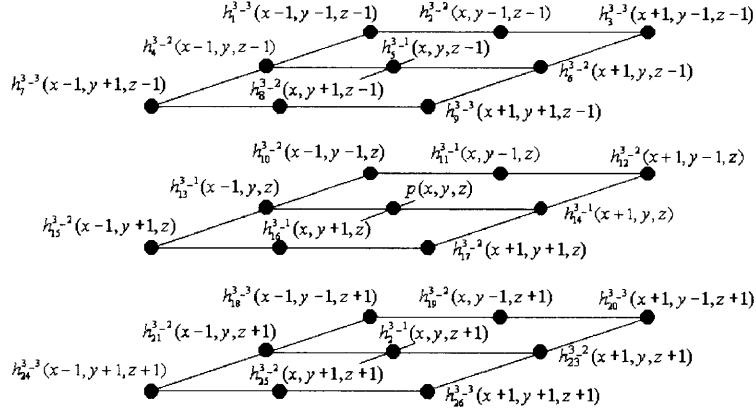
(c) *Euclidean distance*:  $d_e(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2}$ .

Note that the pixels with *city-block distance* 1 counting from  $p$  correspond to 6-neighbors of  $p$ , and with *chessboard distance* 1 correspond to 26-neighbors of  $p$ . These

$d_{ci}$  and  $d_{ch}$  are integer-valued; however,  $d_e$  is not. The *Squared Euclidean Distance* (SED) is defined as

$$d_s(p, q) = [d_e(p, q)]^2$$

The distance functions above can be similarly extended to the  $n$ -D domain.



**Figure 4.3** The 26 neighbors of pixel  $p$ .

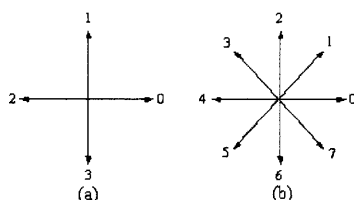
#### 4.2.4 3-D Neighborhood Used in EDT

In the 3-D space, let 26 neighbors of  $p$  be denoted by  $N^3(p) = \{h_1^3, h_2^3, \dots, h_{26}^3\}$  as illustrated in Figure 4.3. They can be categorized into three groups:  $N^{3-1}(p)$ ,  $N^{3-2}(p)$  and  $N^{3-3}(p)$ , in which the corresponding  $h_{index}^{3-d}$  adopts the notations  $d$  and  $index$  representing the number of moves between  $p$  and its neighbor and the sequential number in the neighborhood, respectively. Generally, in the  $n$ -D domain the number of neighbors of  $p$  is  $3^n - 1$  and they can be categorized into  $n$  groups.

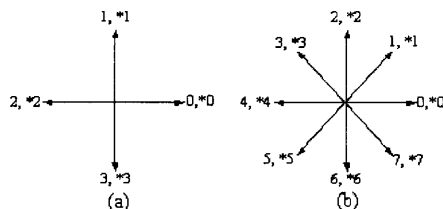
### 4.2.5 Forward and Backward Chain-Codes Representation

The chain-code representation [83, 84] is efficient in contour tracing. It is based on 4- or 8-connectivity of the segments as shown in Figure 4.4 and each direction is represented by a number  $i$ , where  $i = 0, 1, 2, 3$  indicating  $90i^\circ$ , or  $i = 0, 1, \dots, 7$  indicating  $45i^\circ$ , respectively.

Since the backward movement is allowed in car motion, the traditional chain-code representation cannot satisfy this requirement. Therefore, one additional set of chain codes prefixed by “\*” is added to denote “backward,” as shown in Figure 4.5. For example, “0” indicates the forward movement from the center to the east with the car oriented toward east, and “\*0” indicates the backward movement from the eastern neighbor to the center with the car oriented toward east.



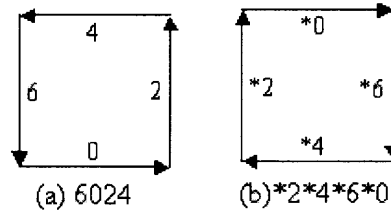
**Figure 4.4** The representation of 4- and 8-connectivity chain codes.



**Figure 4.5** The new chain-code representation.

In this chapter, the 8-connectivity chain codes is adopted. Figures 4.6(a) and (b) show the examples of the traditional (only considering forward movement) and the new chain codes, respectively. Note that, the starting point is located at the top-left corner of

the square and the tracing direction is counter-clockwise. For illustration purpose, 90° turns is allowed in the examples. Therefore, “6024” and “\*2\*4\*6\*0” can be obtained by using the traditional and the new chain codes, respectively.



**Figure 4.6** The examples of (a) the traditional (b) the new chain codes.

#### 4.2.6 Rotational Morphology

The path-planning problem is given as that a car of arbitrary shape moves from a starting position to a destination in a finite space with arbitrarily shaped obstacles in it. When the traditional mathematical morphology is adopted, the drawback is that the fixed directional movement of the structuring element (i.e., the car) is longer than the optimal path in real world applications [53]. By incorporating rotations into the motion of the structuring element, more realistic results can be obtained in solving the shortest path finding problem [63].

Rotational morphology can be considered in the digital image as that the full 360 degrees are equally divided into eight parts. Let  $A$  and  $B$  denote a binary image and a binary structuring element, respectively. The rotational dilation of  $A$  by  $B$  is defined as [63]

$$\begin{aligned} (A \tilde{\oplus} B)(y) &= [A \oplus B_7, A \oplus B_6, \dots, A \oplus B_1, A \oplus B_0](y) \\ &= [P_7, P_6, \dots, P_1, P_0](y). \end{aligned}$$

Similarly, the rotational erosion of A by B is defined as

$$(A \overset{\sim}{\ominus} B)(y) = [A \ominus B_7, A \ominus B_6, \dots, A \ominus B_1, A \ominus B_0](y).$$

Let  $\vec{P}$  denote an 8-bit codeword with a base 2. If  $P_i = 1$ , it indicates that when the SE is rotated by  $45i^\circ$ , it can be fit into the object region. All such  $i$ 's satisfying " $P_i = 1$ " can be used to represent available collision-free rotations at this location in the path finding, so that the shortest path can be achieved.

### 4.3 Euclidean Distance Transformation

*Distance Transformation* (DT) is used to convert a digital binary image that consists of object (foreground) and nonobject (background) pixels into another image where each object pixel has a value corresponding to the minimum distance defined by a distance function from the background. Among different kinds of DT, the *Euclidean distance transformation* (EDT) is often-used because of its rotational invariance property, but it involves the time-consuming calculations such as square-root, and the minimum over a set of floating-point numbers.

In [45, 82, 86], a *mathematical morphology* (MM) approach was proposed to realize EDT using gray-scale erosions with successive small distance *structuring elements* (SE) by decomposition. Furthermore, a squared Euclidean-distance structuring element was used to perform the *squared Euclidean distance transformation* (SEDTE). Shih and Wu [85] decomposed the *squared Euclidean-distance structuring element* (SEDSE) into successive dilations of a set of  $3 \times 3$  structuring components. Hence, SEDTE is equivalent to the successive erosions of the result at the preceding stage by each

structuring component. EDT can be finally obtained simply by a square-root operation over the image. In particular, image analysis tasks can directly take the result of SEDT as input.

The approaches to achieve distance transformation do not adapt directly the definition of the minimum distance from an object pixel to all background border pixels, since their computations are extremely time-consuming. Previous research in [9, 10, 21, 34, 45, 71, 72, 73, 81, 82, 85, 95, 99, 108] represent a sampled set of successful efforts in improving the speed efficiency. In general, the algorithms of DT can be categorized into two classes: one is the *iterative* method which is efficient in a cellular array computer since all the pixels at each iteration can be processed in parallel, and the other is *sequential* (or *recursive*) method which is suited for a conventional computer by avoiding iterations with the efficiency to be independent of object size. Using the general computers that most people have access to, sequential algorithms are often more efficient than iterative ones.

Although many techniques have been presented for obtaining EDT, most of them are either inefficient or complex to implement. Furthermore, they require extra cost such as special structure or storages for information recording. Cuisenaire and Macq [20] proposed EDT by propagation using multiple neighborhoods, but they need a bucket sorting before calculating EDT. Datta and Soundaralakshmi [22] proposed a constant-time algorithm for achieving EDT, but their algorithm is based on a special hardware structure, the reconfigurable mesh. Eggers [24] proposed an algorithm by avoiding unnecessary calculations, but some data must be recorded in the lists. In this

dissertation, the two-scan based algorithm by a deriving approach is developed for achieving correct EDT in a constant time.

### 4.3.1 The Basic Morphological Approach

Mathematical morphology represents image objects as sets in a Euclidean space. Note that, the set is the primary notion and the function is viewed as a particular case of the set; e.g., an N-dimensional multi-valued function is viewed as a set in (N+1)-dimensional space. From this viewpoint then, any function- or set-processing system is viewed as a set mapping (transformation) from one class of sets into another class of sets.

The *Euclidean distance structuring element* (EDSE), denoted by  $k(x, y) = -\sqrt{x^2 + y^2}$ , is adopted for achieving EDT and can be decomposed using the concave decomposition strategy [82]. In other words, the structuring element can be decomposed into the maximum selection of several structuring components and thus needs more operations than the convex decomposition in which the structuring element can be decomposed into successive dilations of these segmented linearly-sloped structuring components. The *squared Euclidean distance structuring element* (SEDSE), denoted by  $k^2(x, y) = -(x^2 + y^2)$ , is expressed as in [45, 85]. Following is an example of SEDSE with size of 9×9:

$$k_{(9 \times 9)}^2 = \begin{bmatrix} -32 & -25 & -20 & -17 & -16 & -17 & -20 & -25 & -32 \\ -25 & -18 & -13 & -10 & -9 & -10 & -13 & -18 & -25 \\ -20 & -13 & -8 & -5 & -4 & -5 & -8 & -13 & -20 \\ -17 & -10 & -5 & -2 & -1 & -2 & -5 & -10 & -17 \\ -16 & -9 & -4 & -1 & 0 & -1 & -4 & -9 & -16 \\ -17 & -10 & -5 & -2 & -1 & -2 & -5 & -10 & -17 \\ -20 & -13 & -8 & -5 & -4 & -5 & -8 & -13 & -20 \\ -25 & -18 & -13 & -10 & -9 & -10 & -13 & -18 & -25 \\ -32 & -25 & -20 & -17 & -16 & -17 & -20 & -25 & -32 \end{bmatrix}$$



Since the SEDSE changes uniformly both in the vertical and horizontal directions, the aforementioned structuring element can be decomposed into

$$k_{(9 \times 9)}^2 = \begin{bmatrix} -2 & -1 & -2 \\ -1 & 0 & -1 \\ -2 & -1 & -2 \end{bmatrix} \oplus \begin{bmatrix} -6 & -3 & -6 \\ -3 & 0 & -3 \\ -6 & -3 & -6 \end{bmatrix} \oplus \begin{bmatrix} -10 & -5 & -10 \\ -5 & 0 & -5 \\ -10 & -5 & -10 \end{bmatrix} \oplus \begin{bmatrix} -14 & -7 & -14 \\ -7 & 0 & -7 \\ -14 & -7 & -14 \end{bmatrix}$$

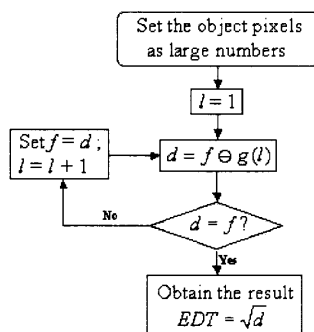
Let the structuring component be denoted by  $g(l)$  and expressed as

$$g(l) = \begin{bmatrix} -(4l-2) & -(2l-1) & -(4l-2) \\ -(2l-1) & 0 & -(2l-1) \\ -(4l-2) & -(2l-1) & -(4l-2) \end{bmatrix} \quad (4.1)$$

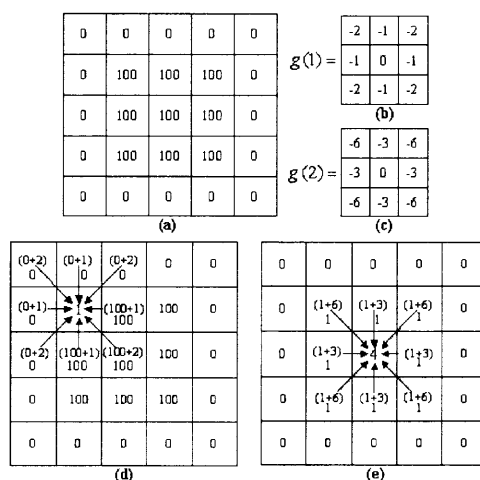
where  $l$  indicates the iteration numbers. Hence, the EDT is the iterative erosions by a set of small structuring components and then a square-root operation. Given a binary image  $f$  which is represented by “ $+\infty$ ” (or a large number) and “0.” Following is the *Iterative Erosion Algorithm* (IEA) and its flowchart is shown in Figure 4.7.

