uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Qingwen Zeng
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.Sc. (Systems Science)
GRADE / DEGREE

Systems Science
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Issues in Integrating and Implementing an Open Source Toolkit and Demonstration System for Digital Image Watermarks

TITRE DE LA THÈSE / TITLE OF THESIS

John C. Nash
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Andrew Adler

Jiying Zhao

Gary W. Slater
LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES / DEAN OF THE FACULTY OF GRADUATE AND POSTDOCORAL STUDIES

Issues in Integrating and Implementing an Open Source Toolkit and Demonstration System for Digital Image Watermarks

Qingwen Zeng

Thesis Submitted to the

Faculty of Graduate and Postdoctoral Studies

in partial Fulfillment of the Requirements

for the M.Sc. Degree in Systems Science

Program of Systems Science

School of Management

University of Ottawa

Your file Votre référence
ISBN: 0-494-11471-1
Our file Notre référence
ISBN: 0-494-11471-1

Abstract

Free/Libre and Open Source Software (FLOSS) has been an important factor in the software world and it has profoundly changed the way people deal with software products. However, it seems that little job appear to have been done to develop digital watermarking systems as FLOSS. Developing such a system requires multiple-disciplinary knowledge of digital image watermarking techniques, digital image processing, information hiding, computer science technology and cryptography. The demands of these diverse subjects may suggest that building such a system is a difficult task.

This thesis explores the feasibility of setting up an Internet-based digital image watermarking system capable of embedding and extracting text messages as FLOSS. Several digital watermarking algorithms, such as Least Significant Bit, Patchwork, brightness modulation and DFT amplitude and phase modulations were explored. A choice was made to adopt a J2EE (Java 2 Platform Enterprise Edition) browser/server structure to build the embedding/extraction subsystems and to use Java, HTML/XML and Javascript languages to implement the system, as these are independent of specific computing platforms (eg, Windows or Linux). A trade-off analysis on watermark imperceptibility, robustness, reliability and capacity of algorithms was carried out and taken into consideration in shortening the processing time in the empirical system. The thesis work has developed an illustrative framework for future extension and research.

Acknowledgement

A lot of thanks are given to my supervisor Professor John Nash. Without his guide, encouragement and help, I could not have finished this thesis. He taught me scientific and efficient ways of research, and he gave many valuable suggestions in forming my thesis. He encouraged me when I met difficulties and rebuilt my confidence when I felt frustrated. He not only guided me on the right track, but also gave me fully freedom in my academic research. He read my paper in its raw form, and helped me with editing and modifying. His professionalism deeply impressed me and he will always be a model in my future career life.

I also want to thank the committee members who will give comments and remarks and valuable suggestion in shaping the thesis.

I specifically thank Tom Chadwick, who gave me valuable advice on my thesis.

Finally, many thanks go to my parents, who took care of my daughter and give both spiritual and emotional encouragement to support me to go through the hard path of the research. Special thanks given to my dear little daughter, who is always my joy and my hope in the future.

Table of Contents

List of Figures

List of Tables

List of Figures in Appendix

Chapter 1 Introduction

Digital watermarking technology has experienced a profound development and has been playing a major role in protecting copyright in digital media since 1992 [7]. Originally, this significant research was triggered by the intention to protect copyright of intellectual property in digital formats. By embedding copyright information in digital media in a way that is imperceptible to human senses and extracting copyright messages when needed, copyright owners are capable of protecting and identifying copyright infringement of their work. Copyright protection may be more reasonable than copy protection, and in this regard digital watermarking research has gained wide attention. Numerous watermarking schemes and improvement approaches have been proposed, and new algorithms keep emerging. As the technology evolves, digital watermarking technologies are also applied to fingerprinting, copy protection and image authentication, to name a few. Among several research directions, technologies to watermark digital images have been an active branch of research in digital watermarking. The author of this thesis also has a special interest in image watermarking.

Over the last few years, Free/Libre and Open Source Software (FLOSS) has become a major force in the information technology field and has brought profound and enduring changes to software development. Different from traditional software, FLOSS gives users the freedom to read the source code, modify it, adapt it to their own systems and redistribute for others' to use and reference [12][32]. As Free and Open Source Software matures, FLOSS, which is mostly organized and developed as a project and distributed under a license in free product form, provides powerful tools in better quality and offers more reliable support for users. Today more and more people are joining the huge communities of the Free and Open Source worlds. They benefit from Free Software and contribute to the big communities. Students and researchers from academic background benefit more. They not only reference free source codes of others'

products, but also rely on products distributed by FLOSS projects to build their development and runtime environment. As students and researchers in universities, we noticed the benefits of using and sharing digital images freely and we are also advocates of Free and Open Source software and focus on work in this field.

As far as we can determine, almost no work has been published on a FLOSS digital image watermarking system, especially one implemented as a Web application. After investigating many Free Software and Open Source forums and organizations, we noted that only a few source codes of digital image watermarking algorithms were available from the Internet and no forums or organizations presented projects of digital image watermarking. Similarly, few algorithms appear to have been made available that can be programmed straightforwardly. There exists a lack of resources for people who want to work in this field, especially for those new to digital image watermarking technology.

Therefore, the project aimed to build a feasible implementation of digital image watermarking system using Web technology. It attempted to analyze a few prevailing digital watermarking algorithms and set appropriate parameters for each algorithm. By adapting FLOSS codes and development tools to develop and run the system, we intended to build an image watermarking system accessible on the Internet.

However, we did not find it easy to develop a free watermarking system. Most prevailing digital image watermarking algorithms were presented in the form of theories and were supported by simulations. To understand these algorithms, people need to have a strong background knowledge in several disciplines such as digital image processing and cryptography. In addition, it took much analysis and testing of parameters to take an algorithm from a theoretical stage to a practical one. All of these factors made the thesis research a challenging task.

The following subsections present the origin and current status of digital watermarking and explain the motivation of the author's efforts in exploring free digital image watermarking system in more detail.

1.1 Digital Image Watermarking Motivations and Applications

The rapid development of digital watermarking technology is greatly motivated by the need for protecting digital copyright. Intellectual property is now taking various digital content forms rather than the traditional form of paper books. Audio-CDs, CD-ROMs, DVDs, television and the Internet carry volumes of digital text, image, video and audio information. With the popularity of personal computers, copy equipment and easy access to the Internet, we are now in an era of easy reproduction. Copyright infringement is becoming a great concern for many copyright owners.

There are two major solutions to the copyright infringement issue. One solution is copy protection. The idea is to limit the illegal access to copyrighted materials by inhibiting the copying process. Encryption algorithms are used and users have to pay for the key or buy special equipment in order to gain access to the protected content. This is widely used and still prevails in many industries, especially in music, film, book and software publishing industry [20]. Many organizations have devoted to enforce the digital rights management (DRM) in order to achieve better encryption technology and better protection of digital content. Examples of copy protection include: technical copy protection mechanisms on the media such as CDs, VCDs and DVDs, access controls to copyrighted software through the use of license servers and copy protection mechanism on DVDs and encrypted digital cable television and satellite television broadcast.

Another solution to this copyright infringement is copyright protection using digital watermarking technology. The basic idea is to discourage illegal copying by inserting copyright information (watermarks) into a digital object. The original object's quality should not be affected in a way that is perceptible to human senses. There is no limitation of access to the copyrighted object. Yet, "whenever the copyright of a digital object is in question, this inserted information is extracted to identify the rightful owners" [7].

In spite of the dispute over copy and copyright protection, Copyright protection using digital watermarking technology may be more reasonable than copy protection in our

understanding. Copy protection benefits the owners, but at the same time causes great inconveniences to the rightful users. Copy protection excludes legal uses of the encrypted digital contents from teachers, professors and students. On the contrary, copyright protection laws specifically allow them to make "fair use" for the purpose of education or training on the premises of an educational institution [16]. Presently, even some governments make laws to restrict certain encryption services to protect the benefits of the rightful users [20]. These governmental behaviours have greatly motivated people to develop new methods and algorithms in digital watermarking field.

Being the technology originally applied to copyright protection, digital watermarking technology has also been used in fingerprinting, copy protection, image authentication, depending on users' specific requirements.

Among efforts made in digital text, audio, video and image, digital image watermarking approaches used for copyright protection are the main interests of this thesis.

1.2 Current Status of Digital Image Watermarking

Digital image watermarking technology and its applications have experienced a profound growth over the past several years. Either in academic research or in business companies, this technology has gained extensive interest, attention and efforts. But not much work has been conducted in Free Software and Open Source communities.

In academic field, much research has been done and numerous algorithms and technologies have been proposed. Each year, many academic labs and research institutes publish research papers and declare new and better algorithms. The author noted that much of their research focusing on theoretical analyses and few work done in building an empirical system.

In business world, there are a few commercial companies specializing in digital watermarking technologies and providing products in this area. Alpha-Tec Ltd, Digimarc Corporation and Sugnum Technologies, just to name a few, all supply products for embedding and reading digital image watermarks. As a global leader in digital watermarking technology and

a pioneer in the commercial application of this technology, Digimarc supplies solutions across a broad array of content in audio, video, image and text formats [8]. The product "MyPictureMarc" embeds an imperceptible digital watermark into images that will identify them with copyright information — no matter where they are used online. Table 1 lists some information of the products posted on the websites of the three companies on Dec 22, 2004 [1] [8] [41].

Product Name	**Company**	**Subscription Level (Number of Users)**		**Remark**	**Price(US dollar)**
Signit(Image watermarking)	Alp Vision, Switzerland	First 15		Fee per Unit	0.00
		10			SF49.95
		100			SF249.00
		1000			SF999.00
MyPictureMarc	Digimarc Corporation, USA	Personal: up to 100		Annual cost	USD49.00
		Portfolio: up to 1000			USD79.00
		Collection: up to 5000			USD199.00
		Professional: up to 5000			USD499.00
SureSign Plug-in	Signum Technologies,U	For corpora-tion users	For first workstation	Annual fee	USD1200.00
			For additional workstation		USD220.00
		For individual users (non-commercial users/workstation)			USD220.00

Table 1 Price List of Digital Image Copyright Software

**USD= US dollar, SF=Swiss France

According to the non-profit forum "Digital Watermarking World", none of the digital

watermarking companies seemed profitable until August 2001 [7]. This seems to imply that the commercial potential is uncertain for commercial companies.

In Free Software and Open Source communities, as far as I know, little work has been done in developing a digital watermarking system accessible on the Internet. Some academic researchers build systems in MATLAB, but these systems cannot be accessed on the Internet. Yet, very few source codes of digital image watermarking algorithms can be found on the Internet. Almost no FLOSS organizations conducted research projects in this field. This situation inspired us to devote our efforts to develop a free Internet-based digital image watermarking system.

1.3 Problem Formation

Digital image watermarking technology applied to copyright protection is undoubtedly needed for Free and Open Source Software, in order that creators of images can provide a measure of identification of their work. On the one hand, people advocating Free Software, especially those students and researchers, are willing to share their intellectual products with others. On the other hand, they also want their intellectual property to be respected and have available a method so that their ownership can be publicly recognized when infringed. The concept of copyright protection is important to owners of Free and Open Source Software. It is thus worthwhile to develop free digital image watermarking systems to protect copyright.

Basically, there are two general ways to achieve copyright protection in digital images for Free Software users. One method is to buy commercial products, and the other way is to use some free products. It is impractical for people, especially those students and scholars who have only limited financial resources to spend a lot on commercial products. It is quite reasonable for them to turn to the big Free Software and Open Source communities.

The lack of free digital image watermarking algorithms using Web technologies presented a research problem for this thesis. After searching the Free Software forums and Open Source websites, we found that very few digital image watermarking systems available on the Internet. Furthermore, very few source codes of those watermarking algorithms are available for free. We

summarize some major difficulties as follows.

First, the research is multi-disciplinary. Fields involved in the research include digital image watermarking techniques, digital image processing, information hiding, computer engineering, and cryptography. We need to integrate knowledge from several disciplines and conduct feasibility studies and trade-off analyses.

Second, the implementation of these algorithms is quite demanding. As most digital image watermarking algorithms available are in the form of theoretical analysis, implementation of these algorithms involves a lot of experiments of setting parameters and trade-off analyses.

Finally, the workload of searching Free Software and Open Source resources is heavy. We need to face the vast Free Software and Open Source world and make choices. In order to construct development and runtime platforms and choose language, we need to look into FLOSS resources and select the appropriate products for our special needs. We also need to look for free source codes that are helpful in building the algorithms and adapt them to our system.

To summarize, the thesis work tried to fill in some blanks that exist in the available literature by overcoming these difficulties and developing a prototype digital image watermarking system accessible on the Internet.

1.4 Objectives of the Thesis

The general objective of the thesis is to explore the feasibility of setting up a digital image watermarking system as Free Software. The attempt at an implementation is to provide the essential learning experience. We need to conduct research and development on digital watermarking techniques and implement a free Internet-based digital image watermarking system capable of embedding and extracting text messages.

To achieve the goal, there are several tasks that need to be addressed.

Our first task was to conduct theoretical analyses of some prevailing digital image watermarking approaches presented in academic papers and design corresponding algorithms. We chose a few typical or representative methods from large volumes of papers and numerous

approaches. We focused on those methods that did not rely on the original images to detect hidden watermarks, which are more useful and efficient.

In regard to the design of corresponding algorithms, we worked to make our watermarking scheme satisfy the requirements of a digital image watermarking system. The first priority was to achieve imperceptibility of hidden watermarks. Imperceptibility is the basic requirement in deciding whether an algorithm is suitable or not. If a hidden watermark causes perceptible effect on human eyes, the algorithm is considered not appropriate for this application. We also made efforts to make watermarks robust to image manipulations such as color to gray, cropping and rotation transforms.

Another concern was system performance. Processing time is critical to an empirical system. As images may contain millions of pixels, optimization is considered and applied to algorithms performed in the space domain to shorten processing time.

After analyzing and grasping theories of digital image watermarking, we may be able to design new techniques and approaches.

Our second task was to apply Web technology to the system and design system structure. We determined which Web technology to be used among a variety of Web technologies. We adopted J2EE (Java 2 Platform Enterprise Edition) Servlet and JSP technologies and use Java language. We designed the system as a Browser-server application. We aimed to provide a user-friendly system which supported different image formats and types and provided online help information.

Our third task was to look into Free Software and Open Source to build the development and runtime environment. We chose among a variety of free application servers, Integrated Development Environments (IDEs) and IDE plug-ins. We also searched for available source codes to help build our system.

Finally, we developed and implemented the system. By testing and setting parameters, we inspected the feasibilities of algorithms and implemented them in an empirical system. In addition, we developed a peripheral supporting subsystem which provides image processing

functions such as cropping and image format transform to support the digital image watermarking system.

1.5 Contribution of the Research

The contribution of the research can be classified into the following five aspects:

- Conducted a comprehensive analysis of several prevailing digital image watermarking algorithms presented in available research papers. Rather than a theoretical analysis, the author focused on the feasibilities, performance and implementations of the algorithms.
- Proposed two novel algorithms, which are performed in the frequency domain of the image and are seemingly more robust than those in the spatial domain.
- Deployed a browser-server architecture to construct a prototype Web-based application. This provides a platform for Open Source enthusiasts to exchange ideas and share algorithms for digital watermarking techniques.
- Demonstrated how to conduct development and runtime activities in a FLOSS environment. From the open IDE platform Eclipse, Free J2EE plug in Lomboz to the open J2EE container Apache Tomcat server, from the Java language to J2EE Servlet and JSP technologies, the author can carry out all tasks and contribute to the FLOSS world as well. (We have satisfied ourselves that any components that are not open source can be fairly easily replaced, and we are currently carrying out this task.)
- Implemented variants algorithms by setting and testing various parameters, and analyzing tradeoffs between robustness, imperceptibility, capacity and processing time.

After the brief discussion on the thesis research motivations, tasks and contribution, we go further into the Free Software and Open Source worlds presented in more detail in Chapter 2.

Chapter 2 An Overview of the Free Software Movement

This chapter briefly introduces the Free Software and Open Source worlds. It presents the major active communities and addresses the influences these have on software development and use.

2.1 Origin of Free Software

It had been a dream for people in the software world to freely exchange and share ideas before the emergence of the Free/Libre and Open Source Software (FLOSS) movement. There are several viewpoints, however, and they are not always in full agreement. Free Software Foundation (FSF) summarizes the definition of Free Software [12] as follows,

> "Free Software is a matter of liberty not price. To understand the concept, you should think of "free" as in "free Speech" not as in "free beer"". "Free Software is a matter of the users' Freedom to run, copy, distribute, study, change and improve the software."

The FSF takes a very strict view of what constitutes "Free Software" and has embodied their ideas in the Free Software license. To make a piece of work Free Software, FSF requires releasing it under the most commonly used GNU General Public License (GNU GPL) or occasionally used GNU Lesser General Public License (LGPL) or Free Documentation License (FDL) [12].