#### **Iterative Erosion Algorithm (IEA):**

1. Set the values of object to a large number, and values of background to 0.
2. Initialize  $l = 1$ .
3.  $d = f \ominus g(l)$
4. If  $d \neq f$ , let  $f = d$  and  $l++$ , and repeat the steps 3 and 4 until  $d = f$ .
5. Take a square root of  $d$  (the distance image). That is  $EDT = \sqrt{d}$ .



**Figure 4.7** The flowchart of the Iterative Erosion Algorithm.



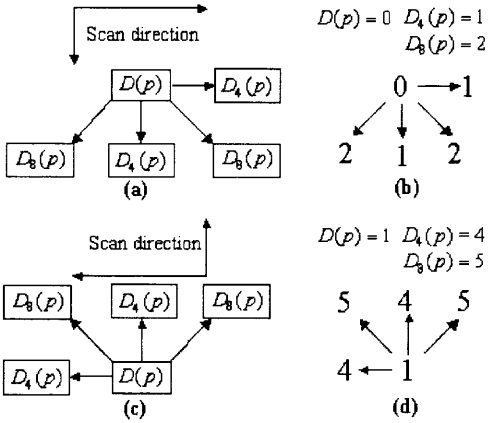
**Figure 4.8** Examples of the acquiring procedure.

### 4.3.2 The Two-Scan Based Algorithm

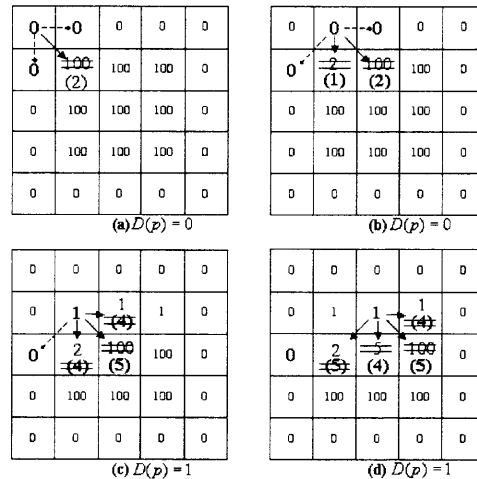
Generally, the approaches for achieving EDT are in the way of acquiring the minimum Euclidean distance by checking neighborhood. That is, the “acquiring” is to acquire the distance information from its neighbors to obtain its current Euclidean distance. Figure 4.8 illustrates the acquiring procedure. Figure 4.8(a) is the original image in which the object pixels are changed into “100”. Figures 4.8(b) and (c) are the structuring components,  $g(l)$ , when  $l=1$  and 2. Figure 4.8(d) shows the condition of achieving EDT by using  $g(1)$  in the top-left object pixel. Note that, the calculations are in the direction of left-to-right top-to-bottom. Figure 4.8(e) shows the condition when calculating EDT

by using  $g(2)$  in the center of the object. From Figures 4.8(d) and (e), the EDT of each pixel is achieved based on its neighborhood.

In order to achieve EDT efficiently without extra cost, the deriving approach is adopted. The “deriving” is to propagate its current distance information to its neighbors for updating their distance values. That is, each pixel decides the *Euclidean distance* of its children (i.e., pixels in its neighborhood). Figure 4.9 shows the examples of the deriving approach. Figures 4.9(a) and (c) indicate the relation between a pixel and its children. In Figure 4.9(a), the scanning path is in the direction of left-to-right, top-to-bottom. On the contrary, the scanning path in Figure 4.9(c) is in the direction of right-to-left, bottom-to-top. Let  $D(p)$  be the value of SED of a pixel  $p$ . Let  $D_4(p)$  and  $D_8(p)$  be the SED values of 4- and 8-neighbors of  $p$  respectively and can be calculated by  $D(p)$ . The method for calculating  $D_4(p)$  and  $D_8(p)$  will be described later. Note that, the concerned children of  $p$  are only 4 pixels in either scanning direction. Figures 4.9(b) and (d) are the examples of the ED value of children of  $p$  when  $D(p) = 0$  and 1, respectively.



**Figure 4.9** Examples of the deriving approach.



**Figure 4.10** The examples when calculating an image based on the deriving approach.

Figure 4.10 shows the examples when calculating EDT in Figure 4.8(a) based on the deriving approach. Figure 4.10(a) shows the deriving scheme when dealing with the top-left pixel, say (1,1). Only the object pixels are changed if there exists a smaller ED value. Therefore, the value in (2,2) is changed from “100” to “2.” Continually dealing with the next pixel (2,1) as shown in Figure 4.10(b), the value in (2,2) is changed once more since there exists a smaller value “1.” Similarly, the value in (3,2) is changed from “100” to “2.” In Figure 4.10(c), the value in (3,2) is not changed because the original value “1” in (3,2) is smaller than the deriving value “4” from pixel (2,2). In the same way, the value in (2,3) is not changed. In Figure 4.10(d), there are only two pixels changed in (3,3) and (4,3) because of the smaller deriving value from pixel (3,2). Note that, the deriving values are shown in parenthesis.

### 4.3.3 The Fundamental Lemmas

In this subsection, three lemmas are presented.

**Lemma 4.1:** Pixels in the *squared Euclidean Distance Transform* (SED<sub>T</sub>) can be divided into a summation of two squared integers.

**Proof:** Let two points be  $p = (x, y)$  and  $q = (u, v)$  in a digital image, where  $x, y, u,$  and  $v$  are integers. According to the definition of SED<sub>T</sub>

$$d_s(p, q) = d_e^2(p, q) = (x - u)^2 + (y - v)^2,$$

it is obvious that  $d_s(p, q)$  is summation of  $(x - u)^2$  and  $(y - v)^2$ . Therefore, Lemma 4.1 is proved by the definition of SED<sub>T</sub>.

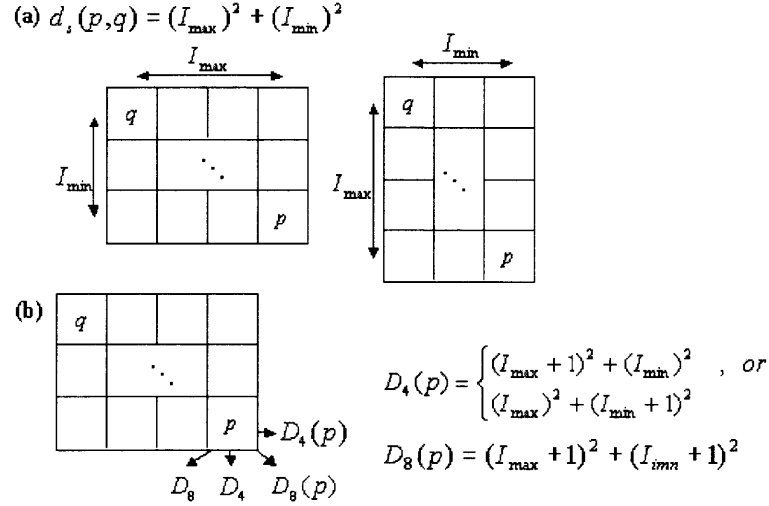
**Lemma 4.2:** If  $d_s(p, q)$  is divided into  $I_{\max}$  and  $I_{\min}$  where

$d_s(p, q) = (I_{\max})^2 + (I_{\min})^2$  and  $I_{\max} \geq I_{\min}$ ,  $D_4(p)$  and  $D_8(p)$  can be obtained by the following equations:

$$D_4(p) = \begin{cases} (I_{\max} + 1)^2 + (I_{\min})^2 \\ (I_{\max})^2 + (I_{\min} + 1)^2 \end{cases}, \text{ or} \quad (4.2)$$

$$D_8(p) = (I_{\max} + 1)^2 + (I_{\min} + 1)^2 \quad (4.3)$$

**Proof:** If  $d_s(p, q) = (I_{\max})^2 + (I_{\min})^2$ , it means the vertical and horizontal pixels between  $p$  and  $q$  are  $I_{\max}$  and  $I_{\min}$  (or  $I_{\min}$  and  $I_{\max}$ ) as shown in Figure 4.11(a). In Figure 4.11(b), the SED<sub>T</sub> of the 4-adjacent pixels of  $p$  can be obtained by increasing one pixel in the vertical or horizontal direction. Similarly, the SED<sub>T</sub> of the 8-adjacent pixels of  $p$  can be obtained by increasing one pixel in the vertical direction and one pixel in the horizontal direction. Therefore, Equations (4.2) and (4.3) are proved.



**Figure 4.11** The  $D_4(p)$  and  $D_8(p)$  of pixel  $p$ .

**Lemma 4.3:** If  $d_s(p, q)$  can be divided into more than one summation of two squared integers, i.e.,

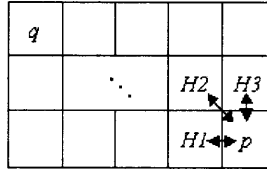
$$d_s(p, q) = \begin{cases} (I_{\max\_1})^2 + (I_{\min\_1})^2 \\ (I_{\max\_2})^2 + (I_{\min\_2})^2 \\ \vdots \\ (I_{\max\_n})^2 + (I_{\min\_n})^2 \end{cases}, \quad (4.4)$$

the neighborhood of pixel  $p$  is checked in order to obtain the exact one.

**Proof:** In Figure 4.12, the value of SEDT of pixel  $p$  can be obtained by its parents ( $H_1, H_2$  and  $H_3$ ) according to Lemma 4.2. Although  $d_s(p, q)$  can be divided into more than one summation of two squared integers, the exact solution can be obtained by checking each pair of squared integers.

Suppose there are  $n$  possibilities when dividing  $d_s(p, q)$ . If  $p$  is derived from  $H_1$ ,  $d_s(H_1, q)$  should be equal to  $(I_{\max} - 1)^2 + (I_{\min})^2$  or  $(I_{\max})^2 + (I_{\min} - 1)^2$ . Similarly, If  $p$  is derived from  $H_3$ ,  $d_s(H_3, q)$  should be equal to  $(I_{\max} - 1)^2 + (I_{\min})^2$

or  $(I_{\max})^2 + (I_{\min} - 1)^2$ . If  $p$  is derived from  $H_2$ ,  $d_s(H_2, q)$  should be equal to  $(I_{\max} - 1)^2 + (I_{\min} - 1)^2$ .



**Figure 4.12** The relation between  $p$  and its parents.

#### 4.3.4 $TS_1$ - Two-Scan Based EDT Algorithm for General Images

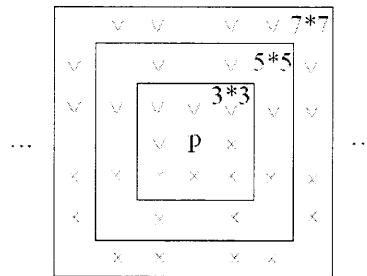
If only eight neighbors are used for EDT, there exist some errors from 24.47 degrees as described in [21]. By reflecting with respect to  $x$ -,  $y$ -,  $45^\circ$ -, and  $135^\circ$ -axes, 7 more error cases appear. On the other hand, the acquiring approach needs additional space to record all the neighborhood information. In order to achieve the correct EDT in the two-scan based approach for general images (i.e., without obstacles), the following two skills are adopted. First, the deriving approach is used instead of the acquiring approach. Second, the neighborhood window is dynamically adjusted based on the Euclidean distance value.

Cuisenaire and Macq [20] pointed out the relationship between the value of SEDT and the neighborhood window. That is, the larger SEDT value a pixel has, the bigger neighborhood window it uses. Based on [20], the cases indicating the relationship of the SEDT and its corresponding neighborhood windows are shown in Table 4.1. For example, if the value of SED of a pixel is greater than or equal to 116, a  $5 \times 5$  neighborhood window is adopted. Figure 4.13 shows the neighborhood window of  $p$ . The signs “ $\times$ ” and “ $\vee$ ” indicate the concerned neighbors in the forward and backward raster scan, respectively. Note that because of no error occurrence, the neighbors in the

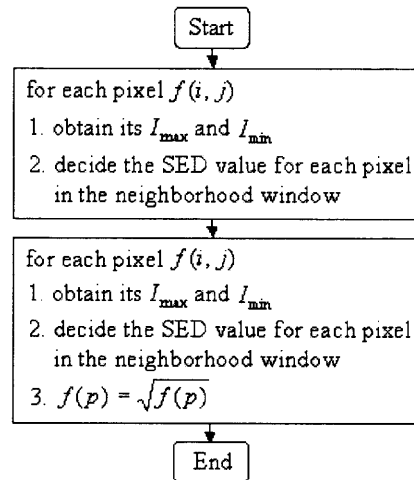
directions 0, 45, 90 and 135 degrees are not concerned when the sizes of the neighborhood window are greater than  $3 \times 3$ .

**Table 4.1** Relationship Between SEDT and Neighborhood Window

Neighborhood	$I_{\max}$	$I_{\min}$	SEDT
$5 \times 5$	10	4	116
$7 \times 7$	22	6	520
$9 \times 9$	44	9	2017
$11 \times 11$	67	11	4610
$\vdots$	$\vdots$	$\vdots$	$\vdots$



**Figure 4.13** The example of the neighborhood window of pixel  $p$ .



**Figure 4.14** The flowchart of the  $TS_1$  algorithm.



Let the object pixels be a large number (larger than the square of a half of the image size) and the background pixels be 0. The two-scan based EDT algorithm,  $TS_1$ , is described below and its flowchart is shown in Figure 4.14. Let  $N_F(i, j)$  and  $N_B(i, j)$  are the concerned neighbors of pixel  $(i, j)$  in forward and backward raster-scans, respectively.

**$TS_1$  Algorithm:**

A. (Forward) Raster-Scan:

1. for ( $i = 0 ; i \leq length ; i ++$ )
2.     for ( $j = 0 ; j \leq length ; j ++$ )
3.     { Divide  $f(i, j)$  into  $I_{\max}$  and  $I_{\min}$ , where  $f(i, j) = (I_{\max})^2 + (I_{\min})^2$ ;
4.         if there exists more than one division, check the parents of  $f(i, j)$  for exacting  $I_{\max}$  and  $I_{\min}$
5.         Decide the size of neighborhood window based on the value of  $f(i, j)$
6.     for all  $q \in N_F(i, j)$ , decide the SED value based on  $f(i, j)$  }

B. Backward (Reverse) Raster-Scan:

7. for ( $i = length ; i \geq 0 ; i --$ )
8.     for ( $j = length ; j \geq 0 ; j --$ )
9.     { Divide  $f(i, j)$  into  $I_{\max}$  and  $I_{\min}$ , where  $f(i, j) = (I_{\max})^2 + (I_{\min})^2$ ;
10.         if there exists more than one division, check the parents of  $f(i, j)$  for exacting  $I_{\max}$  and  $I_{\min}$
11.         Decide the size of neighborhood window based on  $f(i, j)$



In order to speed up the two-scan algorithm, a lookup table is utilized to replace the calculation of splitting a SED value into the summation of two squared integers. Table 4.2 shows a portion of the lookup table. Let  $N_4$  and  $N_8$  be the horizontal/vertical and diagonal neighbors of  $p$ , respectively. For example, if the SED value of  $p$  is 25, based on the lookup table, it is clearly that 25 can be divided into two possible combinations, i.e.,  $9+16$  and  $0+25$ . Then the procedure of checking the SED values of its  $N_4$  and  $N_8$  neighbors is made to determine the correct combination. Note that, “×” denotes for some cases,  $N_4$  and  $N_8$  are not necessary checked.

#### 4.3.5 $TS_\infty$ - Two-Scan Based EDT Algorithm for Images with Obstacles

Generally, the images used in the shortest path planning contain some obstacles. Assume that an object of arbitrary shape moves from a starting point to an ending point in a finite region with arbitrarily shaped obstacles in it. To relate this model to mathematical morphology, the finite region consists of a free space set with values equal to ones, starting and ending points with values equal to zeros, an obstacle set with values equal to negative ones, and the moving object is modeled as a structuring element. Thus, the shortest path finding problem is equivalent to applying a morphological erosion to the free space, followed by a distance transformation on the region with the grown obstacles excluded, and then tracing back the resultant distance map from the target point to its neighbors with the minimum distance until the starting point is reached.

As mentioned earlier, many algorithms cannot obtain the correct EDT when obstacles appear in the image. Figure 4.15 shows the example for the image with obstacles that some algorithms cannot obtain the exact EDT. For simplicity, the SEDT is

adopted to illustrate the example. Figures 4.15(a) and (b) are the images with and without obstacles, respectively. Both Egger’s and Shih and Mitchell’s algorithms cannot obtain the correct EDT. Figures 4.15(c2), (d2) and (e2) are the rules based on Egger’s algorithm [24] from page 109. Based on the rules, Figures 4.15(c1), (d1) and (e1) are obtained as the results of achieving SEDT of Figure 4.15(a) in the first, second and third iterations. Figures 4.15(c4), (d4) and (e4) are the morphological structuring components. Similarly, Figures 4.15(c6), (d6) and (e6) can be obtained. Figures 4.15(c3), (d3), (e3), (c5), (d5) and (e5) are obtained using morphological erosions. By observing Figures 4.15(e1) and (e3), it is clearly that the value in the square box, say “15”, is obtained by the summation of its neighbor with value 5 and the distance 10. Both of them are not correct SED. The correct SED in this position should be 13, as shown in Figure 4.15(e5). Figure 4.15(f) shows the correct result obtained by the proposed two-scan based algorithm.

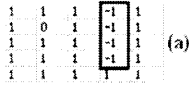
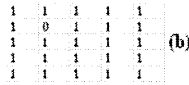

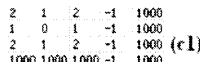
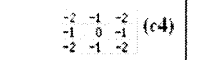
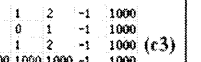
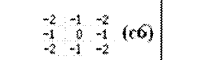
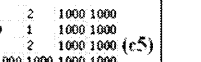

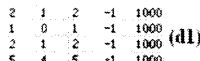
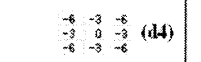
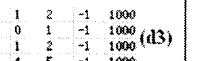
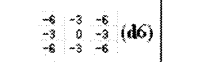
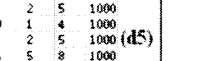

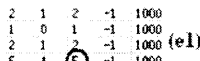
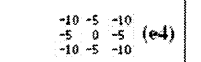
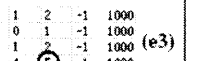
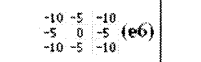
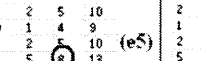

Original Image	 (a)	 (b)		
Iteration	Eggers's algorithm	Shih and Mitchell's	Shih and Mitchell's	Two-scan based
(I) iter =1	 (c2)  (c1) 2 1 2 -1 1000 1 0 1 -1 1000 2 1 2 -1 1000 1000 1000 1000 -1 1000 1000 1000 1000 1000 1000	 (c4)  (c3) 2 1 2 -1 1000 1 0 1 -1 1000 2 1 2 -1 1000 1000 1000 1000 -1 1000 1000 1000 1000 1000 1000	 (c6)  (c5) 2 1 2 1000 1000 1 0 1 1000 1000 2 1 2 1000 1000 1000 1000 1000 1000 1000 1000 1000 1000 1000 1000	
(II) iter =2	 (d2)  (d1) 2 1 2 -1 1000 1 0 1 -1 1000 2 1 2 -1 1000 5 4 5 -1 1000 1000 1000 1000 1000 1000	 (d4)  (d3) 2 1 2 -1 1000 1 0 1 -1 1000 2 1 2 -1 1000 5 4 5 -1 1000 1000 1000 1000 1000 1000	 (d6)  (d5) 2 1 2 5 1000 1 0 1 4 1000 2 1 2 5 1000 1000 5 4 5 8 1000 1000 1000 1000 1000 1000	
(III) iter =3	 (e2)  (e1) 2 1 2 -1 1000 1 0 1 -1 1000 2 1 2 -1 1000 5 4 5 -1 1000 10 9 10 15 1000	 (e4)  (e3) 2 1 2 -1 1000 1 0 1 -1 1000 2 1 2 -1 1000 5 4 5 -1 1000 10 9 10 15 1000	 (e6)  (e5) 2 1 2 5 10 1 0 1 4 9 2 1 2 5 10 10 5 4 5 8 13 10 9 10 13 18	 (f)

Figure 4.15 The example that some algorithms cannot obtain the exact EDT.

The two-scan based EDT algorithm,  $TS_{\infty}$ , for dealing with images with obstacles is described below. Let  $P$  be the SEDT image recording the previous processing image.

**$TS_{\infty}$  Algorithm:**

1. Set  $P = f$ .
2. Obtain SEDT of  $f$  by the  $TS_1$  algorithm except step 13.
3. Repeat steps 1 to 3 until  $P = f$ .
4. Take a square root of  $P$  (the distance image). That is  $EDT = \sqrt{P}$ .

#### 4.3.6 Computational Complexity

For an image with size  $n \times n$ , the complexity for the simple approximate algorithm developed by Danielsson [21] is  $O(n^2)$ . The exact EDT algorithm proposed by Eggers [24] is highly dependent on the image content and can vary between  $O(n^2)$  and  $O(n^3)$ . The complexity of the proposed Improved Iteration Erosion is  $O(dn^2)$ , where  $d$  is the maximum number of chessboard distances. For the two-scan based EDT algorithm, the complexity is  $O(n^2)$ . Furthermore, it can successfully achieve correct EDT for the images with obstacles.

Three images shown in Figure 4.16 are experimented in a PC with Pentium III/866MHz and 256 megabytes RAM. Figure 4.16(a) shows a disk of variable diameter created for test 4.1. The size of Figure 4.16(a) is  $1024 \times 1024$ . Figure 4.16(b) shows a grayscale image with size of  $327 \times 300$ . Figure 4.16(c) shows the binary image after thresholding for test 4.2. Figure 4.16(d) shows the straight line across the image with size of  $1024 \times 1024$  for test 4.3.

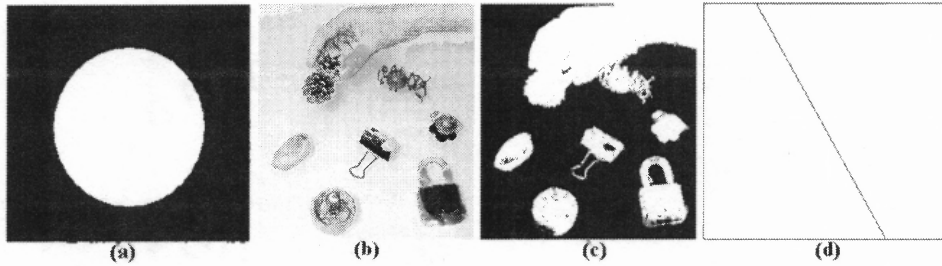


Figure 4.16 The testing images.

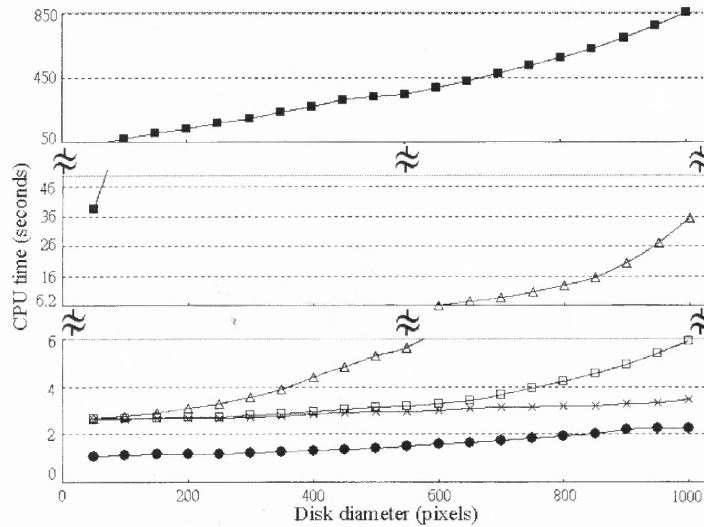


Figure 4.17 Test 4.1: diameters vary from 50 to 1000 pixels.

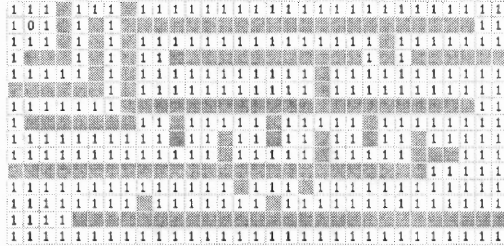
There are six different algorithms experimented in test 4.1 as shown in Figure 4.17. Followings are the symbols used and their corresponding algorithm: (“●”, Danielsson), (“x”,  $TS_1$  with a fixed  $3 \times 3$  window), (“□”, Two-Scan  $TS_1$ ), (“△”, Eggers), and (“■”, Traditional morphology erosion / Iterative Erosion Algorithm). It is obvious that the traditional morphology erosion is time consuming. Although Eggers’s algorithm successfully improved the performance, the execution time is high when the diameter of the disk becomes large.  $TS_1$  reduces the execution time dramatically.  $TS_1$  with a fixed  $3 \times 3$  neighborhood window is also implemented as an approximate

algorithm. Although the execution time is larger than Danielsson’s algorithm, but the correct rate is much higher than Danielsson’s algorithm. The experimental results for tests 4.2 and 4.3 are shown in Table 4.3.