GPL Free Software has brought great benefits and advantages to software users. It has gained wide support and extensive interest, with many contributions from a numerous software users. The Open Source Initiative (OSI), founded in 1998, promotes this trend. OSI believes that: "When programmers can read, redistribute, and modify the source code for a piece of software, the software evolves. People improve it, people adapt it, people fix bugs. And this can happen at a speed that, if one is used to the slow pace of conventional software development, seems astonishing." OSI not only presented a new term definition "Open Source Definition (OSD)",

but also sets standards for the distribution of free software. It believes that Open Source doesn't just mean access to the source code and the distribution terms of open-source software must comply with certain criteria. OSI also strongly encourages open software participants to use the term only to describe software which conforms to the OSD standards [32]. Many other workers share many of the goals of the FSF but have some concerns with the restrictions of the GPL so they have introduced other licensing schemes. Hence, OSI works to review and approve many other licenses submitted and makes the list of approved licenses growing.

As FSF summarizes the Free Software and Open Source software [12],

> "The term ``open source" software is used by some people to mean more or less the same category as free software. It is not exactly the same class of software: they accept some licenses that we consider too restrictive, and there are free software licenses they have not accepted. However, the differences in extension of the category are small: nearly all free software is open source, and nearly all open source software is free."

No matter what, the main issue for this work is that the source code is available for implementation within new packaging and platform software has online documentation and technical support. Free Software and Open Source software mean more or less the same catalog. We will use the collective abbreviation FLOSS.

2.2 Free Software Communities

The foundation of the organization of the Free Software Foundation (FSF) marked the first step in building up Free Software community. Since the foundation of FSF in 1985, Free Software movement has experienced a rapid development and a profound growth in just 20 short years. During the time, hundreds of thousands of websites and forums have been built. Countless developers form an extensive and active community of the Free Software world. The author will only present a few crucial and influential contributors in the form of these organizations: Free Software Foundation, Open Source Initiative, Apache Software Foundation and SourceForge.

Free Software Foundation describes itself as follows: [12]

"The Free Software Foundation (FSF), established in 1985, is dedicated to promoting computer users' rights to use, study, copy, modify, and redistribute computer programs. The FSF promotes the development and use of Free Software, particularly the GNU operating system, used widely in its GNU/Linux variant. The FSF also helps to spread awareness of the ethical and political issues surrounding Freedom in the use of software."

The Open Source organization (OSI) contributes to the standardization of Open Source software. As it emphasizes in its website [32]:

"Open Source Initiative (OSI) is a non-profit corporation dedicated to managing and promoting the Open Source Definition for the good of the community, specifically through the OSI Certified Open Source Software certification mark and program."

Apache Software Foundation is an influential organization among the software participants and software industry. As its website [2] introduces,

"The Apache Software Foundation provides support for the Apache community of open-source software projects. The Apache projects are characterized by a collaborative, consensus based development process, an open and pragmatic software license, and a desire to create high quality software that leads the way in its field. We consider ourselves not simply a group of projects sharing a server, but rather a community of developers and users."

Right now, it supports a variety of projects such as Apache Ant project, HTTP server, Jarkata and Struts, to name a few.

SourceForge.net is the world's largest Open Source software development website. It has largest repository of Open Source codes and applications available on the Internet, which are Free to Open Source developers. Until March 17,2005, it has 97,471 registered projects and 1,035,329 registered users [42].

The communities that develop around FLOSS products and services are important to the overall success of both their development and usage, as we shall now describe

2.3 Characteristics of Free Software

There are many benefits of using FLOSS. The advantages for users are summarized as follows:

- No cost to study

Most Free Software can be downloaded freely from the Internet. While usage in certain contexts may require license payments, those who do research or wish to investigate the software are not presented with a financial obstacle.

- Availability of source code

As the source code is available, users can study the algorithm and modify it according to their own usage and requirements. They can embed the required part into their own system (License terms may allow wider use.). They do not need to start from scratch, but rather use the mature products from others and focus on their own special purposes. This greatly shortens development time and increases productivities.

- Online documentation support

Free Software provides program descriptions or installation guide with the source codes. For Free Software which conforms to OSD, OSI has formulates standards of "Free Redistribution", which make sure all distributed software have complete documentations or online tutorials.

- Reliable technical support

Students, professors and scholars find that it is not difficult to get technical support when they are using someone else's source codes. People in the Free Software world are willing to let others test their codes and give advices.

However, the support may not always be free. For some products, there may be agents willing to provide support for a fee. MySQL, most popular open source database, is available under open source licenses, but it also sells commercial licenses [27].

Even business users find that "mature open-source products are far more reliable to begin

with, and that when support is needed it is dramatically cheaper and easier to get than from closed vendors" [32].

- Higher security

As many people devote their talent and wisdom to Free Software, "open-source operating systems and applications are generally much more security-safe than their closed-source counterparts" [32].

- Penetration into industry and commercial world

The Free Software movement derives from a concept totally different from idea of the commercial software. In return, it does and will, gradually but profoundly change the world of software industry. It even begins to contribute to commercial software. OSI has been evolving and dedicating to promote Open Source standards so that Open Source software can be adapted to the commercial world. "For twenty years it has been building momentum in the technical cultures that built the Internet and the World Wide Web. Now it's breaking out into the commercial world, and that's changing all the rules." [32]

Examples include open-source Linux operating system. It is now a big threat to Microsoft Windows operating system. Several millions of Linux users are business users around the world [32].

Another commercial usage of the Open Source model is the Apache Ant project. Apache Ant is a Java-based build tool [2]. Many software companies across the world are using it as a tool to manage Java projects. Small companies use Apache Tomcat server to build their Web applications.

In fact, Free Software and Open Source communities and industries are interwoven with each other. Commercial company Sun Microsystems supports and distributes free java development platform J2SE (Java 2 Platform Standard Edition). IBM has been helping drive Open Source development. "IBM has contributed more than 30 Open Source projects and launched new online skills-building programs to spur innovation, collaboration and development around the emerging Open Source projects" [42].

As researchers in a university, we have decided to build our watermark test and development and runtime platforms as FLOSS and using such components. We will discuss details in Chapter 4 after presenting some fundamentals in digital watermarking algorithms in Chapter 3.

Chapter 3 A Brief Review of some Digital Image Watermarking Algorithms

This chapter presents a few key background materials for understanding the watermarking techniques in Chapter 4. It introduces a few preliminary concepts and fundamentals in the digital image processing field, discusses some basic requirements of a digital image watermarking scheme and presents several digital image watermarking techniques.

3.1 Preliminaries of Digital Image Processing

This section presents some basic notions in the digital image processing field. These concepts, such as digital image formats, image types (gray or color), image brightness scales and Discrete Fourier Transform of images, are fundamental to the research of digital image watermarking techniques.

- Digital Image Formats

Due to the availability of various digital image formats, it is user-friendly to support more image formats when designing a digital image watermarking system. Digital image formats refer to the different file types used to encode digital images. While some old formats are diminishing, new formats are emerging. Some formats are limited to particular applications, and some are for general usage. Among these formats, JPEG and PNG attract our special attention.

The PNG (Portable Network Graphics) format is a patent-free image format. Originally designed to replace the GIF image format for which a patent was claimed by Unisys, PNG uses lossless compression and shows many advantages over GIF format [35] . (Lossless compression implies that the original image can be completely recovered from the compressed file. The alternative is called *lossy* compression.)

JPEG (Joint Photographic Experts Group), another open and patent-free format, is most commonly used on the Internet. It uses lossy compression, and generally has a smaller file size than the PNG format. PNG is usually chosen for archival purposes, while JPEG is considered

good for browsing and transmitting on the Internet.

BMP (Bitmap Picture) is another common format used particularly under the Windows operating system family. An image in this format is large as it saves a pixel map which contains line by line information. Table 2 lists some image formats and their major characteristics

	BMP	**GIF**	**JPEG**	**PNG**
Full Name	Bitmap Picture	Graphic Interchange Format	Joint Photographic Experts Group	Portable Network Graphic
Pixel Bits	8-bit/24-bit	8-bit	24-bit	1->48-bit
Compression	No	Lossless	Lossy	Lossless
File Size	Very Large	Large	Small	Large
Popularity	*	***	*****	**

Table 2 Some Digital Image Formats and Their Major Characteristics

- Gray and Color Images

An image can be seen either as a gray or a color image. It is common knowledge that all colors in the spectrum can be created by mixing the three primary colors: red, green and blue; this is called the "additive color property" [3]. In a digital computer system, this property is described as the RGB model [11]. (Some workers prefer the term "color space" to "model".) A digital color image can be decomposed into three components—R, G and B. Each component can be considered as a "gray image" of one single color.

Besides the RGB model, there are other models, such as the CMY (Cyan, Magenta and Yellow) and the HIS (Hue, saturation and intensity) color spaces. Most image processing systems support transformations between these models. Figure 1 illustrates the three components of a color image.

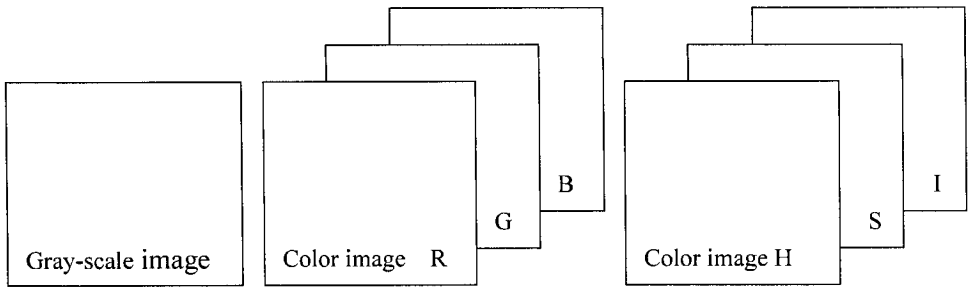

Figure 1 Gray Image and Color Image Space

In the early research of digital image processing, many efforts have been focused on gray images. This is mainly because gray images were popular and color images were not easily processed. The digitization of images makes it easy to decompose color images into three gray images. Some work color images can be derived from the research of gray images by separating the color components. As color images are prevailing everywhere, the watermarking system will support both image types.

- Brightness Scales

In digital images, either a gray or a color image, brightness is quantized and expressed in numbers. For a gray image, gray scales are expressed by various brightness levels. For a color image, each of the three color components (red, green and blue) has its own brightness levels which are similar to those of a gray image.

The actual brightness value saved in a system depends on the supporting package used. Specifically, in J2SE 1.4 system, the brightness level is expressed by an 8-bit number. Hence, the level ranges from 0 to 255 (from the darkest to the brightest). Empirical results have shown the modification of the brightness level can be kept in an image with no great change if the modulated value is between 0 and 255. If a change exceeds this range, software often fails to save a correct value. This may lead to a very great change in the brightness level and cause perceptible distortion. If the brightness level is modified to less than 0, for example, $-x$, the value actually saved in the digital image system is $256-x$. If the brightness level is modified to

greater than 255, say y, the value actually saved is $y-256$.

- DFT--Discrete Fourier Transform

♦ DFT definition

Let $f(x,y)$ represents an $M \times N$ gray image, the DFT of the image is then defined as

$$F(u,v)=\frac{1}{M\times N}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)\times e^{-j\times 2\pi\times u\times x/M\ -\ j\times 2\pi\times v\times y/N}$$

where,

M,N represent the width and height of the gray image

x,y represent the location in the space domain (in original image).

u,v represent the location in the frequency domain.

$F(u,v)$ is a complex number, representing frequency component of the image in the frequency domain. It can also be written as:

$$F(u,v)=|F(u,v)|\times e^{j\times\phi(u,v)}$$

where,

$|F(u,v)|$ is the amplitude of the frequency component. It is also called the amplitude spectrum of the image. It shows the brightness (gray scales) of the image.

$\phi(u,v)$ is the phase of the frequency component. It is also called the phase spectrum of the image. It shows the spatial relationship in the image.

The inverse of the DFT restores the original image using amplitude and phase coefficients. It has the form:

$$f(x,y)=\frac{1}{M\times N}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}F(u,v)\times e^{j\times 2\pi\times u\times x/M\ +\ j\times 2\pi\times v\times y/N}$$

♦ Fast Fourier Transform

The DFT involves a large volume of calculation. It is costly and time-consuming. In practice, a

much faster method called Fast Fourier Transform (FFT) is used in place of the DFT.

- DFT transform of gray and color images

The gray level of a gray-scale image varies in the space and displays certain periodic patterns, which are termed the "spatial frequency". This leads to the creative technique of Fourier Theory invented by French physicist Joseph Fourier applied to the field of image processing [3]. By applying the Fourier theory, any gray image can be analyzed as a sum of several sine and cosine functions of different frequencies.

A color image is the threefold version of a gray-scale image (See above Figure 1). The DFT can be performed on any of the three components of a color image. Due to the imperceptibility requirement of an image watermarking system and the fact that the human visual system is less perceptive of the blue component, the transform is generally performed on this blue component.

3.2 Fundamentals of Digital Image Watermarking Techniques

This section presents the definition of digital watermarks and the digital image watermarking process. It also introduces two other disciplines—cryptography and steganography that have made contributions to the digital image watermarking techniques.

According to the Digital Watermarking World organization [7], the first non-profit forum dedicated to digital watermarking research, a watermark is hidden data that is imperceptibly added to the cover-signal. Blue Spike website introduces that "A digital watermark is, in essence, a hidden message, within a digitized image, video or audio recording" [5]. Digital Watermarking World website also introduced that watermarks can be either visible or invisible depending on the application requirement [7]. Despite the various definitions and their occasional contradictions, most applications demand invisible watermarks. Our research focuses on invisible digital watermarks.

Digital image watermarking is the process of embedding or extracting information (watermark) into or from a digital image. From the techniques proposed in this field, we can see

that many disciplines are involved in the research progress.

Cryptography and steganography are two related disciplines that have played important roles in the development of digital image watermarking technology. Cryptography is "the art and science of keeping messages secure" [38]. It is the major tool used in the DVD and software industries to protect content (copy protection). Cryptographic technologies "scramble" a message so that it cannot be understood, even though it is easy for the encrypted form to be read or copied. Some encryption algorithms protect data in a mathematical vault and use a key to open and use it [20]. The two main approaches are private key encryption and public key encryption [38]. In addition to usage in copy protection, digital watermarking borrows the idea of using an open algorithm with a secret key to get locations to hide watermarks. The private key encryption method adopted in this thesis will be discussed in more detail in the following Section 3.6.

Steganography is derived from Greek and means "covered or hidden writing". It is used typically when point-to-point communication between two parties can be observed by a third party, and those communicators do not want the third party to know the exchange of information between them. Therefore, its primary goal is to hide secret data within a larger communication in such a way that others cannot discern the presence or contents of the hidden message [20].

Technically, steganography and digital watermarking are two sub-disciplines of information hiding or embedding techniques, but with different goals which lead to different technical requirements and philosophies. With steganography, effort is made to make the secret information "invisible" and "secure" so that the third party cannot confirm that a secret message is hidden in the digital object. Digital watermarking emphasizes the resilience of watermarks to attacks. Even if the existence of a hidden digital watermark is known, it should be difficult for the attackers to destroy it while still maintaining the integrity of the original image. In other words, the destruction of digital watermarks should lead to a great change to the original image.

Depending on specific requirements and needs, digital watermarking technology has been applied to several fields, such as copyright protection, "fingerprinting", copy protection and

image authentication. Copyright protection is achieved by embedding watermarks (copyright information) into images. "Fingerprinting" is to hide serial numbers or a set of characteristics that tend to distinguish an object from other similar objects. Fabien a. p. Petitcolas, a pioneer and leading researcher in the digital watermarking field, pointed out that "fingerprinting" can be used to detect copyright violators and "watermarking" to prosecute them. [20].

3.3 Watermarking Techniques – Requirements of Digital Image Watermarking

Conceptually, a digital image watermarking algorithm should conceal messages into digital images imperceptibly. It should also try to make the embedded messages difficult to remove. Some basic properties required in a watermarking system are:

- *Oblivious* vs. *non-oblivious* watermarking

Oblivious watermarking methods do not need the original image to extract the embedded watermark. In contrast, a non-oblivious method needs the original image. Though more challenging technically, oblivious methods are gaining more research attention and are of greater practical usage.

- Security

In digital watermarking, security means a third party, without knowing the secret or private key, cannot retrieve the message embedded in the image, even if he knows the embedding algorithm. Security is often accomplished by selecting "secret" locations in an image to embed information.