**Table 4.3** Experimental Results for Tests 4.2 and 4.3

Algorithm	Test 4.2		Test 4.3	
	Time (sec.)	Errors (pixels)	Time (sec.)	Errors (pixels)
Danielsson	0.45	3	2.1	0
$TS_1$ (Approximate)	0.52	1	2.7	0
$TS_1$	0.54	0	5.8	0
Egger	0.58	0	32.7	0
IEA	12.06	0	842.3	0

For a complex image with obstacles as shown in Figure 4.18, 8 times of  $TS_1$  is required for obtaining the exact EDT.



**Figure 4.18** The image with obstacles.

### 4.3.7 The Extension to the 3-D Space

Let  $o$  denote a background pixel. The method for calculating  $d_s(N^{3-1}(p), o)$ ,  $d_s(N^{3-2}(p), o)$  and  $d_s(N^{3-3}(p), o)$  is described below. Note that, the respective children of pixel  $p$  in either forward or backward scanning contain 13 pixels and are illustrated in Figures 4.19 and 4.20 For simplicity,  $d_s(p, q)$  is substituted by  $d_s(p)$  and  $d_s(h_{index}^{3-d}, q)$  is replaced by  $d_s(h_{index}^3)$  if  $q$  is a background pixel.

$d_s(p) = x^2 + y^2 + z^2$	$d_s(h_{20}^3) = (x+1)^2 + (y+1)^2 + (z+1)^2$
$d_s(h_{14}^3) = (x+1)^2 + y^2 + z^2$	$d_s(h_{21}^3) = (x+1)^2 + y^2 + (z+1)^2$
$d_s(h_{15}^3) = (x+1)^2 + (y+1)^2 + z^2$	$d_s(h_{22}^3) = x^2 + y^2 + (z+1)^2$
$d_s(h_{16}^3) = x^2 + (y+1)^2 + z^2$	$d_s(h_{23}^3) = (x+1)^2 + y^2 + (z+1)^2$
$d_s(h_{17}^3) = (x+1)^2 + (y+1)^2 + z^2$	$d_s(h_{24}^3) = (x+1)^2 + (y+1)^2 + (z+1)^2$
$d_s(h_{18}^3) = (x+1)^2 + (y+1)^2 + (z+1)^2$	$d_s(h_{25}^3) = x^2 + (y+1)^2 + (z+1)^2$
$d_s(h_{19}^3) = x^2 + (y+1)^2 + (z+1)^2$	$d_s(h_{26}^3) = (x+1)^2 + (y+1)^2 + (z+1)^2$

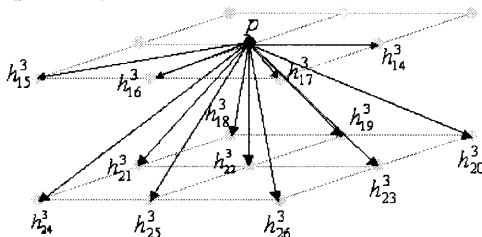


Figure 4.19 The 13 children in the forward scanning.

$d_s(p) = x^2 + y^2 + z^2$	$d_s(h_7^3) = (x+1)^2 + (y+1)^2 + (z+1)^2$
$d_s(h_4^3) = (x+1)^2 + (y+1)^2 + (z+1)^2$	$d_s(h_8^3) = x^2 + (y+1)^2 + (z+1)^2$
$d_s(h_5^3) = x^2 + (y+1)^2 + (z+1)^2$	$d_s(h_9^3) = (x+1)^2 + y^2 + z^2$
$d_s(h_6^3) = (x+1)^2 + (y+1)^2 + (z+1)^2$	$d_s(h_{10}^3) = (x+1)^2 + (y+1)^2 + z^2$
$d_s(h_1^3) = (x+1)^2 + y^2 + (z+1)^2$	$d_s(h_{11}^3) = x^2 + (y+1)^2 + z^2$
$d_s(h_2^3) = x^2 + y^2 + (z+1)^2$	$d_s(h_{12}^3) = (x+1)^2 + (y+1)^2 + z^2$
$d_s(h_3^3) = (x+1)^2 + y^2 + (z+1)^2$	$d_s(h_{13}^3) = (x+1)^2 + y^2 + z^2$

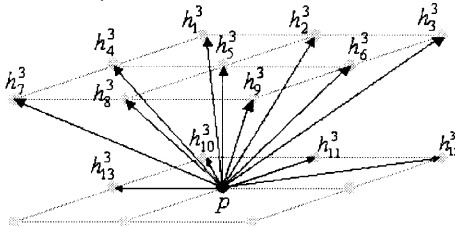


Figure 4.20 The 13 children in the backward scanning.

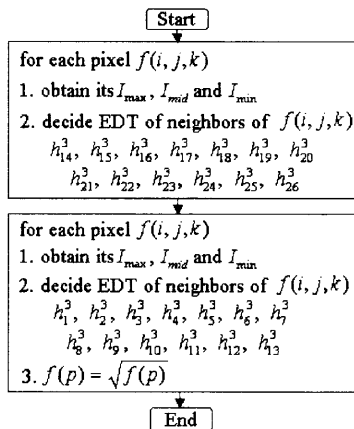


Figure 4.21 The flowchart of the two-scan based algorithm.



Let the object pixels be a large number (larger than the square of a half of the image size) and the background pixels be “0”. The algorithm is described below and its flowchart is shown in Figure 4.21.

### Two-Scan Based Algorithm for 3-D EDT:

A. (Forward) left-to-right, top-to-bottom, near-to-far scan:

1. for ( $i = 0 ; i \leq length ; i ++$ )
2. for ( $j = 0 ; j \leq length ; j ++$ )
3. for ( $k = 0 ; k \leq length ; k ++$ )
4. { Decompose  $f(i, j, k)$  into  $I_{\max}, I_{mid}$  and  $I_{\min}$ , where  

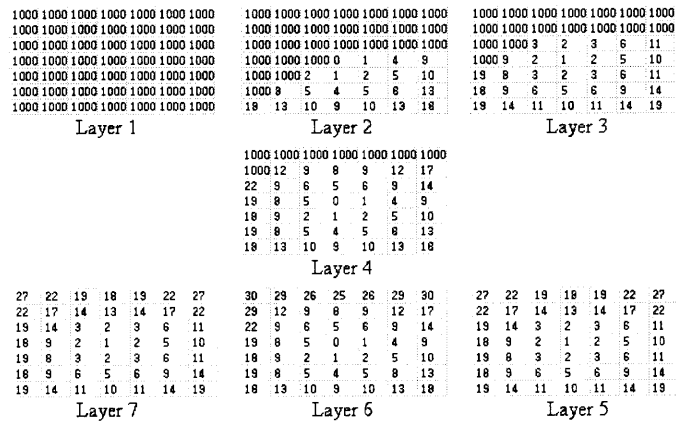
$$f(i, j) = (I_{\max})^2 + (I_{mid})^2 + (I_{\min})^2;$$
5. If there exists more than one decomposition, check the parents of  
 $f(i, j, k)$  to choose the exact  $I_{\max}, I_{mid}$  and  $I_{\min}$  ;
6. For the following three types of neighborhoods, perform:
7.  $f(N^{3-1}(p)) = \min(f(N^{3-1}(p)), (I_{\max} + 1)^2 + (I_{mid})^2 + (I_{\min})^2);$
8.  $f(N^{3-2}(p)) = \min(f(N^{3-2}(p)), (I_{\max} + 1)^2 + (I_{mid} + 1)^2 + (I_{\min})^2);$
9.  $f(N^{3-3}(p)) = \min(f(N^{3-3}(p)), (I_{\max} + 1)^2 + (I_{mid} + 1)^2 + (I_{\min} + 1)^2); }$

2. Backward (Reverse) far-to-near, bottom-to-top, right-to-left scan:

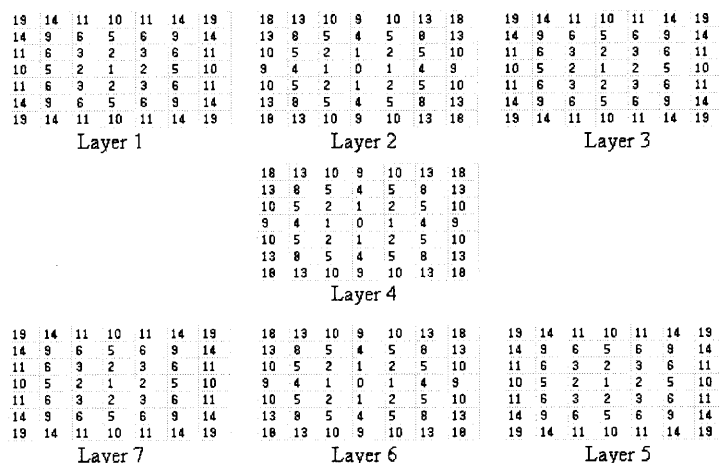
10. for ( $i = length ; i \geq 0 ; i --$ )
11. for ( $j = length ; j \geq 0 ; j --$ )
12. for ( $k = length ; k \geq 0 ; k --$ )

13. { Decompose  $f(i, j, k)$  into  $I_{\max}, I_{\text{mid}}$  and  $I_{\min}$ ,  
where  $f(i, j) = (I_{\max})^2 + (I_{\text{mid}})^2 + (I_{\min})^2$ ;
14. If there exists more than one decomposition, check the parents of  
 $f(i, j, k)$  to choose the exact  $I_{\max}, I_{\text{mid}}$  and  $I_{\min}$ ;
15. For the following three types of neighborhoods, perform:
16.  $f(N^{3-1}(p)) = \min(f(N^{3-1}(p)), (I_{\max} + 1)^2 + (I_{\text{mid}})^2 + (I_{\min})^2)$ ;
17.  $f(N^{3-2}(p)) = \min(f(N^{3-2}(p)), (I_{\max} + 1)^2 + (I_{\text{mid}} + 1)^2 + (I_{\min})^2)$ ;
18.  $f(N^{3-3}(p)) = \min(f(N^{3-3}(p)), (I_{\max} + 1)^2 + (I_{\text{mid}} + 1)^2 + (I_{\min} + 1)^2)$ ;
19.  $E(f(i, j, k)) = \sqrt{f(i, j, k)}$ ;

Figures 4.22 and 4.23 show the forward and backward scanning results of the two-scan based algorithm on an image of  $7 \times 7 \times 7$ . Note that there are only three background pixels in the image. They are located at the center of layers 2, 4 and 6.



**Figure 4.22** The forward scanning of the two-scan based algorithm.



**Figure 4.23** The backward scanning of the two-scan based algorithm.

#### 4.4 The EDT-Based SPP Approach

Since only one background (destination) in the distance map, it is always possible to find a neighbor with a smaller SED value from a pixel if the pixel is not the destination. Therefore, the shortest path from an arbitrary source to the destination can be achieved on the distance map. Assume that an arbitrarily-shaped object moves from a starting point to a destination point in a finite space with arbitrarily-shaped obstacles in it. The relation of this model and the mathematical morphology is obtained using the following strategy:

- (1) The finite space consists of free regions as foreground and obstacles as “forbidden” areas with value “-1,” and the moving object is modeled as the structuring element;
- (2) Set the destination point to be “0;”
- (3) Apply a morphological erosion to the free regions, followed by a distance transformation on the region with the grown obstacles excluded, and then trace the distance map from any starting point to its neighbor of the minimum distance until the destination point is reached.

#### 4.4.1 Improvement on the Rotational Mathematical Morphology

In [63], although the rotational morphology is useful for achieving collision-free, the additional space and time-complexity in their algorithm are not optimal. For example, the additional record is an 8-bit codeword with a base 2 and the processing of mathematical morphology is 8 times for each pixel in the 2-D domain. Moreover, the cost increases dramatically accompanied with the increasing dimensions. For example in the 3-D space, the additional record is a 26-bit codeword and the morphological processing is 26 times for each pixel.

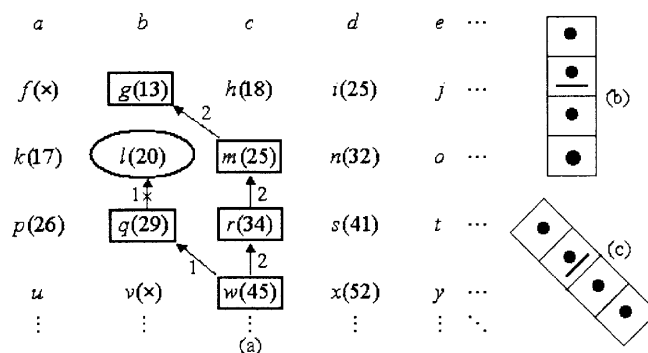
In order to reduce the time-complexity and the additional space, the dynamically rotational mathematical morphology (DRMM) is proposed in this chapter. It is not necessary to calculate the morphological erosion for each pixel in the original image until the pixel is selected in the SPP.

Figure 4.24 shows an example of deciding the next step by the DRMM. Figures 4.24(a) and (b) are the original image and the structure element, respectively. The underline indicates the center of the structuring element. Figure 4.24(c) is the structuring element with a rotation of  $45^\circ$ . Note that the counter-clockwise rotation is counted to be positive angles. Let pixel  $w$  be the starting point. When the next neighboring pixel is decided in SPP, the following two criteria are used:

- (1) The neighbor with the minimum SED is selected.
- (2) The neighbor remaining as an object pixel after a rotational erosion (i.e. without collision) is selected.

When deciding the next step of pixel  $w$ , pixel  $q$  is selected because of its minimum SED and the satisfaction of morphological erosion. Note that the adopted structuring element is rotated by  $45^\circ$  because pixel  $q$  is in the direction of  $45^\circ$  of pixel

$w$ . Assume pixels  $v$  and  $f$  are obstacles. Continually, when deciding the next step of pixel  $q$ , pixel  $l$  is not selected because the condition of applying the erosion on pixel  $l$  by Figure 4.24(b) is not satisfied (i.e., collision with the obstacle pixel  $v$ ). Therefore, it is necessary to go back to  $w$  and reconsider the next step of  $w$ . The pixel  $r$  is selected since it is satisfied by the two criteria. Furthermore,  $m$  and  $g$  are selected.



**Figure 4.24** The example of DRMM.

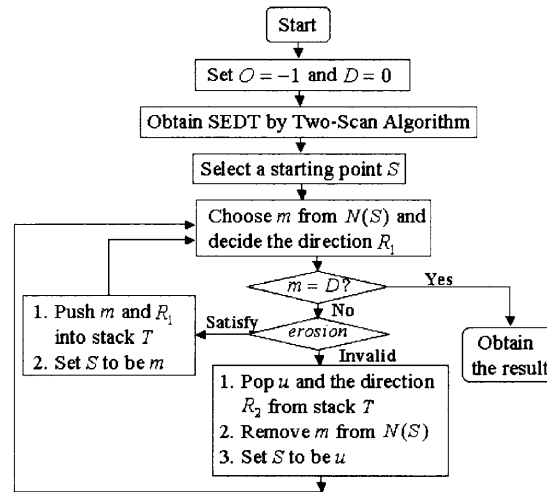
#### 4.4.2 The Novel Algorithm for Shortest Path Planning

Let  $W$ ,  $O$ ,  $E$ ,  $T$ ,  $D$  and  $S$  denote working domain, obstacle set, SE set, stack, destination and starting points, respectively. The proposed algorithm is described below and its flowchart is shown in Figure 4.25.

##### Algorithm:

1. Set the values of the obstacle pixels  $O$  and the destination pixel  $D$  to be “-1” and “0,” respectively.
2. Obtain the SEDT by the two-scan based algorithm described in Section 4.3.5.
3. Select a starting point  $S$  on the SEDT image.
4. Choose pixel  $m$  with the minimum SED from the neighborhood of  $S$ ,  $N(S)$ , and decide the direction  $R_1$  which is a vector from  $S$  to  $m$  to be recorded as a chaincode.

5. Apply a morphological erosion on pixel  $m$  by the structuring element which is rotated based on the direction obtained from the previous step. If the result is satisfied, both  $m$  and  $R_1$  are pushed into the stack  $T$  and set  $S$  to be  $m$ . Otherwise, pop the preceding pixel  $u$  from the stack, remove pixel  $m$  from  $N(S)$ , and set  $S$  to be  $u$ .
6. Obtain the shortest path by repeating the steps 4 and 5 until the destination point  $D$  is reached.



**Figure 4.25** The flowchart for achieving SPP based on DRMM.

#### 4.4.3 Examples of the Proposed Algorithm

Figure 4.26 illustrates the experimental result when achieving SPP based on the DRMM in which the pixels enclosed by an ellipse with values “0” and “2377” indicate the destination and starting points, respectively. The dark pixels with value “-1” are obstacles and the pixels enclosed by a rectangle represent the shortest path.

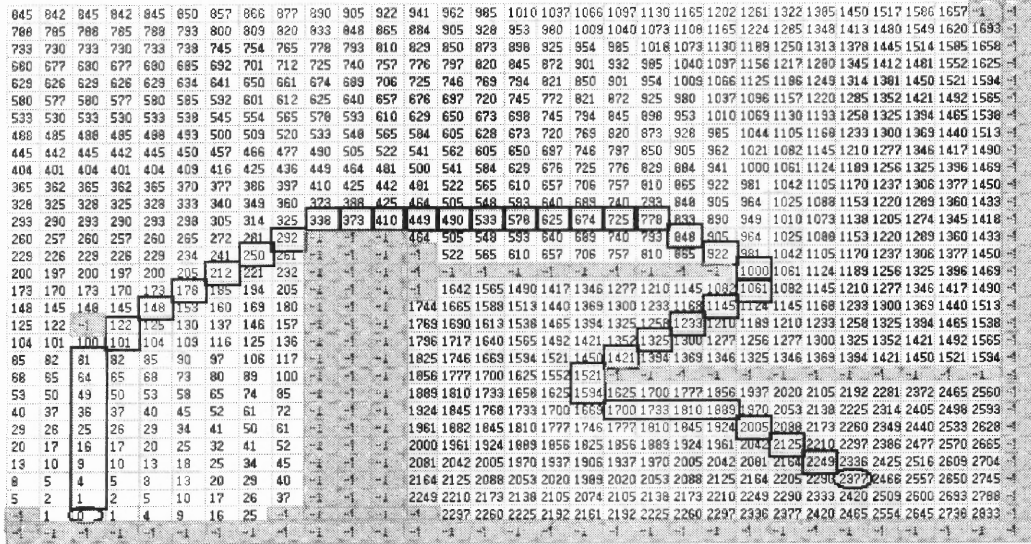


Figure 4.26 The experimental result for achieving SPP based on DRMM.

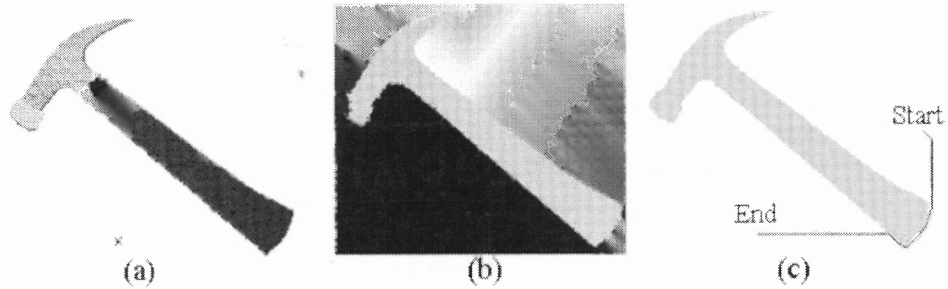


Figure 4.27 The SPP example of an actual image.

Figure 4.27 shows the SPP example on an actual image with size  $300 \times 350$ . Figures 4.27(a) is the original image. Figure 4.27(b) is obtained as the distance map by achieving the two-scan based EDT. Note that Figure 4.27(b) is different to Figure 4.27(b) because of the difference in the object and background pixels. In Figure 4.27(a), the pixels within the hammer are set to be  $-1$  for representing the obstacle pixels, the white pixels are set to be  $1$  for representing the object pixels, and only one background pixel is set to be  $0$  as shown in the center of sign “ $\times$ ”. The SPP is obtained as shown in Figure 4.27(c).

## 4.5 Car-Like Object SPP Problem

In order to simulate the actual motion of cars, the backward movement is allowed and the constraint of the smooth turning-angle is placed.

### 4.5.1 The SPP Algorithm for a Car-Like Object

Let  $W$ ,  $T$ ,  $C$ ,  $R$ ,  $G$ ,  $D$ , and  $S$  denote working domain, stack, car orientation, moving direction, rotation angle, the destination, and the starting point, respectively. The overall of the SPP algorithm for a car-like object is presented as follows and its flowchart is shown in Figure 4.28.

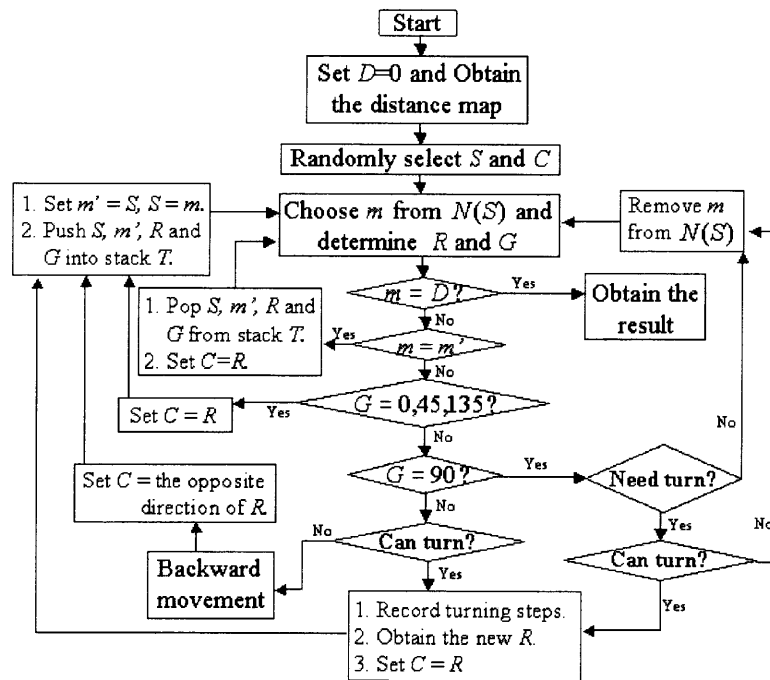
#### The SPP Algorithm for a Car-Like Object:

1. Set  $D$  to be 0.
2. Obtain the distance map by the recursive propagation algorithm
3. Randomly select a starting point  $S$  and a beginning car orientation  $C$  on the distance map.
4. Choose a pixel  $m$  from  $N(S)$  that has the minimum city-block distance value and the collision-free code "1," and record the corresponding  $R$  and  $G$  values.
5. If  $m = D$ , the destination point is reached.
6. If  $m$  is a previously selected pixel  $m'$ , pop a record in which a new set of data  $S$ ,  $m'$ ,  $R$ , and  $G$  from the stack  $T$  is obtained. Set  $C = R$  and go to step 4.
7. If  $G = 0^\circ$ ,  $45^\circ$  or  $135^\circ$ , it indicates that the car can move to the location. Therefore, set  $C = R$ ,  $m' = S$ ,  $S = m$ , and push  $S$ ,  $m'$ ,  $R$ , and  $G$  into the stack  $T$ , and then go to step 4.
8. If  $G = 90^\circ$ , it is necessary to check whether the car is required to turn or not. If the car orientation is different from the next moving direction, it is required to turn.



However, if the turn causes collision, the pixel  $m$  is removed from  $N(S)$  and go to step 4. Otherwise, continue in step 9.

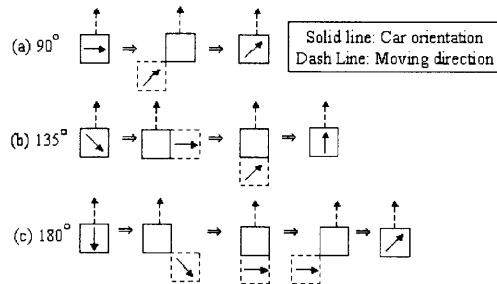
9. If the car can be turned at the location  $m$ , record the turning steps and obtain the new car orientation  $R$ . Set  $C = R$ ,  $m' = S$ ,  $S = m$ , and push  $S$ ,  $m'$ ,  $R$  and  $G$  into the stack  $T$ . Otherwise, the car should take a backward movement at the location  $m$ . Note that, if the car moves backward, its orientation is equal to the opposite direction of  $R$ . Go to step 4 for continuing to seek the next pixel until the destination point is reached.