- Imperceptibility of the watermark

This most fundamental requirement makes sure that an image is not perceptibly distorted by embedding messages. It also marks whether an algorithm has practical usage. (Standards such as PSNR in digital image processing are introduced, but are not discussed in our work)

- Robustness of the watermarking algorithm

Robustness requires that a watermarking method should resist distortions introduced by either malicious or accidental data processing or attack. No perfect method has been proposed so far [7]. "Robustness" is more application-related than a standardized solution.

- Capacity of the watermark

Watermark capacity is the amount of information embedded in an image. A watermarking algorithm with low embedding capacity may mean the algorithm has no practical usage.

- Reliability

Reliability is the probability of correct detection. A higher reliability means a higher detection rate or a lower error rate.

Depending on application requirements, different algorithms may be needed to satisfy the above requirements at different degrees. Furthermore, there always exist certain trade-offs among these requirements. For example, the imperceptibility of the watermark may mean less robustness of the watermarking algorithm. A better image quality with no human perceptibility may mean less data embedded in an image. Imperceptibility, robustness, reliability and capacity may not all be achieved at the same time.

3.4 Watermarking Techniques – Processing in the Space Domain

Since 1992, a wide range of digital image watermarking techniques and algorithms have been proposed. Generally speaking, these algorithms can be classified into two main categories: according to whether they are performed in the spatial domain or in the frequency domain. The spatial domain methods manipulate the pixels' spatial attributes (such as pixel's brightness or pixel component bits) in an image in a manner that a human cannot perceive the change. The frequency domain methods transform the image into a frequency domain and manipulate the components, such as the amplitude and phase, again so that the changes are imperceptible.

The following subsections briefly present three techniques performed in the space domain: the least significant bit, the brightness modulation and the patchwork methods. The major principles of frequency transforms are introduced in Section 3.5.

3.4.1 Least Significant Bit (LSB) Method

The LSB method replaces the least significant bits of pixels with the embedded message string bits (0 or 1) [20]. For a color image, the replacement is operated on the least significant bits of the three color components of each pixel. For example, in the case where the embedded message bits are "011" and the original pixel is (00101111 10001000 01000001), the embedded data is (00101110 10001001 01000001).

The LSB algorithm is easy to apply and allows a large quantity of information to be embedded while not resulting in any human perceptibility of the changes, but it is vulnerable to attacks such as cropping and rotations. In order to improve the watermark's robustness and reliability, improved techniques are used.

3.4.2 Brightness Modulation Method

This method, proposed by Martin Kutter [22], embeds watermark bit by manipulating the relationship between a pixel's brightness level and its prediction value computed from its neighboring points. The theory is based on the empirical observation that "a pixel's brightness can be predicted by a linear combination of those values of neighboring points around it and the cross-shaped neighborhood gives the best performance" [22].

First, the prediction value of a pixel's brightness level is computed as:

$$\hat{B}_{i,j} = \frac{1}{c}\left(\sum_{k=-c}^{c} B_{i+k,j} + \sum_{k=-c}^{c} B_{i,j+k} - 2B_{i,j} \right).$$

Here $B_{i,j}$ is the brightness value of an image at location (i, j) and $\hat{B}_{i,j}$ is the brightness prediction value at location (i, j). c is a constant describing the number of neighboring points.

Under normal circumstances, a pixel's brightness value $B_{i,j}$ is expected to be equal to or approach its expected value $\hat{B}_{i,j}$. By modifying $B_{i,j}$ in a range that causes no or small

perceptible change in the image, the relationship between $B_{i,j}$ and $\hat{B}_{i,j}$ is changed, which implies a watermark bit is embedded. Let $B_{i,j}$ be slightly greater than $\hat{B}_{i,j}$ if bit '1' is to be embedded. Let $B_{i,j}$ be equal to or less than the expected value $\hat{B}_{i,j}$ if bit '0' is embedded.

Since the human visual system is least sensitive to the blue component of a color image, an operation on the blue component causes little or no perceptible effect. Therefore this method is performed on the blue component when processing a color image.

3.4.3 Patchwork Method

This famous algorithm, proposed by Bender et al., is based on the statistical assumption that the difference of luminance of two sets of pixels in an image is equal to zero if the pixels are chosen to be random, independent and identically distributed [20].

The luminance values of two sets of pixels are modified respectively in order to hide watermark '1'. In the process of embedding bit '1', a secret key is used to generate random numbers to get the locations of two sets of N-pixel groups X and Y (each containing N pixels). Let $I(x_i)$, $I(y_j)$ represent the original luminance (brightness or intensity) of the pixels in each set and $I'(x_i)$, $I'(y_j)$ represent the modified luminance values of corresponding pixels, here $0 \le i, j \le N-1$. One set of pixels is made brighter by adding a brightness scale of δ ($I'(x_i) = I(x_i) + \delta$) and the other is made darker by subtracting the same brightness scale of δ ($I'(y_i) = I(y_i) - \delta$). Figure 2 shows the process.

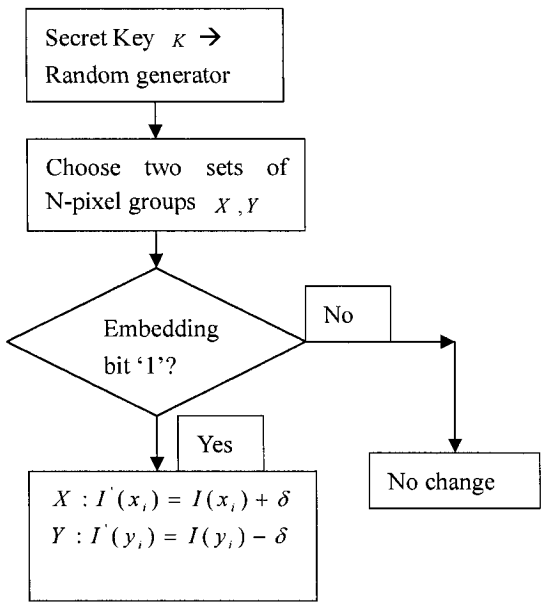

Figure 2 Illustration of the Embedding Process of the Patchwork Algorithm

In the extraction process, the same private key is used to get the same two sets of N-pixel groups as those in the embedding process. The subtraction of the sum of brightness of the two groups of pixels is calculated as ($S' = \sum_{i=0}^{N-1}(I'(x_i) - I'(y_i))$). If the result is equal to $2\delta * N$, we conclude that the watermark "1" is embedded. Otherwise, a watermark '0' is detected.

The above conclusion is based on the following analyses and the statistical assumption $E(S) = E\left[\sum_{i=0}^{N-1}(I(x_i) - I(y_i))\right] = 0$.

If a watermark bit '1' is embedded, the subtraction of the sum of brightness of the two N-pixel groups is calculated as

$$E(S') = E\left[\sum_{i=0}^{N-1}(I'(x_i) - I'(y_i))\right] = E\left[\sum_{i=0}^{N-1}((I(x_i)+\delta) - (I(y_i)-\delta))\right] = E\left[\sum_{i=0}^{N-1}((I(x_i) - I(y_i)) + 2\delta)\right],$$

$$= E\left[\sum_{i=0}^{N-1}(I(x_i) - I(y_i))\right] + E[2\delta * N] = 0 + 2\delta * N = 2\delta * N$$

under the assumption that $E(S) = E\left[\sum_{i=0}^{N-1}((I(x_i) - I(y_i))\right] = 0$.

If no watermark bit is embedded, the subtraction of the sum of brightness of the two N-pixel groups is calculated as,

$E(S') = E\left[\sum_{i=0}^{N-1}(I'(x_i) - I'(y_i))\right] = E\left[\sum_{i=0}^{N-1}((I(x_i) - I(y_i))\right] = E(S) = 0$, under the assumption that

$E(S) = E\left[\sum_{i=0}^{N-1}(I(x_i) - I(y_i))\right] = 0$.

Figure 3 presents the extraction process.

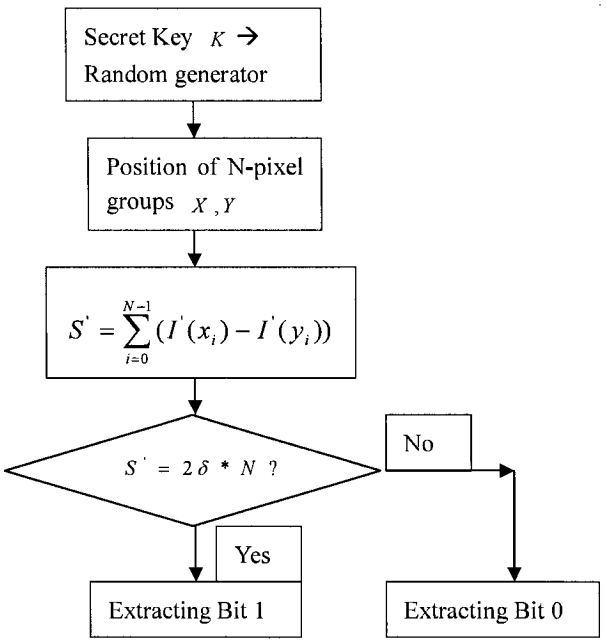

Figure 3 Illustration of the Extraction Process of the Patchwork Algorithm.

3.5 Watermarking Techniques – Processing in the Frequency Domain

Another branch of watermarking techniques is carried out in the frequency domain or the

transform domain. A few prevailing methods include the Discrete Fourier Transform (DFT), the Discrete Cosine Transform (DCT), the Mellin-Fourier Transform and the Discrete Wavelet Transform (DWT) [20]. Here we focus on the DFT frequency methods and conducted analyses on the methods of manipulating two components in the frequency domain—the amplitude and the phase. Since the objective of this research is to provide a framework for investigating the major approaches to digital image watermarking, the DFT methods were chosen as examples to server as a prototype for others to be implemented.

In the DFT frequency domain, both amplitude and phase modulations must comply with the "negative symmetry" principle of the DFT transform [20]. Otherwise, the transformed image will be distorted and unexpected visual effect may happen. Suppose $A(u,v)$ is the amplitude component and $\phi(u,v)$ is the phase component of an image in the DFT frequency domain.

For the amplitude $A(u,v)$, a change to it must obey the rule:

$$A'(u,v) = A(u,v) + r$$
$$A'(N-u,N-v) = A(u,v) + r$$

For the phase $\varphi(u,v)$,

$$\phi'(u,v) = \phi(u,v) + \delta$$
$$\phi'(N-u,N-v) = \phi(u,v) - \delta$$

Among several techniques proposed in the DFT frequency domain, the quantization index modulation (QIM) method and the spread spectrum technique are commonly used. Some algorithms need the original image for the extraction process while others don't. We prefer algorithms that do not need original images in the extraction process as no large volume of storage space is required.

3.6 Watermarking Techniques – Secret Key Watermarking

In order to hide the existence of invisible watermarks from a third party, owners of watermarks want to make sure that their watermarks cannot be easily removed. If possible, the location where the watermark is concealed should be difficult to determine. These issues are referred to as watermark security [20]. To implement watermark security, we can borrow many of the techniques and much practical wisdom from cryptography [20].

Cryptography is "the art and science of keeping messages secure" [38]. Auguste Kerckhoffs presented the first principle of cryptographic engineering as "the method used to encipher data is known to the opponent, so security must lie only in the choice of key" [20]. This means that the security is based in the key rather than in the algorithms. The algorithms can be published and analyzed and all the security in the algorithms is based on the key [20].

Figure 4 and Figure 5 show the encryption and decryption process of the cryptographic algorithm using encryption and decryption keys.

There are two general types of key-based cryptographic algorithms. One is symmetric, and the other is the public-key method. Symmetric algorithms, also called secret-key algorithms, are "algorithms where the encryption key can be calculated from the decryption key and vice versa."[38]. In most cases, these two keys are the same. In public-key algorithms, the encryption key is different from the decryption key. The decryption key can not be calculated from the encryption key. The encryption key is also called public key as it can be made public. The corresponding decryption key is also called private key as it is usually held by specific persons [38].

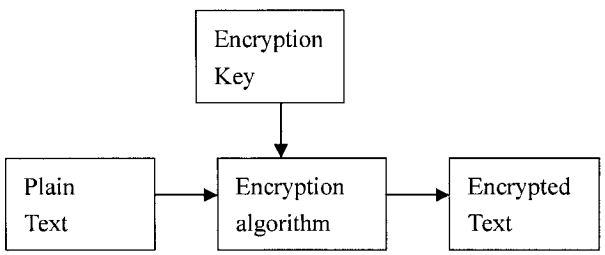

Figure 4 Encryption Process of the Cryptographic Algorithm

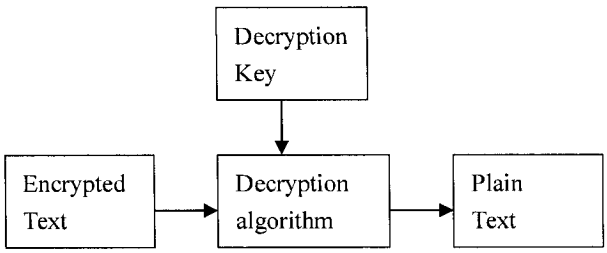

Figure 5 Decryption Process of the Cryptographic Algorithm

In digital watermarking, the application of Kercknoffs' principle is that the watermarking algorithm is public and known to all and the watermark is secure and can not be accessed easily in order to prevent casual removal. This is achieved by choosing the secure locations to hide watermark messages [20].

Borrowing ideas from cryptographic algorithms, watermarking algorithms can be grouped into secret key and public key algorithms. The secret key is held by the owner of digital images and he/she is the only one who can embed and extract the watermark. In many implementations, a secret key is used as a seed to a Pseudo Random Number Generator (PRNG), which produces a sequence of values based on the seed—secret key. The public key algorithm allows the public recovery of the watermark and two keys to be used: the private key is used to embed the watermark and the public key is used to extract and verify watermark [20].

In our watermarking system, a secret key scheme is appropriate based on the following trade-off analyses.

First, a secret key scheme satisfies the image owners' needs. Often owners of images do not necessarily want the public to retrieve the embedded watermarks. They prove the ownership of images by retrieving the copyright information using secret keys or passwords. The flaw is that owners take the risk of being unable to verify the ownership if they forget or lost keys.

Second, a public key method is costly in computation and much complex in implementation and management. Theoretically, a public key scheme is better and more suitable

for applications on the Internet, but they cost more and need more computational and managerial effort. For a relatively small watermarking system, such effort may not be worthwhile.

Finally, a secret key method is easily implemented in the Java system. J2SE API provides classes such as java.util.Random that support a PRNG to implement a private key method. An instance of the class is used to generate a stream of pseudorandom numbers

Therefore, we used the PRNG approach to choose locations to embed copyright messages in the implementation system. Later, the same secret password is used to retrieve the embedded message. Figure 6 and Figure 7 below briefly show the embedding and extraction watermarking algorithms using the same secret key.

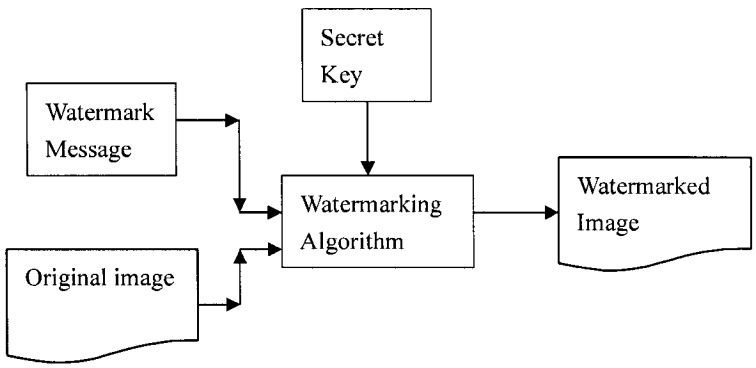

Figure 6 Embedding Process of a Watermarking Algorithm Using a Secret Key

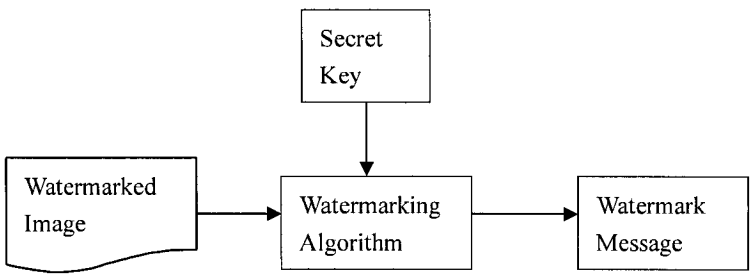

Figure 7 Extraction Process of a Watermarking Algorithm Using a Secret Key

3.7 Watermarking Techniques – Formatting the Watermark Bits

Before the embedding process, watermarks are preprocessed for the purpose of reliability. As many algorithms rely on statistical assumptions, the correct detection or retrieval of one watermark bit is a probability issue. Adding detection bits lowers the rate of error detection, thus improves reliability, as we shall now discuss.