**Figure 4.28** The flowchart of the shortest path planning.

The turning cases in step 9 can be categorized into three classes:  $90^\circ$ ,  $135^\circ$  and  $180^\circ$  depending on the inner angle between the car orientation and the moving direction. Figure 4.29(a) shows an example of a backward movement followed by a forward movement in order to turn  $45^\circ$ . In other words, two movements are required to turn  $45^\circ$  at the same location. Similarly, it takes three and four steps for turning the car if the

angles between the car orientation and the moving direction are respectively  $135^\circ$  and  $180^\circ$ , as shown in Figures 4.29(b) and (c).



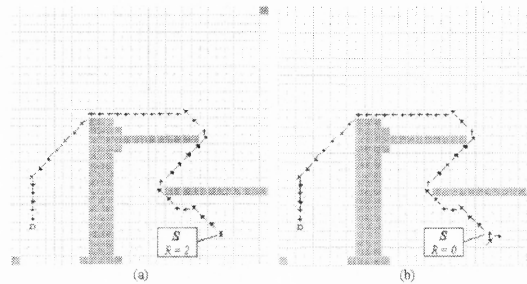
**Figure 4.29** The turning cases for different degrees.

### 4.5.2 Experimental Results

A few examples based on autonomous vehicles moving in a factory space of size  $30 \times 30$  are experimented. In Figure 4.28, the starting point is located at the position (25,27). Figures 4.30(a) and (b) show the examples when the beginning car orientation is toward north and east, respectively. In Figure 4.30(a), since the angle between the car orientation and the moving direction is  $45^\circ$ , it is not necessary to turn the car orientation. Therefore, the chain-code representation is “3334433211111233444444444444445555555666666.” In Figure 4.30(b), since the angle between the car orientation and the moving direction is  $135^\circ$ , it takes three additional steps to rotate the car orientation. Therefore, the chain-code representation is “0\*123334433211111233444444444444 5555555666666.”

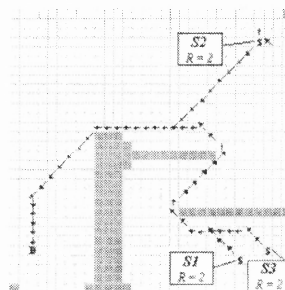
The experimental results of the shortest path planning algorithm are made on a computer with Pentium III/866MHz and 256 megabytes RAM under Window 2000 environment. The programs are written in Delphi 5. In Figure 4.30(a), the execution time for establishing the city-block distance map and collision-free codes is 0.56 seconds for

the size of  $30 \times 30$ . Once the required data are established, the time for obtaining the shortest path is 0.12 seconds.



**Figure 4.30** The example of the shortest path planning in the different car orientation.

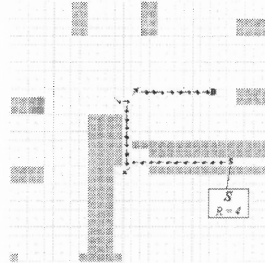
As soon as the distance map and the collision-free code have been established off-line in the configuration space, the shortest paths of cars starting from any location toward the destination can be promptly obtained on-line. In Figure 4.31, the corresponding chain codes for the three starting points are “333443321111123344444 4444445555555666666” for  $S_1$ , “2\*345555555544444444 5555555666666” for  $S_2$ , and “33444444332111112334444444444444555555 666666” for  $S_3$ .



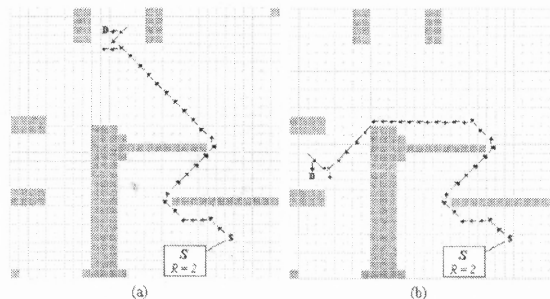
**Figure 4.31** The shortest path planning with different starting points.

The allowance for the backward motion can sometimes find a feasible path by using the complicated geometric methods [52, 67]. When the forward movement is solely used in finding a path in a narrow space and no solution can be found, it is a good choice

to try the backward movement. Figure 4.32 shows the car that cannot pass through the narrow space because of the collision with obstacles. By adding the backward movement, the path is obtained which is “4444444444445 \*6\*6\*6\*6\*6\*6\*6\*6\*701000000000.”



**Figure 4.32** The shortest path planning with backward movement.



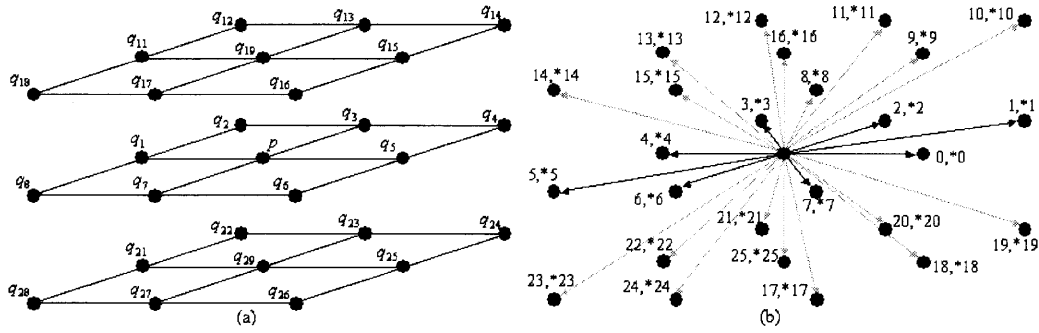
**Figure 4.33** Shortest path planning for parallel parking.

The shortest path planning algorithm can also be applied to automated parallel parking, as shown in Figure 4.33. The goal aims at placing the vehicle from the source (or starting) location to a designated (or destination) parking space. First, the shortest path planning algorithm is utilized to obtain a path from the source to an intermediate location, which is parallel to the parking space approximately and in the middle of upper and lower boundaries. From the intermediate location, the vehicle will move forward being still parallel to the parking space, until it reaches its upper boundary. Next, the vehicle will turn outward  $45^\circ$  and move backward to enter the parking space as long as it does not hit the boundary as applied by the aforementioned erosion rule. Finally, the vehicle will

adjust itself to the parking space by turning  $45^\circ$  inward and moving forward closer to its upper boundary. It is unnecessary to assign the direction of its driving and the direction of parking. The algorithm for automated parallel parking assumes that the vehicle can be parked in two directions depending on the vehicle direction in which it is oriented at the intermediate location. The chain codes in Figures 4.33(a) and (b) are obtained as “3344433211111 2333333333344\*5\*5 44” and “3344433211111233444444444444555 556\*7\*766,” respectively.

### 4.5.3 The Extension to the 3-D Space

To extend the proposed algorithm to three dimensions (3D), the number of neighbors for a pixel  $p$  is expanded from “8” in 2D to “26” in 3D, as shown in Figure 2.34(a). The distances between  $p$  and its neighbors are  $d(p, q_1) = d(p, q_3) = d(p, q_5) = d(p, q_7) = d(p, q_{19}) = d(p, q_{29}) = 1$ ,  $d(p, q_2) = d(p, q_4) = d(p, q_6) = d(p, q_8) = d(p, q_{11}) = d(p, q_{13}) = d(p, q_{15}) = d(p, q_{17}) = d(p, q_{21}) = d(p, q_{23}) = d(p, q_{25}) = d(p, q_{27}) = 2$ , and  $d(p, q_{12}) = d(p, q_{14}) = d(p, q_{16}) = d(p, q_{18}) = d(p, q_{22}) = d(p, q_{24}) = d(p, q_{26}) = d(p, q_{28}) = 3$ . Meanwhile, the number of chain codes required for the representation is enlarged from “16” to “52”, as shown in Figure 4.34(b). In order to denote each 3-D chain code in just one digit, the 26 lower-case alphabets are used, as shown in Table 4.4.

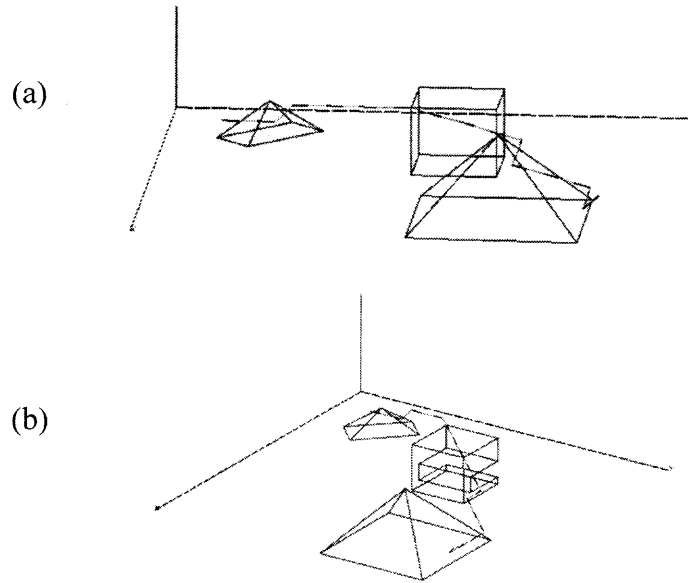


**Figure 4.34** The 26 neighbors of a pixel  $p$  and its 3-D chain-code representation.

Figure 4.35 illustrates the algorithm in the 3-D space. There are two obstacles, a cube and an octahedron, existed in the space, and an airplane model is used. Figure 4.35(a) shows the result of the proposed algorithm applying on the 3-D space with convex obstacles, and Figure 4.35(b) shows the result when there is a concave obstacle. The starting and destination points are located at  $(3,3,1)$  and  $(27,27,3)$ , respectively.

**Table 4.4** 3-D Chain-Code Representation by Using 26 Alphabets

Original	New	Original	New	Original	New	Original	New
0	a	13	n	*0	*a	*13	*n
1	b	14	o	*1	*b	*14	*o
2	c	15	p	*2	*c	*15	*p
3	d	16	q	*3	*d	*16	*q
4	e	17	r	*4	*e	*17	*r
5	f	18	s	*5	*f	*18	*s
6	g	19	t	*6	*g	*19	*t
7	h	20	u	*7	*h	*20	*u
8	i	21	v	*8	*i	*21	*v
9	j	22	w	*9	*j	*22	*w
10	k	23	x	*10	*k	*23	*x
11	l	24	y	*11	*l	*24	*y
12	m	25	z	*12	*m	*25	*z



**Figure 4.35** The example of the 3-D shortest path planning.

#### 4.5.4 The Rule of Distance Functions for Shortest Path Planning

If a distance function satisfies the following rule, it can be adopted for achieving SPP.

**Rule:** Let  $N^{n-1}(p)$ ,  $N^{n-2}(p)$ ,  $\dots$ , and  $N^{n-n}(p)$  be the  $n$  categories of neighbors of  $p$  in an  $n$ -D space based on a defined distance function. If the values of  $d(N^{n-1}(p), p)$ ,  $d(N^{n-2}(p), p)$ ,  $d(N^{n-3}(p), p)$ ,  $\dots$ , and  $d(N^{n-n}(p), p)$  are all different among them and in an increasing order, the distance function can be used for achieving SPP.

Based on the rule, it is obviously that Euclidean, squared Euclidean, and city-block distance functions can be used for achieving SPP; however the chessboard distance function cannot. In overall, the Euclidean distance is the one can preserve the true metric measure.

## CHAPTER 5

### SUMMARY AND FUTURE RESEARCH

This dissertation is aimed at developing efficient algorithms for multimedia security, image processing and robot vision. In this chapter, the summation and contributions of this dissertation will be discussed.

#### 5.1 The Contributions of This Dissertation

The contributions of this dissertation can be divided into two major parts: the theoretical and the application researches.

##### 5.1.1 Theoretical Research

1. *The deriving two-scan approach for EDT.* The two-scan based algorithm by a deriving approach is developed for correctly and efficiently achieving EDT in a constant time. Besides, many algorithms cannot correctly achieve EDT to the cases containing obstacles except the proposed two-scan based algorithm. Moreover, it does not require the additional cost for pre-processing and relative-coordinates recording tasks.
2. *The combinational image watermarking.* A novel conception using the combinational spatial and frequency domains approach is presented. Users will determine the strategy of splitting the watermark based on their preferences and data importance. Usually, the important part of data will be embedded into the frequency domain for robustness purpose. The proposed combinational image watermarking possesses the following advantage. More watermark can be inserted into the host image, i.e., spatial



and frequency domains, so that the capacity is increased.