3.7.1 Hamming Code

Hamming code is one of the most commonly used types of error correcting codes [48]. It is an error correction scheme used to verify the integrity of data by parity checking. It is capable of correcting one error or detecting two errors, but not capable of doing both simultaneously.

Generally, the Hamming code is expressed in the form of the ordered set (d, p). Here, d is the number of total data bits, including the original data bits and parity bits. p is the number of parity bits. Certain rules are applied to the original data bits to form the parity checking bits.

The Hamming code $(7, 3)$ uses 3 parity bits to detect and correct 1 bit error among the original 4 data digits. Each parity bit is computed on a different combination of the original data bits. Let d_1, d_2, d_3, d_4 represent the original 4 data bits and p_1, p_2, p_3 represent the generated 3 parity bits. To get these 3 parity bits from the 4 original data bits, the calculation is as follows:

$$p_1 = d_1 \oplus d_2 \oplus d_3$$
$$p_2 = d_1 \oplus d_3 \oplus d_4$$
$$p_3 = d_2 \oplus d_3 \oplus d_4$$

In order to correct the possible errors in the original 4 data bits, 3 check bits c_1, c_2, c_3 are generated to assist the detection process:

$$c_1 = p_1 \oplus d_1 \oplus d_2 \oplus d_4$$
$$c_2 = p_2 \oplus d_1 \oplus d_3 \oplus d_4$$
$$c_3 = p_3 \oplus d_2 \oplus d_3 \oplus d_4$$

These three check bits are used to find out possible 1 bit error in the original 4 data bits:

Flip d_4 : $d_4 = 1 \oplus d_4$, if $c_1 + c_2 + c_3 = 3$; or

Flip d_1 : $d_1 = 1 \oplus d_1$, if $c_1 + c_2 = 2$; or

Flip d_2 : $d_2 = 1 \oplus d_2$, if $c_1 + c_3 = 2$; or

Flip d_3 : $d_3 = 1 \oplus d_3$, if $c_2 + c_3 = 2$.

As we have known, the Hamming code $(7, 3)$ uses 3 parity bits to detect and correct 1 bit error among the original 4 data digits. Accordingly, 6 parity bits are needed to correct 2 bit errors in 8 data bits. Another version of the Hamming code $(13, 5)$ uses 5 parity bits to correct 1 bit error among 8 data bits. Compared to Hamming code (13, 5), Hamming code $(7, 3)$ needs 1 more bit to correct 1 more bit error in 8 data digits, which means more redundancy but more reliable (a higher correction rate).

By adopting the Hamming code $(7, 3)$, the watermarking system improves its reliability of embedded bits information and reduces the detection errors. (As an option, another version of the Hamming code $(13, 5)$ can be supported by the system.)

3.7.2 Data Redundancy

Another way of improving reliability is to embed the same watermarking information in an image for several times. The majority values of extracted bits determine the outcome bit. The more redundancy is introduced, the more accuracy the extraction process has. However, it is at the risk of perceptible distortion and lower processing speed.

Chapter 4 System Design and Implementation

This chapter explores the feasibility of implementing a digital image watermarking system as FLOSS. It describes each stage of implementing a practical system. By comparing several prevailing Web technologies and explaining the advantages of the J2EE technology, it presents a J2EE system structure and a user-friendly environment for developing the system. Note that the system is intended for learning about and testing ideas at this stage, so that only a sample of digital watermarking methods are provided. The system has the capacity to be comprehensive if algorithms of interest are added.

4.1 System Architecture

Our general objective of the thesis is to explore the feasibility of setting up a digital image watermarking system as FLOSS. Furthermore, it is believed that the ease of access to a Web-based system would benefit participants to a greater degree than traditional desktop or client-side applications by avoiding many installation and configuration issues. Therefore, we designed the system to be a Web-based application. People can use this Web-based application or reading the source codes freely, without any restrictions. Furthermore, users, especially those new to the watermarking field, can have a better understanding of the complicated watermarking theories. Using the tools provided, people can watermark their copyrighted digital work. Those so interested can also contribute to the system and help it evolve by uploading new ideas, remarks and source codes to the system.

The system runs in the Browser-Server mode (see Figure 8). A user accesses the system by a Web browser (IE, Netscape or Mozilla, etc). The access request is sent to a J2EE application server (or "container"). Then the application server transfers the request to the watermarking system deployed in that application server. The designated subsystem processes the user's request and returns a response back to the user. In particular, the watermarking system deployed

in J2EE application server is composed of Java servlets and JavaServer Pages (JSPs).

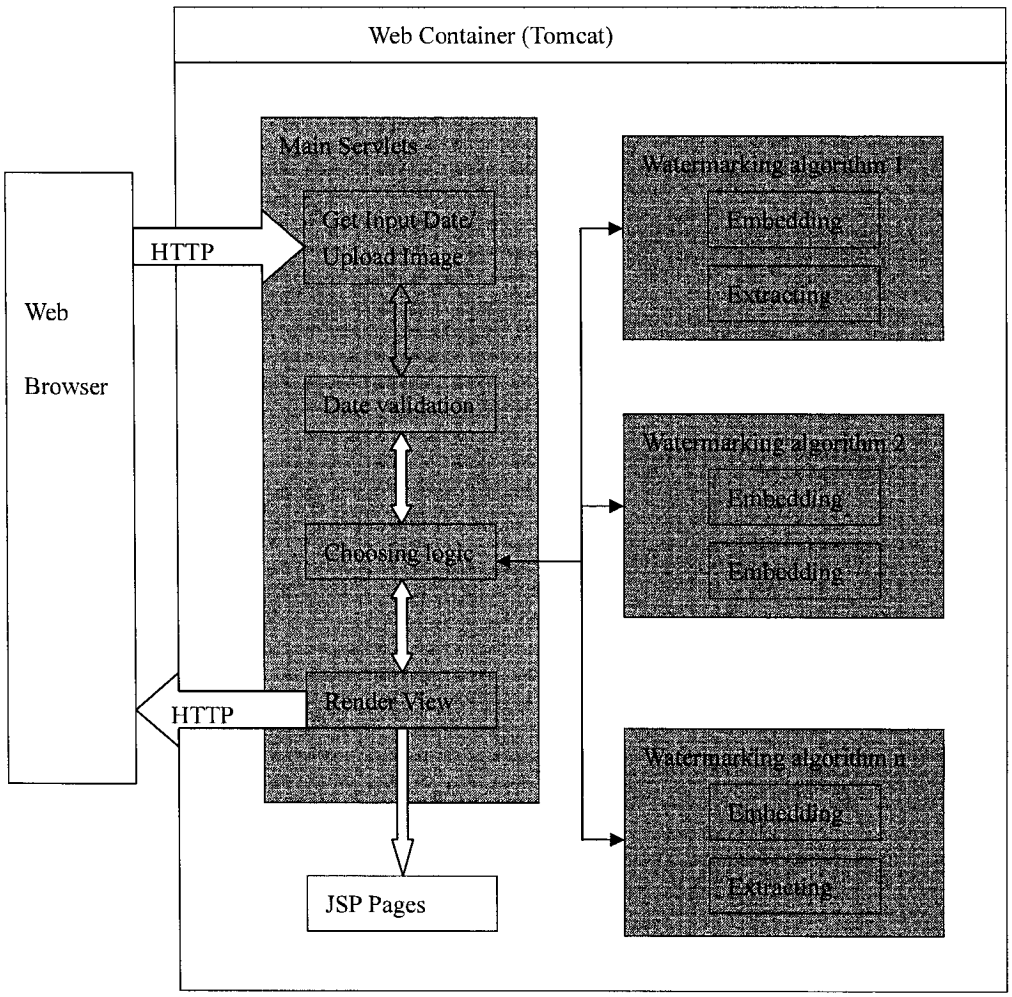

Figure 8 System Structure of the Digital Watermarking System

4.2 Setting Up the Development Environment

4.2.1 Selecting the Development Language

Our system adopts Java and JavaScript languages. Java is used to implement Web services and core watermarking applications, and JavaScript helps to offer help information shown on browsers for end users. Java can run on the server or client machines, but JavaScript, which despite its name is a different programming language, runs always on the client machine, indeed, within the browser.

The reason why Java was chosen is that Java is not only suitable for free Web applications but also well-adapted to FLOSS digital image processing development, as there are a number of available source-code image handling packages[11] [43]. Table 3 lists a few candidates programming tools.

Language	**For Free Web Application**	**For Free Image Processing**
Perl	Good	Slow
C/C++	Not good	Fast
Java	Good	Fast

Table 3 Candidate Language for Development

For developing free Web applications, Perl is another good candidate. Perl is a scripting language developed by Larry Wall in 1986. It has been quite popular among Free Software participants for many years [33]. For powerful and fast image processing, C and C++ are also good candidates. Indeed, Perl generally process digital image by interfacing with external C/C++ libraries [33]. This may cause a longer processing time; C and C++ do not directly support Web services. Therefore, these languages are not suitable for our system.

Java supports free and powerful image processing libraries as well as Web applications [43]. Historically, Java was developed and founded by Sun Microsystems in 1995. It was

designed and modified to be well suited for the World Wide Web. As a high-level language, it is can be considered "open", despite some controversy. Furthermore, it is supported by Sun Microsystems Corporation: Java platform J2SE (Java 2 Platform Standard Edition includes Java 2 SDK (Standard Development Kit) and Java 2 JRE (Java runtime—virtual machine)) can be obtained and used freely under relatively benign licenses, and documentation references are also downloadable from the website [43]. Java has now gained wide popularities in both the FLOSS and commercial worlds. Many Java development tools and source codes can be downloaded. The following summarizes the main advantages of Java language: [45]

- Portability.

Java is platform-independent. Java codes mean "Write once, run anywhere" [14]. They can run on any operating platforms which have installed Java Virtual Machine (JVM) [43].

- Large and well-documented class libraries, APIs (Application Programming Interface) and packages.

These packages mean a java programmer needs not to get started from scratch. Specifically, Java also supports advanced image and graphic processing capabilities, which greatly benefit the watermark processing in images.

- Advanced object-oriented (OO) design
- Automatic memory management

Traditional C/C++ languages allow for pointer operations that permit direct write to memory; this brings larger flexibility but at the same time may cause greater risk for a system. Java manages memory automatically and overcomes the drawbacks of memory leakage (memory area allocated to the program, but can not be used due to lose of reference to the area) or application breakdown occurring unexpectedly. This attribute also makes debugging and spotting bugs easier than other languages.

- A "language for teaching".

Currently, Java Forums and large volumes of free java programs are available on websites, which allows enthusiasts of Java to learn the language more efficiently and adopt the

encapsulated functions to promote productivity of development.

In addition to Java language, JavaScript language is used in order to provide help information about the system. JavaScript is an "interpreted" language that does not need compiling process. It is also object-oriented and is widely used in Web browsers to create dynamic HTML contents.

4.2.2 Building the Development Environment

While constructing the system development platform, we adopt FLOSS tools. No commercial products were used in the implementation of the system. The following lists all FLOSS tools and their corresponding version information applied to the system development.

The operation system: Windows XP (Linux, in various distributions)

J2SE: v1.4.2, including

Java Development kit and Java Runtime: v1.4.2

J2EE Application Server: Apache Tomcat server 5.0

(This server implements Java servlet v2.4/JSP v.20 of J2EE v1.4.)

Integrated Development Environment: Eclipse 3.0

Eclipse plugin for J2EE application servers: Lomboz 3.01

The following presents and analyzes alternatives to satisfy the requirements of developing our system.

1. Operating system

As Java applications can run on platforms installing Java virtual machine (cross platform), the choice of the operating system is not a major concern. More important is the ease with which the programmer can manipulate files and carry out the bookkeeping tasks to build the system, and this ease of use is largely a matter of experience and habit.

2. Java technology—J2SE

As mentioned earlier in Section 4.2, the Java technology has many advantages over competing

technologies. Besides, J2SE is also the basis for Sun's Java Web technology J2EE (The Java platform consists of J2SE, J2EE and J2ME (Java 2 platform Micro Edition) components.).

3. Web technology—Java Servlet and JSP

We faced a lot of choices while designing the development environment: Microsoft.Net vs J2EE, Web server vs J2EE aaplication server, J2EE Servlet and JSP vs CGI (Common Gateway Interface). In the watermarking system, various watermarking algorithms run at the server side. J2EE application server technology supports this need in a more secure and efficient way. It also has the advantages of allowing the system to extend in the future.

Two major contenders of the Web technology are Microsoft.Net and Sun J2EE technologies. Another giant in the Web services field, "Microsoft.NET technology provides the ability to quickly build, deploy, manage, and use connected, security-enhanced solutions with Web services." [24]. Unfortunately, however, Microsoft.NET is a commercial product. Given our FLOSS intention, we did not take it into account.

Choosing the J2EE application servr is based on the following fact. There are two major technologies supporting Web applications. Both the Web server and the J2EE application server technologies support the Browser-server mode, but they are different concepts and have different usages. In the early days of the Web technology, the Web server or HTTP server (eg, Apache) was used to accept requests from the client Web browser, processed the requests and sent HTTP pages generated by the server back to the Web browser. "At its core, a Web server serves static content to a Web browser by loading a file from a disk and serving it across the network to a user's Web browser. This entire exchange is mediated by the browser and server talking to each other using Hypertext Transfer Protocol (HTTP)" [39]. A Web server can also generate dynamic content when CGI is used. However CGI is resource-consuming and prone to safety flaws. As more complicated business logic needs to be processed by the server, another competing technology – J2EE application server – has emerged. It is used with the Web server technology to build a more robust and secure Web application.

Currently in many applications, a Web server is mainly used for processing static HTTP

pages, while a J2EE application server is good at generating dynamic pages and functioning as a middleware, which connects Web front ends to databases or to other J2EE Web servers.

Java 2 Platform, Enterprise Edition (J2EE) technology, supported by Sun Microsystems, defines the standard for developing component-based, multi-tier enterprise applications using Java Servlet, JSP and EJB (Enterprise JavaBeans) technologies based on the Java language. J2EE technology focuses on the distributed multi-tiered structure extendable for complicated enterprise applications [43]. Specifically, Java Servlets are Java classes that dynamically process requests and construct responses. They focus on processing application logic. JavaServer Pages (JSP) are text-based documents (HTML documents) embedded with Java codes, which compile as and execute as Java servlets [43]. Focusing on presentation of information, JSPs provide a simplified and fast way to create dynamic Web content. The EJB approach is not discussed further in this thesis as we chose to use the Apache Tomcat server that does not support them.

Java Servlet and JSP technologies have many advantages over other competing technologies. We briefly summarize these as follows.

First, we examine Java Servlet and CGI programming technologies. Java servlets and CGI programs reside on the server, accept requests, use the server-side resources and generate HTML pages as a response. CGI programming deploys programs to Web servers, and Java servlets run in application servers. Before Java Servlet technology, CGI was the prevailing method to develop applications for Web servers and it is still popular. Lately, Java Servlets have become popular in building dynamic Web sites and connecting Web front ends to databases and applications on a server. Java Servlets have proven to be more efficient, easy-to-use, more powerful, more portable and safer than traditional CGI programming [14]. Table 4 summarizes some of the advantages that Java servlets have over CGI programming.

	Java servlets	CGI programs
Efficiency	Thread-based, more efficient	Process(task)-based, less efficient
Convenience	Java language	Perl/C/C++/Java language
Powerfulness	Can talk directly to server; session tracking; multiple servlets data sharing; database accessing	Can only take to server through sever-specific API
Portability	Servlets is server-independent and platform-independent (may need configuration)	Platform independent for client; but often require very specific server-side configuration
Security	Array bounds checking; Memory protection feature;	May need to filter out certain characters that cause unanticipated side-effects due to interaction with the operating environment; may do not have array or string bounds checking
Flexibility	Dynamically generate Web page	Most static contents; Dynamic content via links to database management tools and similar

Table 4 Java Servlets vs. CGI Programs

We can also compare JSP technology and competing technologies such as ASP (Active Server Page) and PHP (Pre-Hypertext processor or Hypertext Preprocessor) [14] [15]. JSP technology is server-independent and platform-independent: JSP pages follow the write-once, run-any-where philosophy; Rather than restricted to specific type of operating systems and application servers, JSPs are widely supported on any application server or operating platform that has installed a java virtual machine (VM). In contrast, ASP is a presentation technology used

in Microsoft.Net. It embeds VBscript (a form of Microsoft Visual Basic) and Jscript (MicroSoft JavaScript-like tool) in HTML documents, but it can only run on Microsoft platforms. PHP does offer a FLOSS alternative to JSP, but with fewer options for processing on both client and server, and also, as far as we can determine, few image handling tools suitable for digital watermarking. Rather, PHP image handling tools appear to be intended for display manipulation within HTML pages [14] [15].

Before we move further to the next section, we present Apache Http server and Apache Jakarta Tomcat server as example implementations of a Web server and a J2EE application server that are Open Source projects. "The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows NT. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards" [2]. We introduce the Tomcat server in detail in the next section.

4. J2EE application server

Among a few leading J2EE application servers are the BEA WebLogic, JBoss and Apache Jarkarta Tomcat, IBM® WebSphere,® Sybase® EAServer and the integrated Borland® Enterprise Servers. The latter three are commercial products, which are not taken into consideration.

The JBoss Application server, one of the JBoss Federation of Open Source projects, has "Open Source and business friendly license. JBoss is free to download, deploy, and embed." [18]. It is "A full featured, standards-based J2EE application server. The #1 most widely used and deployed application server on the market today." [18]. The latest version JBoss 4.0 is officially certified to be fully compliant to the J2EE 1.4 specification. It also supports EJBs, which support database access. EJBs are considered to focus more on developing middleware solutions and decoupling components [43].

The BEAWeblogic application server is a commercial product of the BEA company. "BEA WebLogic Server, the industry leading application server, combines easy-to-use development,

integration and administration with enterprise-strength clustering, security and reliability for distributed applications." [4]. BEA WebLogic Server is one component of the BEA WebLogic Platform (Other components include WebLogic Integration, WebLogic Portal, WebLogic JRockit, and WebLogic Workshop). It provides a free, scale-limited, non-expiring development license and deployment license for non-commercial usages. However its source codes are not available. Furthermore, it is restricted to five IP connections [4], which limit its utility for building free Web systems.

Being an Open Source subproject of the Apache Jakarta project, Apache Tomcat server is "the servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer Pages technologies" [2]. It supports Java Servlet and JSP specifications developed by Sun Microsystems. The latest 5.x release implements the Servlet 2.4 and JSP 2.0 specifications. However, it does not support EJBs which are considered suitable for complicated business logic [43]. Tomcat is developed "in an open and participatory environment and released under the Apache Software License" [2]. Developers from around the world are welcomed to join and contribute to the project group.

As there is no complicated business logic needed in the anticipated structure of the watermarking system, we chose the Apache Jarkata Tomcat server. Weblogic seems not suitable even for a free, small scale software system due to its limited IP connections. JBoss could be another good solution, and we may leave this option to future work.

5. Java IDE—Eclipse

A Java Integrated Development Environment (IDE) provides developers with an integration platform and enables them to develop, build, debug and deploy Java and J2EE applications using visual controls. Although J2SE and J2EE support the basic runtime environment, it is inconvenient to develop basic Java applications or complicated J2EE applications on this environment. Basically, J2EE and J2SE are inefficient in handling the complete development cycle, which includes coding, debugging, testing or deploying. By using a Java IDE, we accelerate development, testing and deployment of Java and J2EE applications.

Many companies, organizations and individuals devote their time and effort to provide better IDEs. There are a variety of choices that needed to be addressed.

In the business world, Borland® JBuilder®, a commercial IDE product by the Borland Company, supports larger and industry enterprise Java applications [6]. IBM WSAD (WebSphere Studio Application Developers) is IBM's commercial IDE contributing as full-featured Java development environment [17]. BEA WebLogic Workshop is also claimed to support enterprise-scale web applications, Web services, JSPs, Portals, EJB, and Business Process for free [4]. The BEA's offering is free but it is not Open Source.

In the FLOSS world, JCreator, NetBeans and Eclipse are all free IDEs. Among these IDEs, NetBeans and Eclipse appears to be the prevailing and representative tools among Java developers [10] [28].

As the product of NetBeans, an Open Source organization sponsored by Sun MicroSystems, NetBeans IDE is free for both non-commercial users and commercial users. It supports Java developers to develop desktop, Web and mobile applications across different platforms. In 2004, NetBeans 4.0 began to fully support J2EE applications [28].

Organized by the Eclipse Foundation, Eclipse is an open platform developed by the Eclipse project. The Eclipse Foundation, initially named elicpse.org, was formed and sponsored by a few industry leaders such as Borland, IBM, MERANT, QNX Software Systems, Red Hat and Webgain, etc. The organization has played an important role in advocating a universal tool platform - Eclipse. Eclipse presents an open and extensible IDE running on prevailing operating systems such as Linux and Windows. It also provides powerful debugging tool. As its website [10] states,

> "Eclipse has formed an independent open eco-system around royalty-free technology and a universal platform for tools integration. Eclipse based tools give developers Freedom of choice in a multi-language, multi-platform, multi-vendor environment. Eclipse provides a plug-in based framework that makes it easier to create, integrate and utilize software tools, saving time and money."

While both Eclipse and Netbeans could have been chosen to help develop our system, it was necessary to make a choice and Eclipse seems the most appropriate at the time development was started. Note that many users of the system will not care about the IDE used to develop it, though they may find it useful if they choose to add watermarking algorithms. For our development, however, we wished to avoid the need for proprietary software.

6. Eclipse J2EE plugin--Lomboz

In order to run application servers conforming to the J2EE standard in Eclipse, an Eclipse plugin is needed for J2EE applications. Lomboz, Sysdeo and MyEclipse are all Eclipse plugins for application servers.

"MyEclipse is the eclipse plugin solution for all your Web, J2EE, JSP, XML, Struts, JSF, Hibernate and application server integration needs" [26]. However, it needs an annual payment to get a membership. Thus, we do not choose it in the development environment.

Sysdeo is Free and Open Source. Unfortunately, it only supports the Apache Tomcat application server [44].

Lomboz is an open source product under the LGPL License [31]. It "integrates various J2EE components and web application development tasks within Eclipse and some of the most popular application servers", such as Apache Tomcat server, JBoss,WebLogic [31]. It supports the complete development cycle: coding, deploying, testing and debugging of J2EE applications.

In order to allow our system to be extensible and flexible, Lomboz was selected to support more types of application servers. With the Lomboz plugin for Eclipse, our watermarking system can be built, debugged and deployed to most Java application servers supporting Java Servlet and JSP technology.

4.3 Framework of the Application System

Figure 9 outlines the application system and its subsystems as a whole.

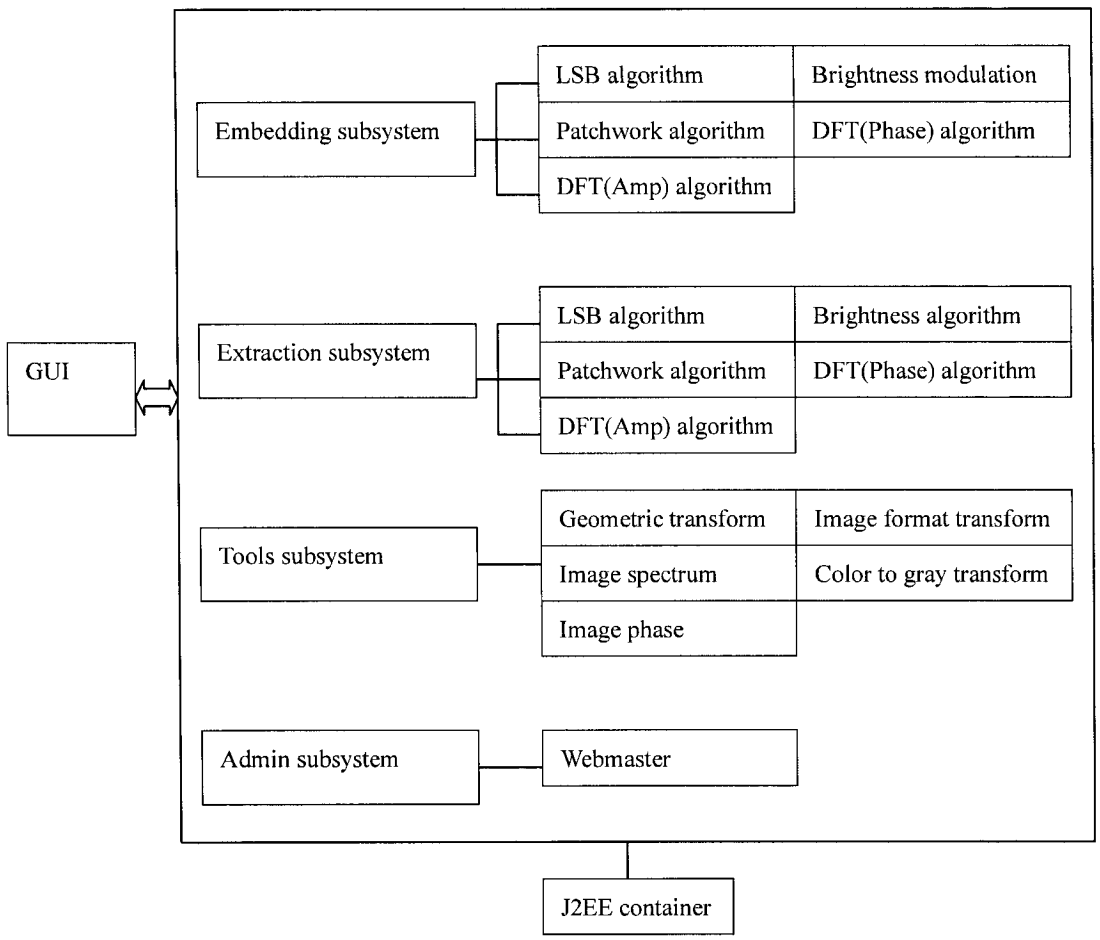

Figure 9 Framework of the Application System

4.3.1 Watermarking Subsystems

A digital image watermarking scheme is designed to meet application requirements. For an application intended for copyright protection where the objective is to hide/retrieve information about ownership, the prevailing trend is to construct a system that does not require reference to the original image. Generally, an image watermarking system consists of two subsystems. The embedding subsystem embeds a watermark message into an image using a secret key or

password and generates an embedded image. The extraction subsystem tries to extract watermarks using the same secret key or password to conclude whether a watermark is embedded.

In addition to the core embedding and extraction subsystems, the system also has an admin subsystem which supports email to webmaster and a tool subsystem which provides some image processing tools.

Figure 10 and Figure 11 show the user interface of the embedding and extraction subsystems on the Web browser.

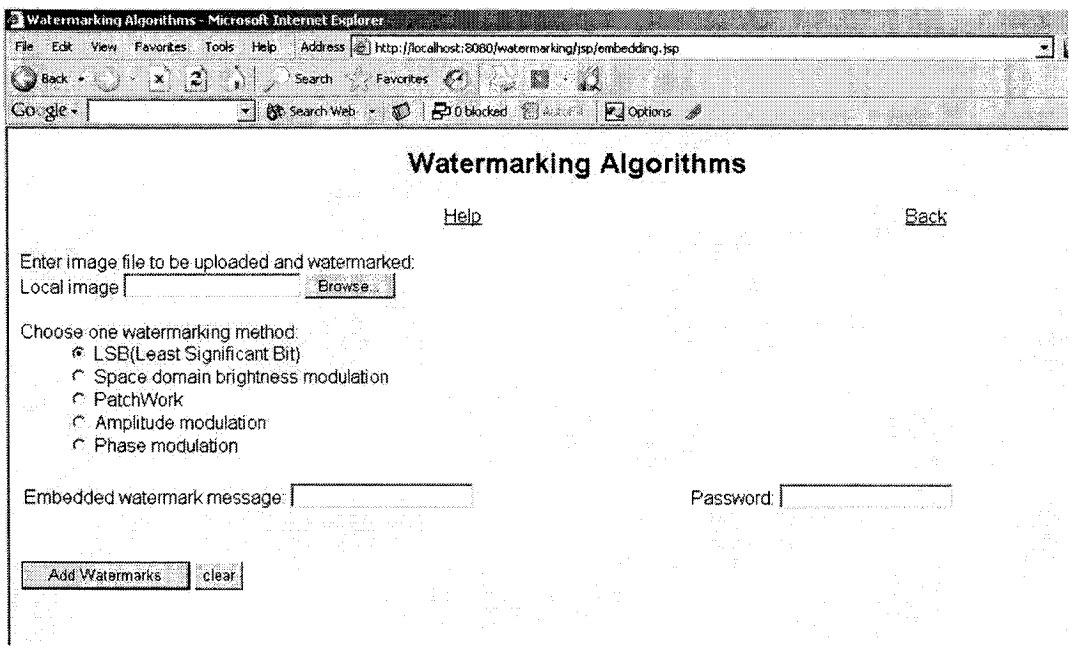

Figure 10 Embedding Subsystem

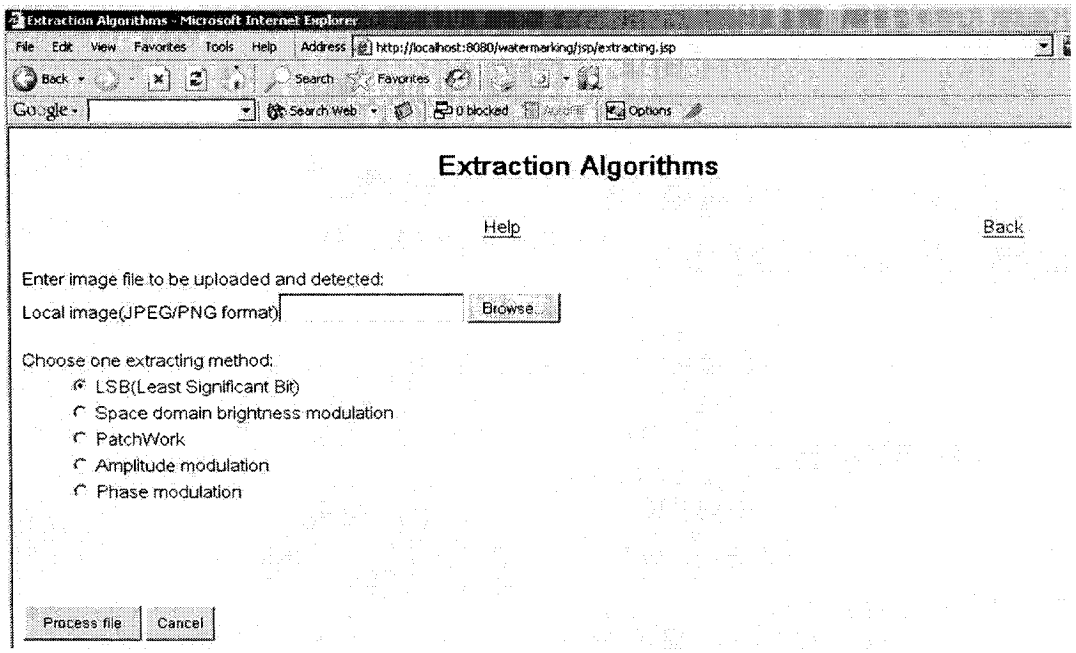

Figure 11 Extraction Subsystem

4.3.2 Implementing Properties of the Digital Watermarks

As mentioned in Section 3.3, several requirements must be satisfied while designing the individual algorithms. The following summarizes our choices in attempting to provide desired properties in the empirical watermarking system.

- Oblivious watermarking

We focused on oblivious watermarking methods. No original image is used in the detection process of each algorithm.

- Security:

The secret key watermarking method (presented in detail in Chapter 3.6) is adopted in the system. Specifically, a PRGN is used to generate locations to hide watermark bits.

- Imperceptibility

Watermark imperceptibility is the most fundamental goal to achieve while hiding a watermark.

An algorithm is unusable if it causes perceptible distortion in an image with a watermark. According to optical theory, the human visual system is least sensitive to the blue among the three color components of an image. Operations on the blue component cause the least or no perceptible effect. Therefore, we chose to manipulate the blue component of a color image, either in the space or in the frequency domain.

- Robustness

The system has been designed to resist attacks such as cropping, rotation and color to gray image transform.

- By embedding the same message all over the image in the space domain, LSB and brightness modulation algorithms can resist cropping. The FFT amplitude and phase algorithms, which are performed in the middle frequency domain, can also resist cropping to a certain degree.

- By designing rotation formulas, all algorithms can detect hidden watermarks when images rotate by 90, 180 and 270 degrees.

- By performing operation on the blue component of a color image and generating a gray image from the blue component, watermarks can survive the color-to-gray transform.

- Processing time

A long processing time means the algorithm is not applicable in an empirical system even if the algorithm itself is correct theoretically. Consideration must be taken to shorten the processing time.

- Reliability

As presented in Section 3.7, data reliability is improved by using Hamming code and adding data redundancy in this system.

We first apply the Hamming code (7, 3) is used to correct possible 1 bit error among 4 bits of data. More detail is discussed in Section 3.7.1.

Second, adding data redundancy greatly reduces detection errors by embedding watermark bits several times in an image. The "voting" majority of values of the extracted bits determine

the outcome bit. Currently, we have set the number of times a bit is embedded to 3 in the system, but this is a parameter that can be altered. If we embed bit "1", we insert "111" in the image. A bit "1" (or "0") is derived if 2 bits of "1"s (or "0"s) are found.