3. *GA-based Watermarking*. The GA-based watermarking is developed in this dissertation to provide a novel approach and resolve some problems such as rounding error problem. The GA-based watermarking can be adapted to the medical image for increasing the compression rate. Furthermore, it can be considered as a fundamental platform for other fragile watermarking techniques.
4. *Adjusted-Purpose Watermarking*. There are many purpose-oriented watermarking techniques. For example, robust watermarks are used to withstand malicious attacks, fragile watermarks are applied for ownership authentication, and semi-fragile watermarks are designed for selective authentication that detects illegitimate distortion while ignoring applications of legitimate distortion. Further examples are the embedding strategies, some in spatial or frequency domain, and some in the combination of both. To integrate all of these, an adjusted-purpose digital watermarking technique is presented that uses two parameters, Varying Sized Transform Window and Quantity Factor, to choose the desired one particular watermarking. Therefore, when users design their own watermarking approaches, they can adopt the GA-based watermarking as a platform to make a suitable adjustment.
5. *Robust High-Capacity Watermarking*. A robust high-capacity watermarking is developed based on the block-based chaotic map and the intersection-based pixels collection. Since the block-based chaotic map can partially break the local spatial similarity of an image, the significant coefficients for embedding watermarks will increase dramatically. Meanwhile, the container and the reference register generated

by the intersection-based pixels collection perform robustly to resist attacks.

6. *GA-based Steganography*. In this dissertation, a new concept of developing a robust steganographic system is developed by artificially counterfeiting statistic features instead of the traditional strategy by avoiding the change of statistic features. The GA-based methodology is applied by adjusting gray values of a cover-image while creating the desired statistic features to generate the stego-images that can break the inspection of steganalytic systems.

### **5.1.2 Application Research**

1. *Shortest path planning problem*. In this dissertation, the property of rotation is incorporated into the traditional mathematical morphology, allow backward movement, and place the smooth turning-angle constraint to give more realistic results for the motion of mobile robots.
2. *Forward and backward chain-code representation*. A new chain-code representation is developed to record the motion path. It allows 16 different movements in 2D and 52 different movements in 3D. Furthermore, the backward movement is allowed and the constraint of the smooth turning-angle is placed to simulate the actual motion of cars
3. *GA-based algorithm for decomposing arbitrary binary morphological structuring elements*. In order to decompose arbitrarily shaped binary structuring elements efficiently, the 13 prime factors from (Park and Chin, 1994) is adopted to generate the initial population and recursively apply genetic algorithms in size reduction.
4. *The algorithm for decomposing arbitrary grayscale morphological structuring*

*elements*. By adoption of the union operator, it will guarantee the decomposability for arbitrary structuring elements. The column or row decomposition and the combined dilation and maximum operators are useful and efficient. The proposed grayscale structuring element decomposition algorithms are not required to perform extra pre-processes, such as deciding what type of a structuring element it is. Moreover, there is no need to check numerous rules for decomposing arbitrary 1-D or 2-D structuring elements.

5. *Robust watermarking and compression for medical images*. In order to achieve optimal compression as well as satisfactory visualization of medical images, the *ROI* is compressed by the lossless compression, and the rest by the lossy compression. In this dissertation, a robust technique is developed by embedding the watermark of signature information or textual data around the *ROI* of a medical image based on genetic algorithms.

## 5.2 Future Research

Following is the proposed future research:

1. Several algorithms have been successfully developed on area of image watermarking and steganography. However, there exist some fundamental contradictions in this area. For example, it is difficult to obtain a watermarking satisfies both criteria of high capacity and high robustness. Therefore, to find a new mechanism to break the contradictions attracts me a lot. On the other hand, the proposed algorithms are only focused on the static images. The extension of currently proposed methods to the video watermarking is also an interesting research topic.

2. The GA-based methodology is utilized in the steganographic system. However, the cost of time issue of the GA prohibits the potential to the commercial application. How to efficiently generate the stego-images which can pass the inspection of the steganalytic system is an exciting challenge to me.
3. The traditional morphological structuring element decomposition is analyzed in my previous works. How to extend the knowledge to different kinds of mathematical morphology such as fuzzy morphology, sweep morphology, and soft morphology is also important. Therefore, the extension of the proposed decomposition work to those morphological techniques is a challenging job.

## REFERENCES

1. Acharya, U. R., Anand, D., Bhat, P. S., and Niranjana, U. C., "Compact storage of medical image with patient information," *IEEE Trans. on Information Technology in Biomedicine*, vol. 5, no. 4, pp. 320-323, 2001.
2. Anelli, G., Broggi, A., and Destri, G., "Decomposition of arbitrarily shaped binary morphological structuring elements using genetic algorithms," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 2, pp. 217-224, Feb. 1998.
3. Avcibas, I., Memon, N., and Sankur, B., "Steganalysis using image quality metrics," *IEEE Trans. on Image Processing*, vol. 12, no. 2, pp. 221-229, 2003.
4. Avcibas, I. and Sankur, B., "Statistical analysis of image quality measures," *Journal Electron. Imag.*, vol. 11, pp. 206-223, 2002.
5. Barni, M., Bartolini, F., Rosa, A. D., and Piva, A., "Capacity of the watermark Channel: How Many Bits Can be Hidden Within a Digital Image?," *Proc. SPIE*, vol. 3657, pp. 437-448, San Jose, CA, Jan. 1999.
6. Bas, P., Chassery, J.-M., and Macq, B., "Image watermarking: an evolution to content based approaches," *Pattern Recognition*, vol. 35, pp. 545-561, 2002.
7. Berghel, H. and O'Gorman, L., "Protecting Ownership Rights through Digital Watermarking," *IEEE Computer Mag.*, pp. 101-103, July 1996.
8. Boneh, D., DeMillo, R., and Lipton, R., "On the importance of eliminating errors in cryptographic computations," *Journal of Cryptology*, vol. 14, no. 2, pp. 101-119, 2001.
9. Borgefors, G., "Distance transformations in arbitrary dimensions," *Computer Vision, Graphics, and Image Processing*, vol. 27, pp. 321-345, 1984.
10. Borgefors, G., "Distance transformations in digital images," *Computer Vision, Graphics, and Image Processing*, vol. 34, pp. 344-371, 1986.
11. Brooks, R. A., "Solving the find path problem by good representation for free space," *IEEE Trans. on Systems, Man, Cybernet.*, vol. 13, no. 3, pp. 190-197, 1983.
12. Brown, A., *S-Tools for Windows, Shareware 1994*,  
<ftp://idea.sec.dsi.unimi.it/pub/security/crypt/code/s-tools4.zip>.
13. Bruyndonckx, O., Quisquater, J.-J., and Macq, B., "Spatial method for copyright labeling of digital images," *Proc. Of IEEE Workshop on Nonlinear Signal and Image processing*, Neos Marmaras, Greece, pp. 456-459, 20-22 June 1995.

14. Camps, O. I., Kanungo, T., and Haralick, R. M., "Gray-scale structuring element decomposition," *IEEE Trans. on Image Processing*, vol. 5, no. 1, pp. 111-120, January 1996.
15. Caronni, G., "Assuring ownership rights for digital images," *Proc. of Reliable IT Systems, VIS'95*, Viewveg Publishing Company, Germany, 1995.
16. Celik, M. Sharma, U., Saber, G., E., and Tekalp, A. M., "Hierarchical watermarking for secure image authentication with localization," *IEEE Trans. on Image Processing*, vol. 11, no. 6, pp. 585-595, June 2002.
17. Chu, R., You, X., Kong, X., and Ba, X., "A DCT-based image steganographic method resisting statistical attacks," *International Conference on Acoustics, Speech, and Signal Processing*, Montreal, Quebec, Canada, May 2004.
18. Cox, I., Kilian, J., Leighton, T., and Shamoon, T., "Secure spread spectrum watermarking for images audio and video," *Proc. 1996 IEEE Int. Conference on Image Processing (ICIP 96)*, vol. 3, pp. 243-246, 1996.
19. Cox, I., Killian, J., Leighton, T., and Shamoon, T., "Secure spread spectrum watermarking for multimedia," *IEEE Trans. on Image Processing*, vol. 6, no. 12, pp. 1673-1687, 1997.
20. Cuisenaire, O. and Macq, B., "Fast Euclidean distance transformation by propagation using multiple neighborhoods," *Computer Vision and Image Understanding*, vol. 76, no. 2, pp. 163-172, Nov. 1999.
21. Danielsson, P. E., "Euclidean distance mapping," *Computer Graphics Image Processing*, vol. 14, pp. 227-248, 1980.
22. Datta, A. and Soundaralakshmi, S., "Constant-time algorithm for the Euclidean distance transform on reconfigurable meshes," *J. of Parallel and Distributed Computing*, vol. 61, pp. 1439-1455, 2001.
23. Dittman, J. and Nack, F., "Copyright – copy wrong," *IEEE Multimedia*, vol. 7, no. 4, pp. 14-17, 2000.
24. Eggers, H., "Two fast Euclidean distance transformations in  $Z^2$  based on sufficient propagation," *Computer Vision and Image Understanding*, vol. 69, no.1, pp. 106-116, 1998.
25. Falkenauer, E., "A new representation and operators for genetic algorithms applied to grouping problems," *Evolutionary Computation*, vol. 2, no. 2, 1994.
26. Falkowski, B. J., "Forward and inverse transformations between Haar wavelet and arithmetic functions," *Electronics Letters*, vol. 34, no. 11, pp. 1084-1085, 1998.

27. Falkowski, B. J., "Relationship between arithmetic and Haar wavelet transforms in the form of layered Kronecker matrices," *Electronics Letters*, vol. 35, no. 10, pp. 799-800, 1999.
28. Fan, K.-C. and Lui, P.-C., "Solving find path problem in mapped environments using modified A\* algorithm," *IEEE Trans. on Systems, Man, Cybernet.*, vol. 24, no. 9, pp. 1390-1396, 1994.
29. Farid, H., "Detecting steganographic message in digital images," Technical Report, TR2001-412, Computer Science, Dartmouth College, 2001.
30. Fridrich, J., Goljan, M., and Baldoza, A. C., "New fragile authentication watermark for images" *Proc. IEEE Int. Conf. Image Processing, Vancouver, BC, Canada*, pp. 10-13, Sept. 2000.
31. Fridrich, J., Goljan, M., and Hoge, D., "New methodology for breaking steganographic techniques for JPEGs," *Proc. EUSIP*, Santa Clara, CA, pp. 143-155, Jan. 2003.
32. Fujii, M., and Hofer, M. J. R., "Field-Singularity correction in 2-D time-domain haar-wavelet modeling of waveguide components," *IEEE Trans. on Microwave Theory and Techniques*, vol. 49, no. 4, pp. 685-691, 2001.
33. Gader, P. D., "Separable decompositions and approximations of grayscale morphological templates," *CVGIP: Image Understanding*, vol. 53, no. 3, pp. 288-296, 1991.
34. Giardina, C. R. and Dougherty, E. R., *Morphological Methods in Image and Signal Processing*, Prentice Hall, 1988.
35. Grochenig, K., and Madych, W. R. "Multiresolution analysis, Haar bases, and self-similar tilings of  $R^n$ ," *IEEE Trans. on Information Theory*, vol. 38, no. 2, pp. 556-568, 1992.
36. Haralick, R. M., Sternberg, S. R., and Zhuang, X., "Image analysis using mathematical morphology," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 9, no. 4, pp. 532-550, July 1987.
37. Hashimoto, R. F. and Barrera, J., "A note on park and chin's algorithm," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 139-144, Jan. 2002.
38. Hashimoto, R. F., Barrera, J. and Ferreira, C. E., "A combinatorial optimization technique for the sequential decomposition of erosions and dilations," *J. of Mathematical Imaging and Vision*, vol 13, no. 1, pp. 17-33, 2000.