We used the Hamming code method for all five algorithms. For the amplitude and phase modulations performed in the DFT domain, we embedded watermarks three times to overcome a larger detection error in these methods. We conducted the DFT by using the FFT Java class from the image processing packages provided by Nick Efford [11]. (These are copyrighted in a way that precludes our distributing the watermarking system as Open Source software with them included. We are sure that we can find an open source replacement or write a Java class by ourselves. However, we stick to it as being a coherent body of tools, it allows for more rapid coding and verification of the system.) Empirical results have shown that the modification of the amplitude and phase values cannot be kept exactly the same as it was saved. The different setting and getting values lead to a larger detection error. Thus, data redundancy is introduced to lower the detection error.

- Capacity

An algorithm which has small capacity may have no practical usage. It not only means fewer messages could be hidden, but also a lower detection rate due to less or no data redundancy.

4.3.3 Data Validation

To make sure that data entered by users are correct, either in the required length or the correct format required by algorithm, data validation is performed before data are sent to the core part of corresponding watermarking algorithm. There are two alternatives to accomplish this task, either done in the client side by JavaScript or done at the server side by servlets. We choose to construct data validation on the server side.

There are both advantages and disadvantages to using JavaScript to validate input data. The greatest advantage is that no communication between client and server side is needed, thus shortening response and processing time. But not all validations can be done on the client side as

some need data that resides on the server side. Worse, some versions of JavaScript are not fully supported on certain browsers. Another disadvantage is that errors made in JavaScript cannot be easily found, located and debugged. This proves to be a great obstacle for a beginning or intermediate JavaScript programmer.

Servlets are written in Java that can be programmed in many IDE environments by which many programmers, including the author, find very user-friendly. Errors are easily debugged. Although the transmission of data between client and server sides may take time, the data volume is relatively small in this system. From these considerations, we decided to perform data validation on the server side.

In current system, all data input by users are validated immediately after they are uploaded to the server side of the system. The validation process includes, for example,

- all required fields are completed;
- the "password" field is numeric;
- the embedded watermark length is less than 50 characters

If the input data passes such validation, the entered data can be sent to the instantiated algorithm object to execute the specific watermarking algorithm. Otherwise, an error message box is popped up to show an appropriate error message.

For the tool subsystem, data validation is also needed and is performed to make sure that the subsystem gets correct inputs.

4.4 Building the Production Environment

The production environment can be detailed as follows:

The operating system: Windows or Linux

J2SE

Java Development kit and Java Runtime: v1.4.2

J2EE Application Server: Apache Tomcat server 5.0

Watermarking packages: watermarking.war

System working directory: for example d:/uploadDir (a storage directory specified by the user)

The detailed procedures are presented in Appendix B.

The next chapter presents five example algorithms.

Chapter 5 Implementation of Algorithms and Analysis

This chapter analyzes the feasibility of each watermarking algorithm, designing program flow logic and testing empirical parameters for the system. It conducts a brief analysis and comparison of each algorithm, but we note that the focus of the thesis is on providing the capability for empirical analysis and testing of algorithms and not an in-depth review of the particular watermarking methods.

5.1 Flow Charts of the Algorithms

This section discusses in detail how each of the example watermarking algorithms works. Besides presenting program flow charts, we discuss special considerations of imperceptibility, robustness, processing time, security and parameter choices in each algorithm.

The watermarking process (embedding and extracting) can be performed in either the spatial or frequency domain. The spatial domain methods implemented in the current system include the LSB, the Patchwork and the brightness modulation algorithm. Two novel DFT algorithms performed in the frequency domain are proposed ad preliminary tests are reported.

5.1.1 LSB Algorithm

The basic idea of this algorithm is described in the previous Section 3.4.1. The algorithm is composed of an embedding process and an extraction process. In the embedding process, an image is divided into small blocks and watermark message bits are embedded in each block. In the extraction process, blocks located in a small region are tested to extract watermark bits. Figure 12 and Figure 13 illustrate the flow charts of the embedding and extraction processes.

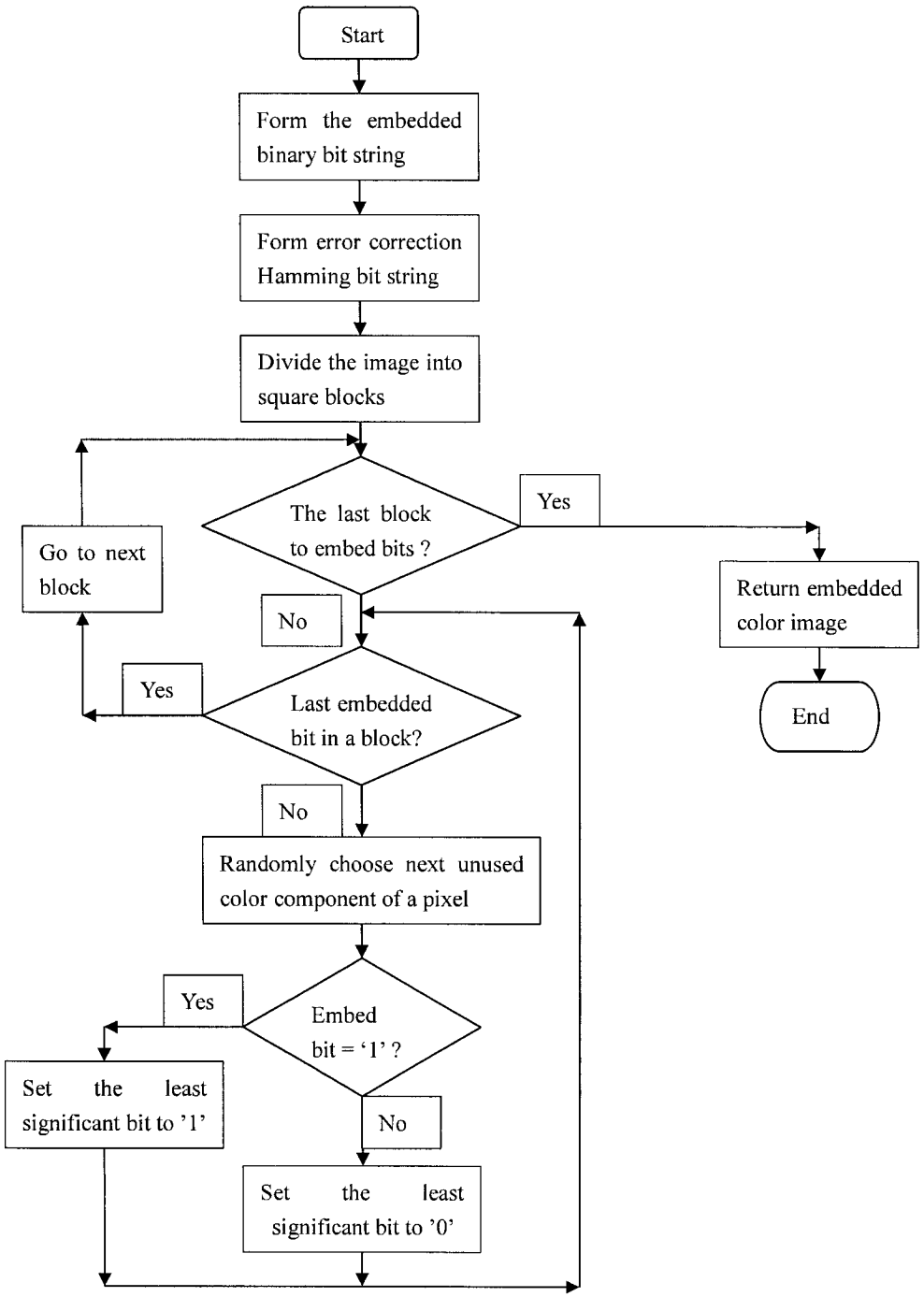

Figure 12 Flow Chart of the Embedding Process for the LSB Algorithm

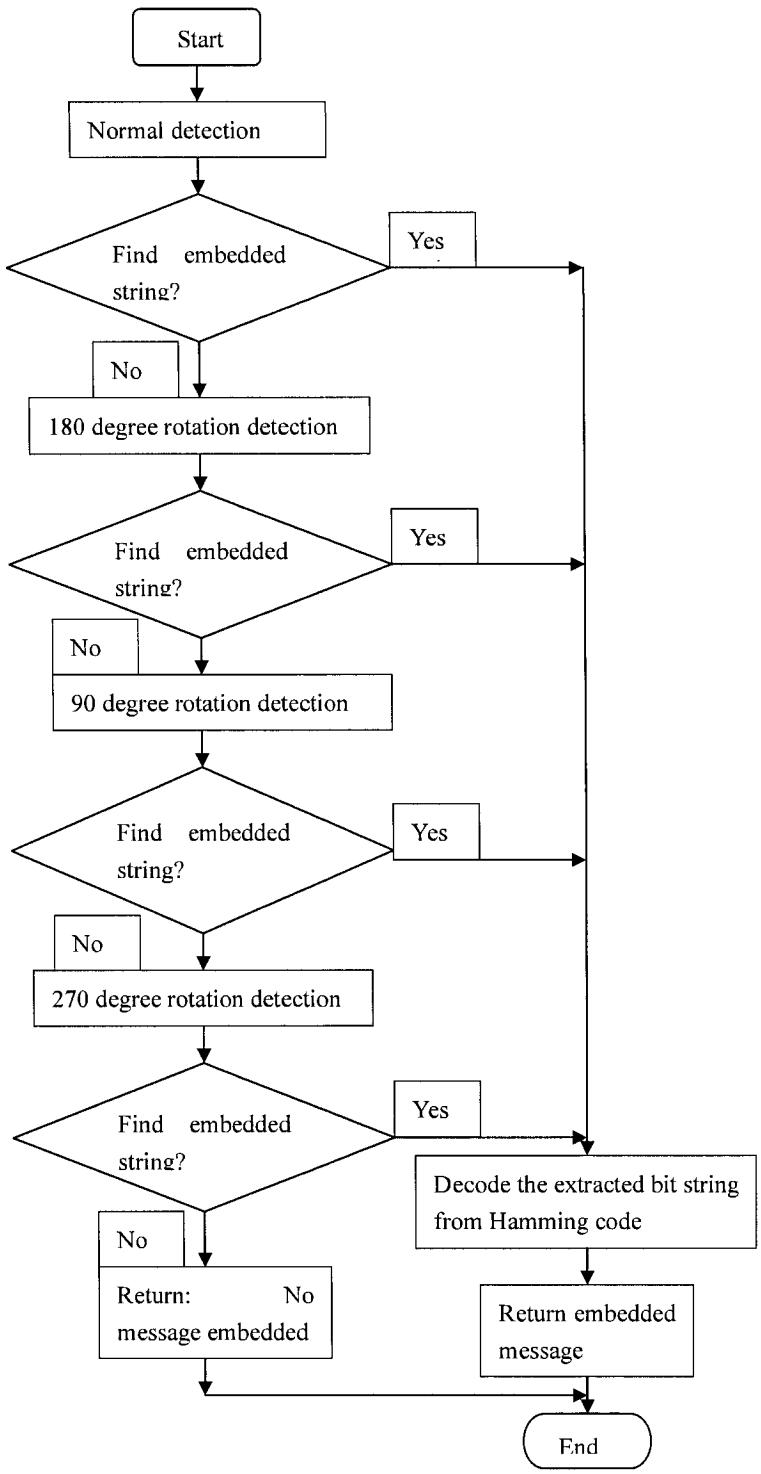

Figure 13 Flow Chart of the Extraction Process for the LSB Algorithm

While planning the program flow, we make sure that the algorithm is designed to conform to requirements of digital image watermarking system. We discuss the issues on imperceptibility, robustness, processing time, security and parameter settings in the system.

1. Imperceptibility

This algorithm has the best imperceptibility quality among the algorithms implemented in this paper, as the changing of least significant bits of image pixels does not cause any image distortion to the human visual system.

2. Robustness

We discuss how to make the watermark survive attacks when image changes from color to gray image and when an image is cropped, rotated by 90,180 and 270 degrees.

In order to make the watermark survive when an image changes from color to gray, least significant bits of three color components of a randomly chosen pixel are all set to the same value ('1' or '0') as the embedded message bit ('1' or '0') for a color image. For a gray image, only the least significant bit of the gray component is set.

In order to make the watermark survive when an image is cropped, the watermark message string is embedded all over the original image in the embedding process. We divide the image into small blocks of the same size and embed the same message content in each block. Figure 14 shows that the image is divided into small square blocks and each small block labeled "w" embeds a watermark message.

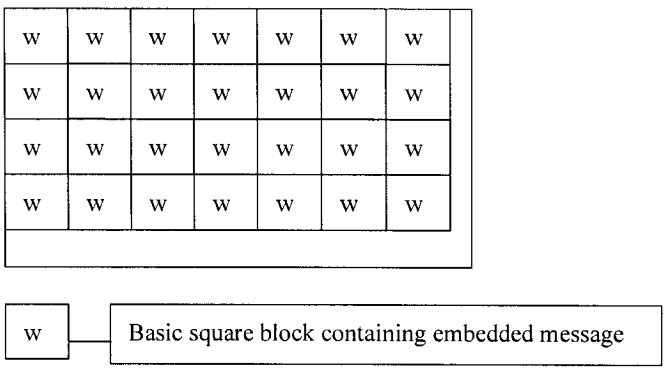

Figure 14 Illustration of the Division of an Image

(Notice that a small portion of the image does not contain any watermark message as shown in the blank area.)

In order to make the watermark survive when an image rotates clockwise by 90,180 and 270 degrees, rotation formulas are considered in the extraction process.

Suppose the block size (side length) is expressed as K, the coordinate in the original image is (row, col), the corresponding position after rotation is (new_row, new_col). Figure 15 shows their relationship under three rotations.

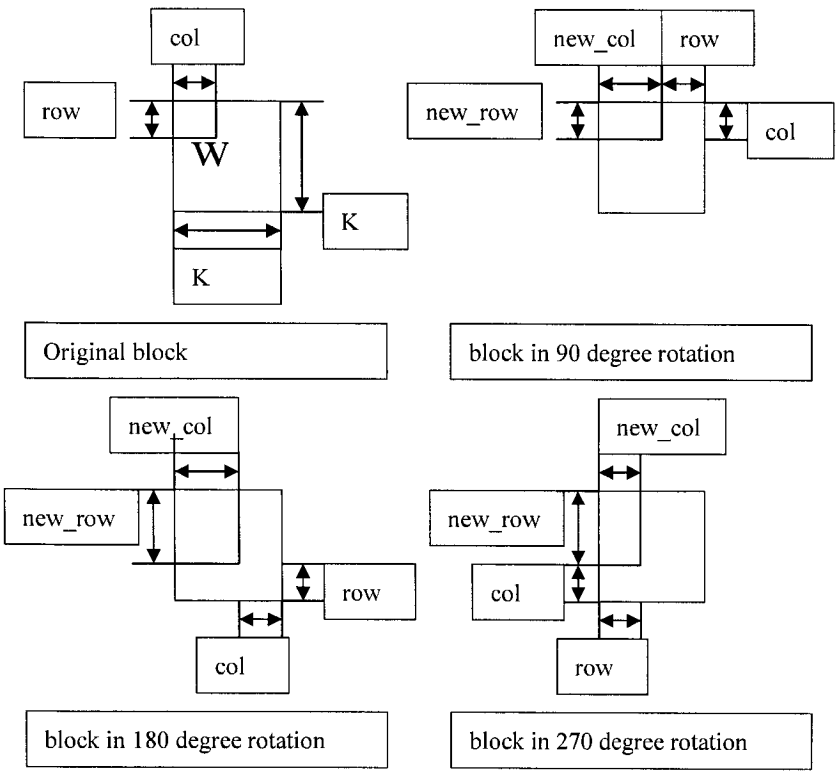

Figure 15 Relationship between a Block and Its 90,180 and 270 Degree Clockwise Rotations

The formulas are:

$$\begin{aligned} new\_row &= col \\ new\_col &= (K-1) - row \end{aligned}$$, for 90 degree clockwise rotation

$$\begin{aligned} new\_row &= (K-1) - row \\ new\_col &= (K-1) - col \end{aligned}$$, for 180 degree clockwise rotation

$$\begin{aligned} new\_row &= (K-1) - col \\ new\_col &= row \end{aligned}$$, for 270 degree clockwise rotation

3. Processing time

Processing time is a great concern for the extraction process. A long waiting time for the resultant message suggests that the algorithm is unacceptable for many users and therefore impractical in most cases for a usable system. However, to reach the correct conclusion in the extraction process, we need to search a larger area of the image. Thus, the extraction process must face the tradeoff between the processing time and the image area searched.

Since the watermarks are embedded all over the image, only a small portion of the image needs to be searched no matter whether the image is cropped or rotated. In our implementation, the search begins with the block in the upper left corner of the image. If a watermark message is found, the extraction process will stop. Otherwise, the next neighboring block area will be processed. Suppose the block size is K (side length of the square block in pixels), which is a parameter as discussed below. If we base our first block at the top left pixel of the image, the first search for a watermark is the block consisting of the first K rows and first K columns of pixels. However, given that our image may have been cropped or rotated, a failure to find a message could mean that we have the wrong origin for our search. We need to retry the search for K^2 possible starting or base pixels for a block.

Figure 16 shows that all possible upper left base pixels of K^2 that lie in the block labeled letter B.

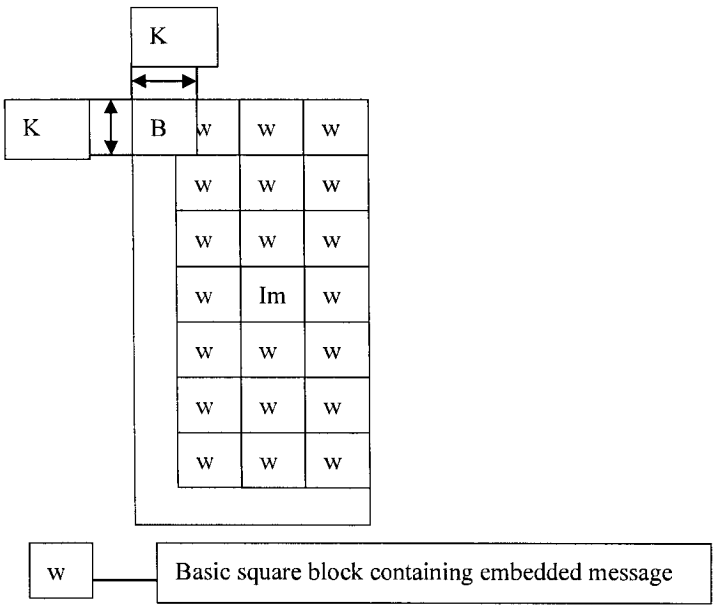

Figure 16 Neighboring Blocks in the Upper Left Corner of an Image

When detecting watermarks in each block, we first check the existence of the header message. If it does not exist, no further extraction of bits is needed. If it exists, we continue extracting bits in the block for a certain length and then check if a watermark is embedded (that is, judge if there is an ending flag). In this way, we greatly save processing time.

4. Security

A secret key or password is used to randomly choose the locations within a block to hide watermark bits in order to prevent possible intentional attacks from easily removing the embedded message content.

5. Parameter issues

Block size is a key issue for an empirical system. For the LSB algorithm, the block is a "small" square one. The size of the block is determined by the number of bytes of information to be embedded in it. Suppose N bytes are embedded in a block. (eg, western language characters needs one byte to represent and Chinese characters need 2 bytes). As the lengths of the header

and the ending flags are h_1 and h_2 bits, the total data bits are $h_1 + N*8 + h_2$. By applying the Hamming code (7, 3) (adding Hamming check bits), the final bit length is $(h_1 + N*8 + h_2)*7/4$. Currently, the maximum message length is $N = 50$, and $h_1 = 40$ and $h_2 = 16$. Therefore, the total length is $(40 + 50*8 + 16)*7/4 = 798 \approx 800$ bits. This requires a block size of at least $\sqrt{800} \approx 29$ bits or pixels per side to embed these bits. The author chose 30 pixels as the block size for the LSB method.

5.1.2 Brightness Modulation Algorithm

The author follows Martin Kutter's theory to implement the spatial brightness amplitude modulation method in a gray image or in the blue component channel of a color image [22]. The general principle has been presented in previous section 3.4.2. In the implementation, 4 neighboring points marked by * are chosen to calculate the expected value of a pixel as shown in Figure 17

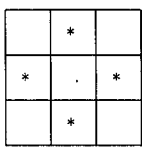

Figure 17 Pixels Chosen to Calculate the Prediction Value of a Pixel

Hence, the general calculation formula of the expected value can be written specifically as follows:

$$\hat{B}_{i,j} = \frac{1}{4c}\left(\sum_{k=-c}^{c} B_{i+k,j} + \sum_{k=-c}^{c} B_{i,j+k} - 2B_{i,j}\right) = \frac{1}{4}\left(\sum_{k=-1}^{1} B_{i+k,j} + \sum_{k=-1}^{1} B_{i,j+k} - 2B_{i,j}\right)$$
$$= \frac{1}{4}\left(B_{i-1,j} + B_{i+1,j} + B_{i,j-1} + B_{i,j+1}\right)$$

where $c = 4$.

When bit '1' is embedded, the brightness of a randomly chosen pixel is set to be greater

than the expected value calculated from neighboring 4 points. In order not to cause human perceptibility of the change, the modification level is set as a parameter and tested, which will be discussed in the following part.

When bit '0' is embedded, the pixel is set to equal to the expected value derived from the 4 neighboring points.

The extraction process retrieves an embedded bit '1' when a pixel's brightness is greater than its expected value and a bit '0' in other circumstances.

The following Figure 18 and Figure 19 show the main procedures of the embedding and extraction processes.

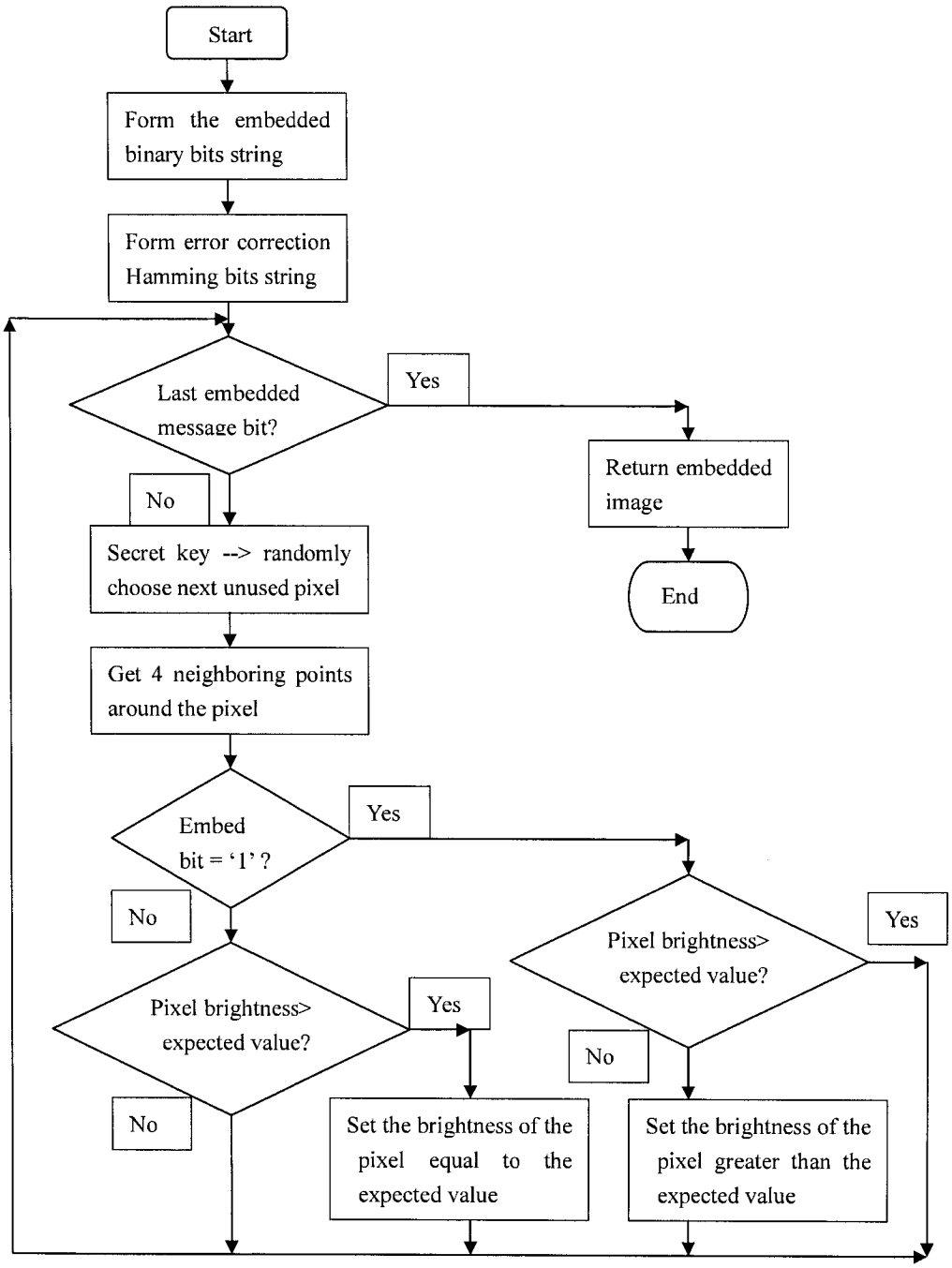

Figure 18 Flow Chart of the Embedding Process for the Brightness Modulation Algorithm

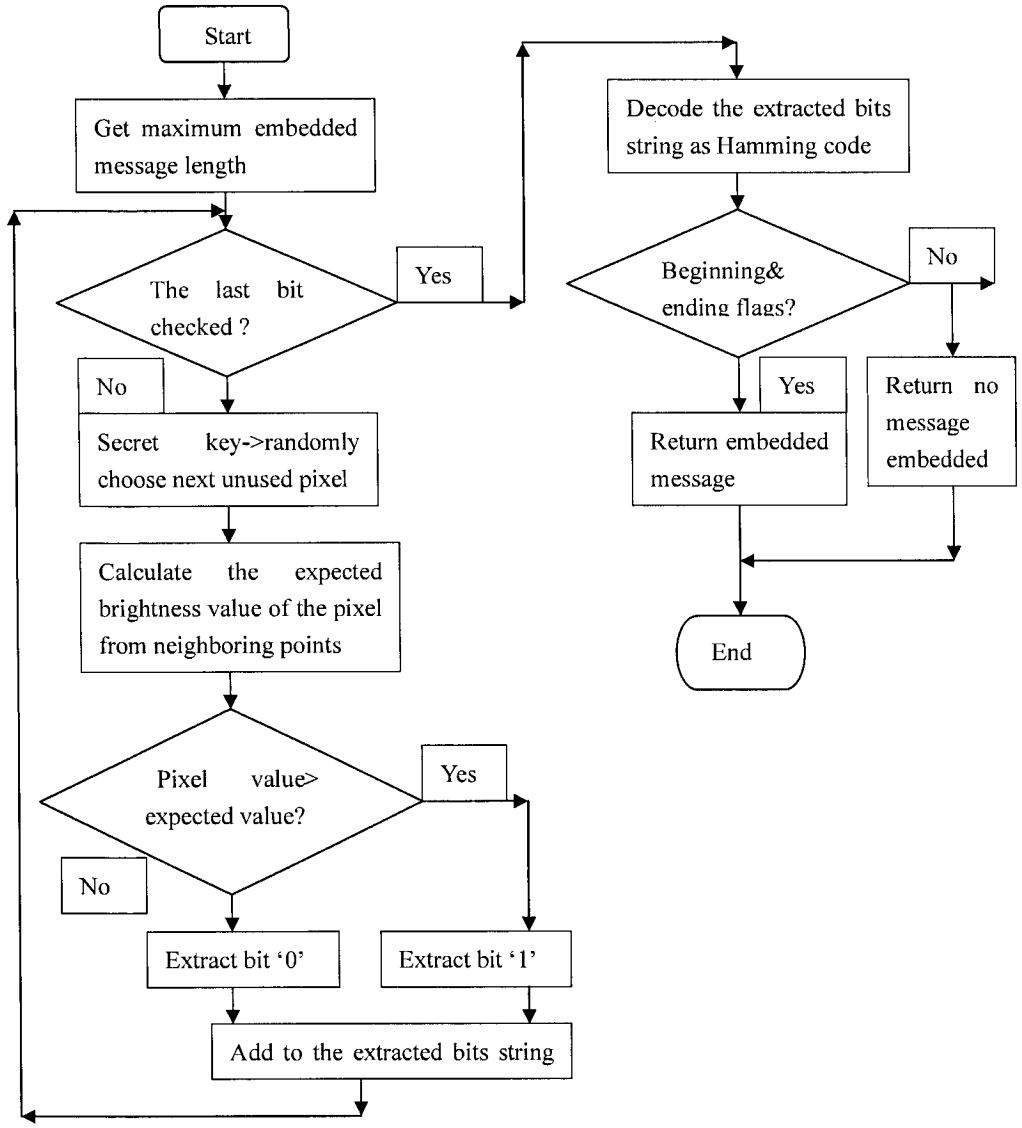

Figure 19 Flow Chart of the Extraction Process for the Brightness Modulation Algorithm

In the implementation of the algorithm, imperceptibility, robustness, processing time, security and parameter settings were considered as follows.

1. Imperceptibility

As the human visual system is least sensitive to brightness changes in the blue component of a color image, the above embedding and extraction process for a color image are all performed in the blue component. Unfortunately, this algorithm may still cause perceptible change in areas with a bright and pure color.

2. Robustness

The same idea as in the LSB algorithm is applied here to make the watermark more robust when an image changes from color to gray and when an image is cropped or rotated by 90,180 and 270 degrees: In order to make the watermark survive when an image changes from color to gray, brightness changes are manipulated in the blue component of a color image or in the gray component of a gray image; In order to make the watermark survive when an image is cropped, the image is divided into several small blocks of the same size (say K) and each block embeds the same message content; In order to make the watermark survive when an image is rotated by 90,180 and 270 degrees, the three rotation formulas are used, same as presented in Section 4.4.1. The only difference is the different block size. Refer to Section 4.4.1 for more detail.

3. Processing time

The consideration here is exactly the same as that for the LSB algorithm discussed in Section 4.4.1: We shorten processing time by checking each of the K^2 blocks for the header message and abandoning a single block if the header is not found. Similarly, once it is found, the extraction process continues for the block.

4. Security

A pseudo random generator is used to randomly select locations to embed and extract watermark bits in the embedding and the extraction processes, as with the LSB algorithm. We mark the neighboring points of selected pixels so that these neighboring pixels cannot be chosen again.

5. Parameter issues

- Block size

Same as the analysis conducted in Section 4.4.1, the square block size is determined by the number of characters embedded in the image. The length of message bits is calculated as $(h_1 + N*8 + h_2)*7/4$ in order to embed N bytes information using Hamming code (7.3). Suppose h_1 is the length of header message bits, h_2 is the length of ending message bits, and N is the length of characters embedded. If $N = 50$, $h_1 = 40$ and $h_2 = 16$, the length of message bits is $(40 + 50*8 + 16)*7/4 = 798 \approx 800$. As the algorithm requires the using of 4 neighboring points, it requires at least $800*(4+1) = 4000$ bits or pixels to embed these bits messages. Therefore, it requires at least $\sqrt{4000} \approx 64$ bits or pixels per side. The author chose 70 pixels as the block size.

- Brightness adjustment level

Brightness levels in the standard Java system are expressed as floating-point numbers and involve precision consideration. Empirical experiments have shown that the brightness value that the Java system retrieves is not exactly the value that was set (this may be due to input-output conversion rounding). The retrieved value is less than or equal to the preset value. For example, the brightness value is retrieved as 24.0 even if it was set as 24.25, 24.50 or 24.75. Thus, we set the brightness adjustment level to 1 to make sure a pixel's brightness is higher than the average value of neighboring points.

5.1.3 Patchwork Algorithm

1. Empirical analysis of the Patchwork algorithm

As discussed in Section 3.4.3, the Patchwork approach is build on the statistical assumption that

the difference of luminance of two pairs of pixels in an image equals to zero if the pixels are chosen randomly and are independent and identically distributed [20]. But this assumption proves not to be true in the empirical environment.

We tested the assumption in images. We used various numbers of pixels in two data pairs and different secret keys to locate pixels. The experiments clearly showed that the luminance difference of two pairs of pixels is not exactly zero. This leads to the revised version of the algorithm.

2. Revised Patchwork algorithm

Although the stated statistical assumption is not exactly true, the proposed method can still be analyzed and improved by using statistical theories. Doo Gun Hong and his colleagues conducted a further analysis in their research paper [9]. Refer to Section 3.4.3, the luminance difference of two pairs of N pixels is $S = \sum_{i=0}^{N-1}(I(x_i) - I(y_i))$. According to the Central Limit Theorem, S is a random variable which has a normal distribution with a sample mean of zero ($E(S) = 0$) when $N > 35$. If a watermark is embedded, S will change to W. Here,

$$W = \sum_{i=0}^{N-1}((I(x_i) + \delta) - (I(y_i) - \delta)) = N * \delta - N * (-\delta) + \sum_{i=0}^{N-1}((I(x_i) - I(y_i)) = 2N\delta + S \quad .$$

Hence, W is also a normal random variable with a mean $E(W) = 2N\delta + E(S) = 2N\delta$. The relationship between these two normal variables is shown in Figure 20.