39. Herrera, F., Lozano, M., and Verdegay, J. L., "Applying Genetic Algorithms in Fuzzy Optimization Problems," *Fuzzy Systems and Artificial Intelligence*, vol. 3, no. 1, pp. 39-52, 1994.
40. Ho, S.-Y., Chen, H.-M., and Shu, L.-S., "Solving large knowledge base partitioning problems using the intelligent genetic algorithm," *Proc. Genetic and Evolutionary Computation Conference*, pp. 1567-1572, 1999.
41. Holland, J. H., *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, 1975.
42. Holland, J. H., *Adaptation in Natural and Artificial System: an introductory analysis with applications to biology, control, and artificial intelligence*, MIT press, 1992.
43. Holliman, M. and Memon, N., "Counterfeiting attacks on oblivious block-wise independent invisible watermarking schemes," *IEEE Trans. on Image Processing*, vol. 9, no. 3, pp. 432-441, Mar. 2000.
44. Hsu, C.-T. and Wu, J.-L., "Hidden signature in images," *IEEE Trans. on Image Processing*, vol. 8, pp. 58-68, Jan. 1999.
45. Huang, C. T. and Mitchell, O. R., "A Euclidean distance transform using grayscale morphology decomposition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 16, no. 4, pp. 443-448, Apr. 1994.
46. Huang, J., Shi, Y. Q., and Shi, Y., "Embedding image watermarks in DC components," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 10, no. 6, pp. 974-979, Sep. 2000.
47. Kambhampati, S. K. and Davis, L.S., "Multiresolution path planning for mobile robots," *IEEE Trans. on Robotics Automation*, vol. 2, no. 3, pp. 135-145, 1994.
48. Kanungo, T. and Haralick, R. M., "Discrete half-plane morphology for restricted domains," *J. of mathematical Imaging and Vision*, vol. 2, no. 1, pp. 51-82, October, 1992.
49. Kanungo, T. and Haralick, R. M., "Morphological decomposition of restricted domains: a vector space solution," *Proc. IEEE Conf. Computer Vision and Pattern Recognition, Champaign, IL*, pp. 627-629, June 1992.
50. Kanungo, T. and Haralick, R. M., "Vector-space solution for a morphological shape-decomposition problem," *J. of Mathematical Imaging and Vision*, vol. 2, pp. 51-82, 1992.
51. Latombe, J., *Robot Motion Planning*, Kluwer Academic Publishers, 1991.



52. Laumond, J. P., Jacobs, P. E., Taix, M. and Murray, R. M., "A motion planner for nonholonomic mobile robots," *IEEE Trans. on Robotics Automation*, vol. 10, no. 5, pp. 577-593, 1994.
53. Lin, P. L. and Chang, S., "A shortest path algorithm for a non-rotating object among obstacles of arbitrary shapes," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 23, no. 3, pp. 825-833, May 1993.
54. Lin, S. D. and Chen, C.-F., "A robust DCT-based watermarking for copyright protection," *IEEE Trans. on Consumer Electronics*, vol. 46, no. 3, pp. 415-421, Aug. 2000.
55. Lozano-Perez, T., "Spatial planning: a configuration space approach," *IEEE Trans. on Comput.*, vol. c-32, pp. 108-120, 1983.
56. Menezes, A., Oorschot, P. van, and Vanstone, S., *Handbook of Applied Cryptography*, Boca Ration, FL: CRC, 1997.
57. Miller, M. L., Doerr, G. J., and Cox, I. J., "Applying informed coding and embedding to design a robust high-capacity watermark," *IEEE Trans. on Image Processing*, vol. 13, no. 6, pp. 792-807, 2004.
58. Moulin, P. and Mihcak, M. K., "A framework for evaluating the data-hiding capacity of image sources," *IEEE Trans. on Image Processing*, vol. 11, no. 9, pp. 1029-1042, 2002.
59. Nikolaidis, N. and Pitas, I., "Robust image watermarking in the spatial domain", *Signal Processing*, vol. 66, no. 3, pp. 385-403, 1998
60. Oorschot, P. C and Wiener, M. J., "Parallel collision search with application to hash functions and discrete logarithms," *Proc. 2nd ACM Conference on Computer and Communication Security*, pp. 210-218, 1994.
61. Park, H. and Chin, R. T., "Optimal decomposition of convex morphological structuring elements for 4-connected parallel array processors," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 16, no. 3, pp. 304-313, March 1994.
62. Park, H. and Chin, R. T., "Decomposition of arbitrarily shaped morphological structuring elements," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 17, no. 1, pp. 2-15, Jan. 1995.
63. Pei, S.-C., Lai, C.-L. and Shih, F. Y., "A morphological approach to shortest path planning for rotating objects," *Pattern Recognition*, vol. 31, no.8, pp. 1127-1139, 1998.
64. PictureMarc, Embed Watermark, v 1.00.45, Digimarc Corporation.

65. Piper, J. and Granum, E., "Computing distance transformation in convex and non-convex domain," *Pattern Recognition*, vol. 20, no. 6, pp. 599-615, 1987.
66. Pitas, I., Kaskalis, T., "Applying signatures on digital images, Workshop on Nonlinear Signal and Image Processing," *IEEE Neos Marmaras*, pp. 460-463 June 1995.
67. Reeds, J. A. and Shepp, R. A., "Optimal paths for a car that goes both forward and backward," *Pacific J. Math.*, vol. 145, no. 2, pp. 367-393, 1990.
68. Rencher, A. C., *Methods of Multivariate Analysis*, New York: John Wiley, 1995.
69. Rivest, R. L., "The MD5 message digest algorithm," *Internet RFC 1321*, <http://www.faqs.org/rfcs/rfc1321.html>, April 1992.
70. Rivest, R. L., Shamir, A., and Adleman, L., "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120-126, Feb. 1978.
71. Rosenfeld, A. and Kak, A.C., *Digital Picture Processing*, Academic Press, 1982.
72. Rosenfeld, A. and Pfaltz, J. L., "Sequential operations in digital picture processing," *J. Assoc. Comp. Mach.*, vol. 13, no. 4, pp. 471-494, 1966.
73. Rosenfeld, A. and Pfaltz, J. L., "Distance functions on digital pictures," *Pattern Recognition*, vol. 1, pp. 33-61, 1968.
74. Said, A. and Pearlmana, W. A., "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. on Circuits and System for Video Technology*, vol. 6, no. 3, pp. 243-250, Jun. 1996.
75. Saito, T. and Toriwaki, J.-I., "New algorithms for Euclidean distance transformation of an n-dimensional digitized picture with applications," *Pattern Recognition*, vol. 27, no. 11, pp. 1551-1565, 1994.
76. Serra, J., *Image Analysis and Mathematical Morphology*, New York: Academic, 1982.
77. Serra, J., *Image Analysis and Mathematical Morphology, Vol. 2, Theoretical Advances*, Acad. Press, New York, 1988.
78. Shapiro, J., "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. on Signal Processing*, vol. 41, no. 12, pp. 3445-3462, 1993.
79. Shih, F. Y., King, C. T., and Pu, C. C., "Pipeline architectures for recursive morphological operations," *IEEE Trans. on Image Processing*, vol. 4, no. 1, pp. 11-18, Jan. 1995.

80. Shih, F. Y. and Mitchell, O. R., "Threshold decomposition of grayscale morphology into binary morphology," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 11, no. 1, pp. 31-42, Jan. 1989.
81. Shih, F. Y. and Mitchell, O. R., "Decomposition of gray scale morphological structuring elements," *Pattern Recognition*, vol. 24, no. 3, pp. 195-203, March 1991.
82. Shih, F. Y. and Mitchell, O. R., "A mathematical morphology approach to Euclidean distance transformation," *IEEE Trans. on Image Processing*, vol. 1, no. 2, pp. 197-204, April 1992.
83. Shih, F. Y. and Wong, W., "An improved fast algorithm for the restoration of images based on chain codes description," *Computer Vision, Graphics, Image Processing*, vol. 56, no. 4, pp. 348-351, July 1994.
84. Shih, F. Y. and Wong, W.-T., "An adaptive algorithm for conversion from quadtree to chain codes," *Pattern Recognition*, vol. 34, no. 3, pp. 631-639, March 2001.
85. Shih, F. Y. and Wu, H., "Optimization on Euclidean distance transformation using grayscale morphology," *J. of Visual Communication and Image Representation*, vol. 3, no. 2, pp. 104-114, 1992.
86. Shih, F. Y. and Wu, H., "Decomposition of geometric-shaped structuring elements using morphological transformations on binary image," *Pattern Recognition*, vol. 25, no. 10, pp. 1097-1106, 1992.
87. Shih, F. Y. and Wu, Y.-T., "Combinational image watermarking in the spatial and frequency domains," *Pattern Recognition*, vol. 36, no. 4, pp. 969-975, April 2003.
88. Shih, F. Y. and Wu, Y.-T., "Enhancement of image watermark retrieval based on genetic algorithm," *Journal of Visual Communication and Image Representation*, vol. 16, pp. 115-133, 2005.
89. Shor, P. W., "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal of Computing*, vol. 26, no. 5, pp. 1484-1509, 1997.
90. Simmons, G. J., "Prisoners' problem and the subliminal channel," *Proc. International Conf. on Advances in Cryptology*, pp. 51-67, 1984.
91. Sternberg, S. R., "Grayscale morphology," *Comput. Vision, Graphics, Image Processing*, vol. 35, pp. 333-355, 1986.
92. Strom, J. and Cosman, P. C., "Medical image compression with lossless regions of interest," *Signal Processing*, vol. 59, pp. 155-171, 1997.

93. Sun, Q. and Chang, S.-F., "Semi-fragile image authentication using generic wavelet domain features and ecc," *Proc. IEEE International Conference on Image Processing*, pp. 901-904, 2002.
94. Sussner, P., Pardalos, P. M. and Ritter, G. X., "On integer programming approaches for morphological template decomposition problems in computer vision," *J. of Combinatorial Optimization*, vol. 1, no. 2, pp. 165-178, 1997.
95. Sussner, P. and Ritter, G. X., "Decomposition of gray-scale morphological templates using the rank method," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 6, pp. 649-658, June 1997.
96. Sussner, P. and Ritter, G. X., "Rank-based decomposition of morphological templates," *IEEE Trans. on Image Processing*, vol. 19, no. 6, pp. 1420-1430, 2000.
97. Tang, K.-S., Man, K.-F., Liu, Z.-F., and Kwong, S., "Minimal fuzzy memberships and rules using hierarchical genetic algorithms," *IEEE Trans. on Industrial Electronics*, vol. 45, no. 1, pp. 162-169, 1998.
98. Vincent, L., "Exact Euclidean distance function by chain propagations," *Proc. IEEE Computer Vision and Pattern Recognition*, Hawaii, pp. 520-525, 1991.
99. Vossepoel, A. M., "A note on distance transformations in digital images," *Computer Vision, Graphics, and Image Processing*, vol. 43, pp. 88-97, 1988.
100. Voyatzis, G. and Pitas, I., "Applications of toral automorphisms in image watermarking," *Proc. International Conf. on Image Processing*, vol. 1, pp. 16-19, 1996.
101. Wakatani, A., "Digital watermarking for ROI medical images by using compressed signature image," *Proceedings of the 35th Hawaii International Conference on System Sciences*, 2002.
102. Westfeld, A. and Pfitzmann, A., "Attacks on steganographic systems breaking the steganographic utilities EzStego, Jsteg, Steganos, and S-Tools and some lessons learned," *Proc. 3<sup>rd</sup> International Workshop on Information Hiding*, Dresden, Germany, pp. 61-76, Sep. 1999.
103. Wolfgang, R. and Delp, E., "A watermarking technique for digital imagery: Further studies," *International Conference on Imaging Science, Systems and Technology*, Los Vegas, Nevada, Juillet 1997.
104. Wong, P. W., "A public key watermark for image verification and authentication," *Proc. IEEE Int. Conf. Image Processing*, Chicago, IL, pp. 425-429, 1998.

105. Wu, C. W., Coppersmith, D., Mintzer, F. C., Tresser, C. P., and Yeung, M. M., "Fragile imperceptible digital watermark with privacy control," *Proc. SPIE, Security and Watermarking of Multimedia Contents I*, vol. 3657, pp. 79-84, Jan. 1999.
106. Wu, M. and Liu, B., "Attacks on digital watermarks," *33th Asilomar Conf. on Signals, Systems, and Computers*, pp. 1508-1512, 1999.
107. Xu, J., "Decomposition of convex polygonal morphological structuring elements into neighborhood subsets," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 13, no. 2, pp. 153-162, 1991.
108. Yamada, H., "Complete Euclidean distance transformation by parallel operations," *Proc. 7th Inter. Conf. Pattern Recognition*, Montreal, pp. 69-71, 1984.
109. Yeung, M. and Mintzer, F., "An invisible watermarking technique for image verification," *Proc. IEEE Int. Conf. Image Processing*, Santa Barbara, CA, pp. 680-683, Oct. 1997.
110. Zhao, D., Chen, G., and Liu, W., "A chaos-based robust wavelet-domain watermarking algorithm," *Chaos, Solitons and Fractals*, vol. 22, pp. 47-54, 2004.
111. Zhao, J. and Koch, E., "Embedding robust labels into images for copyright protection," *Proc. of the International Congress on Intellectual Property Rights for Specialized Information, Knowledge and New Technologies*, Vienna, Austria, pp. 21-25, August 1995.
112. Zhuang, X. and Haralick, R. M., "Morphological structuring element decomposition," *Computer Vision, Graphics, and Image Processing*, vol. 35, pp. 370-382, 1986.