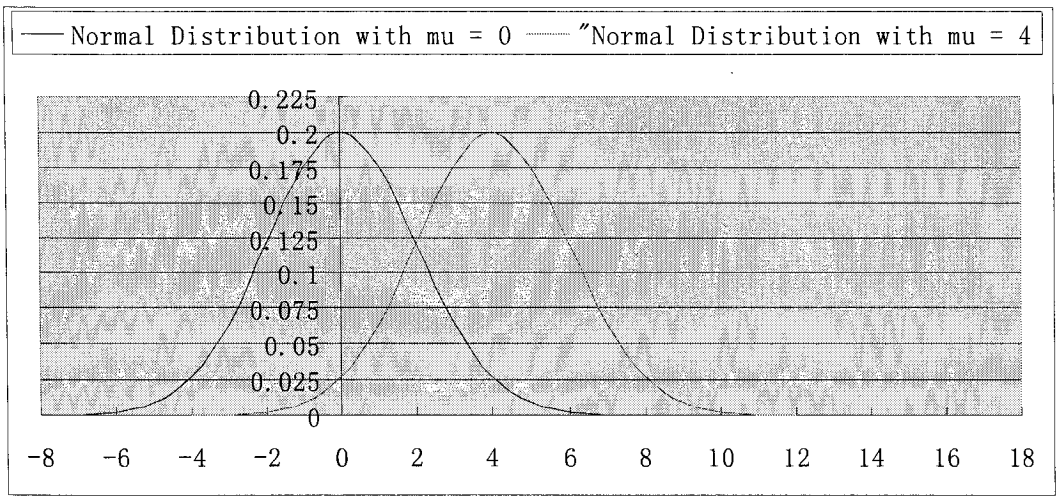

Figure 20 Shift of Expectation (Based on Fig 1 in [9])

Doo Gun Hong pointed out that the statistic value W shifted to the right when an image embeds a message. The intersectional point of two normal distribution curves marks the threshold value as $N\delta$. The detection rule is defined as: A message bit '0' is detected when is less W than $N\delta$, and a message bit '1' is found when the value W is greater than $N\delta$. The intersection part of two curves and horizontal coordinate in Figure 20 shows the possible detection errors, which are the error when watermark is not detected in an image with a watermark embedded and the error while the detection of a non-existing watermark.

With a larger N and δ, the random variable W will shift further away from the original distribution S, which reduces the intersection part of the two distributions. This means lower detection errors and better veracities of extractions. However, N and δ are limited by the image size and perceptibility requirement. Therefore, the appropriate values must be chosen to solve the trade-off.

Based on the above analysis, experiments are conducted on several images using different parameters N and δ. Table 5 lists the experiment performed on a typical image named mattew1.png. This helped the author to decide appropriate values for N and δ, which are

presented in detail in the following part.

Secret Key	No. of Pixels	Brightness Difference	Image Quality
100	100	736	Not affected
	500	-235	Not affected
	1000	2046	Not affected
	2000	-327	Not affected
	5000	3480	Not affected
	10000	2076	Not affected
	20000	-560	Not affected
13579	100	339	Not affected
	500	-1096	Not affected
	1000	-1083	Not affected
	2000	1824	Not affected
	5000	5654	Not affected
	10000	11653	Not affected
	20000	19113	Not affected

Table 5 Number of Pixels and Difference of Luminance Level in the Image Named mattew1.png When Brightness Adjustment Level is 10. (The quality of the image may be affected if the no. of pixels is small, for which no cases are presented here.)

3. Implementation of the revised Patchwork algorithm

Unlike the LSB and brightness modulation algorithms, the Patchwork algorithm chooses a group of pixels to embed one bit message. This attribute determines that fewer message bits can be embedded in an image. Besides, the detection is based on the statistical analysis and tends to have a higher probability of errors. The lower message capacity and higher detection rate of error limit its empirical usage. For example, a slight damage to an image leads to the destruction

of the whole embedded watermark.

The embedding process is designed as shown in Figure 21. First, Hamming checking bits are added to the watermark bits to form the Hamming code bit string. Then, the whole image is divided into small blocks. Each block embeds one bit message. If bit '0' is embedded, no change is made in the block. If bit '1' is embedded, the revised Patchwork calculation is applied to the block: Pixels are randomly chosen by using a private key. The brightness of half pixels is changed to a slightly higher value while the brightness value of another half is decreased to a lower value. This process continues until all bits are embedded.

In the extraction process, the image is divided into small blocks using the same block size as defined in the embedding process. In each block, pixels are randomly chosen by using the same private key as used in the embedding process. The revised Patchwork calculation is conducted: the sum of brightness values of half selected pixels is subtracted from the sum of brightness values of the other half pixels. If the result is greater than a predefined level, a bit '1' message is detected. Otherwise, a bit '0' is found. The process continues until all blocks are examined. Finally, the extracted bits string is decoded to get the embedded information. Figure 22 shows the general idea of the process.

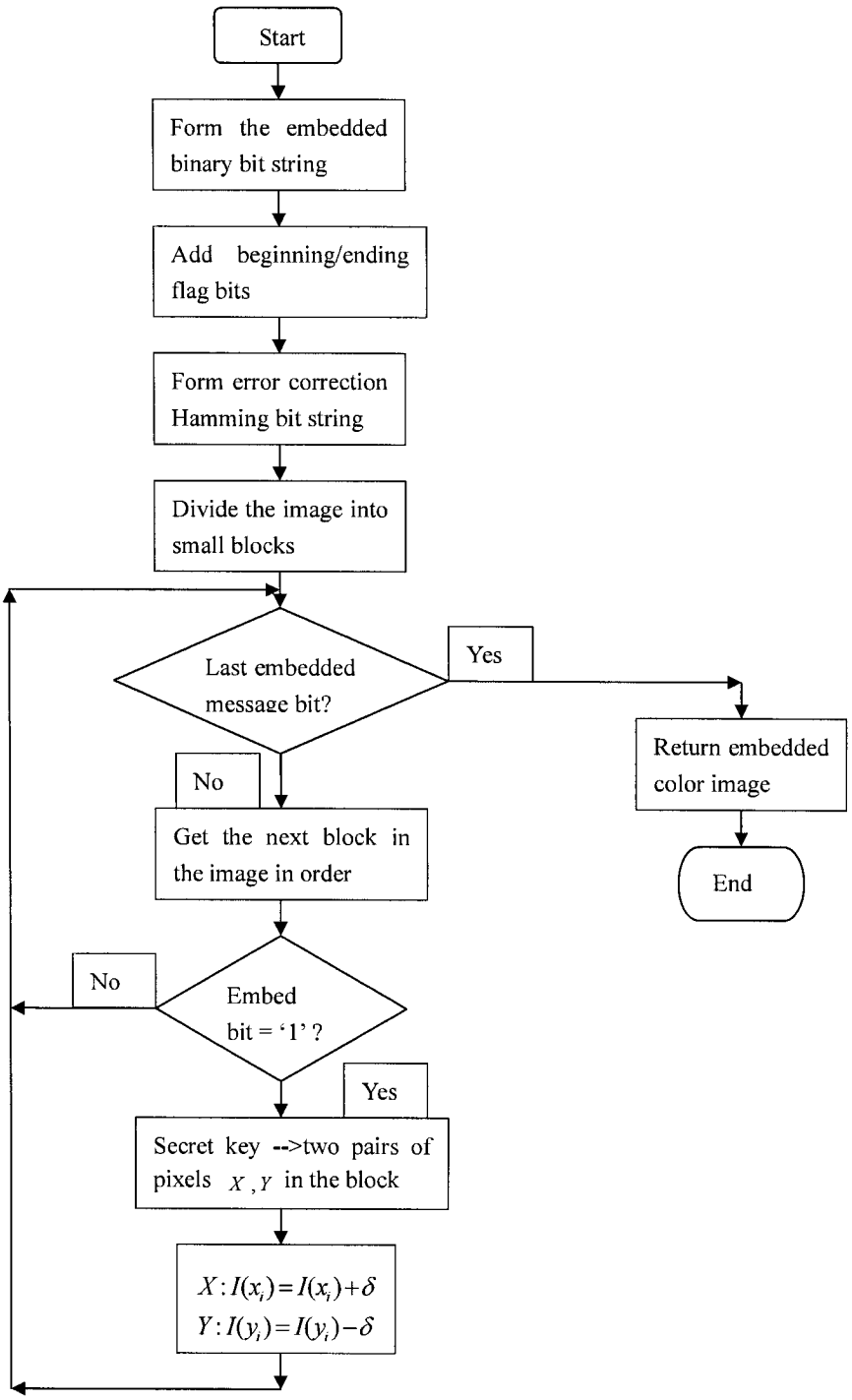

Figure 21 Flow Chart of the Embedding Process for the Revised Patchwork Algorithm

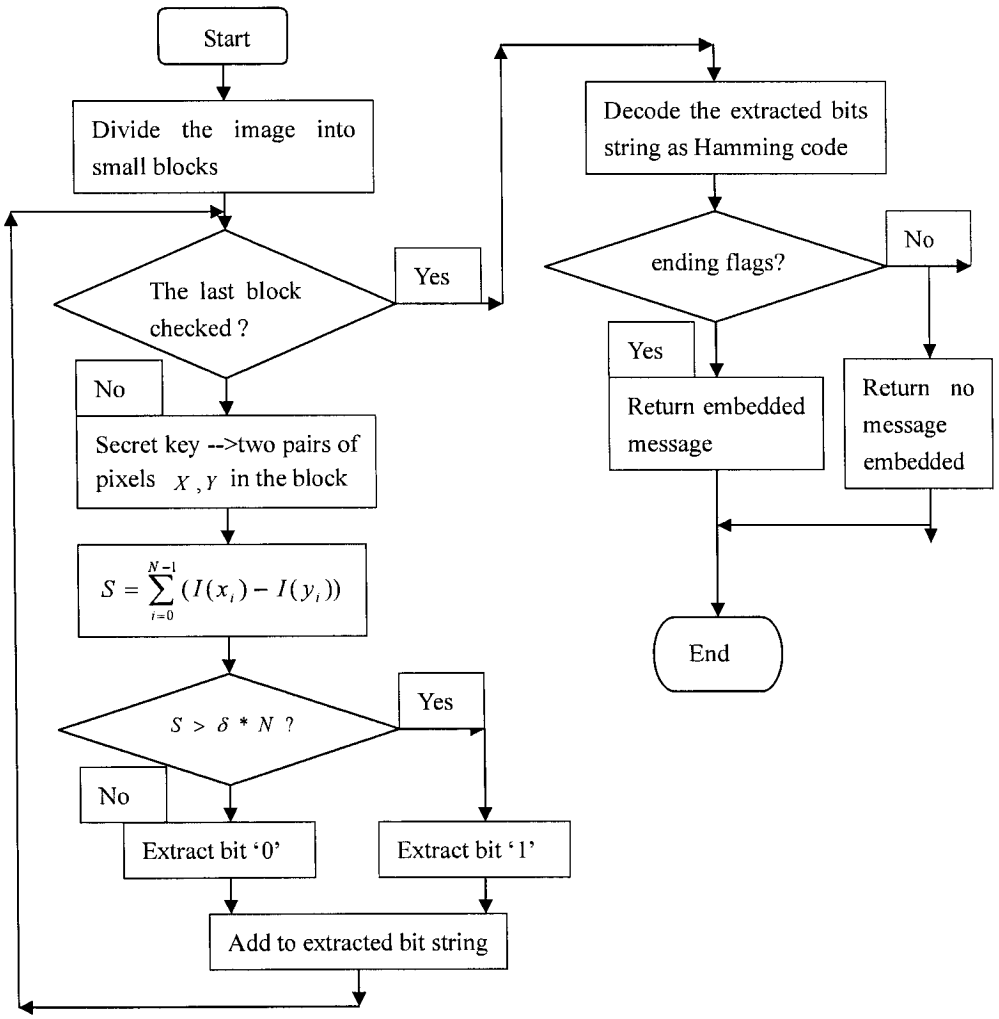

Figure 22 Flow Chart of the Extraction Process for the Revised Patchwork Algorithm

The following presents our efforts to achieve the requirements of imperceptibility, robustness, processing time, and security. We will also discuss parameter settings for the system.

- Imperceptibility

As the human visual system is least sensitive to brightness changes in the blue component of a color image, the above embedding and extraction process for a color image are all performed in

the blue component.

Experiments have show that this algorithm causes no perceptible distortion in images.

- Robustness

Due to limited capacity of the algorithm, message bits can only be embedded once in the image. Therefore, the embedded watermark can not survive if the image is cropped.

In order to make the watermark survive when an image changes from color to gray, the Patchwork calculation is performed in the blue component of a color image or in the gray component of a gray image.

In order to make the watermark survive when an image rotates by 90,180 and 270 degrees, three rotation formulas are used. Different from previous two algorithms, the rotation calculation is performed on the whole image, rather than in a small block of image. Suppose the original image has a width and height expressed as old_width and old_height. The coordinate of a pixel in the original image is (row, col), the corresponding position after rotation is (new_row, new_col).

The formulas are modified as follows:

$$\begin{aligned} new\_row &= col; \\ new\_col &= (old\_height - 1) - row \end{aligned}$$, for 90 degree rotation

$$\begin{aligned} new\_row &= (old\_height - 1) - row; \\ new\_col &= (old\_width - 1) - col \end{aligned}$$, for 180 degree rotation

$$\begin{aligned} new\_row &= (old - width - 1) - col \\ new\_col &= row \end{aligned}$$, for 270 degree rotation

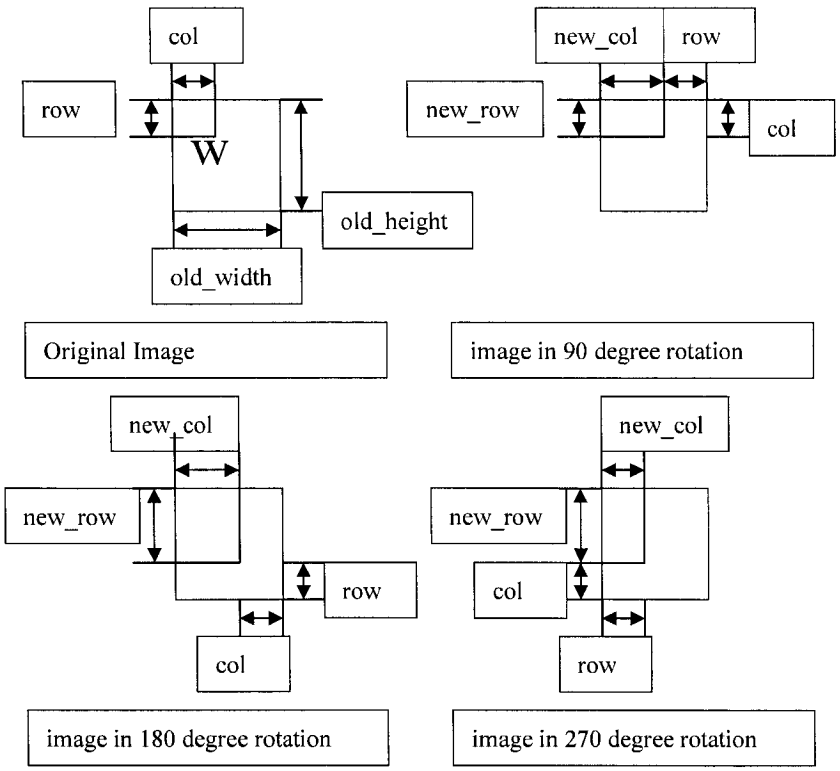

Figure 23 Relationship between an Image and Its 90, 180, and 270 Degree Clockwise Rotations

- Processing time

As message bits are hidden across the whole image without redundancy, the whole image must be checked to get the embedded bit. However, the processing time is fairly short. We shorten the time by checking the existence of the head message in the extraction process.

- Capacity

Patchwork algorithm has low capacity. According to the observation made by the website [5], "watermark messages may not exceed 100 bits (12 characters) for the vast majority of commonly available images on the Internet" [5]. In our system, the capacity depends on the image size. For most images, we can get embedding space more than 10 characters. Hence Patchwork algorithm is still of practical usage.

- Security

A secret key or password is used to randomly choose the locations within each block to hide watermark bits.

- Parameter issues

While implementing the algorithm, we have difficulties and questions such as: How many pixels are chosen to embed a message bit? How many changes are made to the brightness level in order not to cause visual perceptibility? These present the parameter issues.

♦ Number of pixels to embed 1 bit message

The number of pixels should be properly chosen to balance the tradeoff between retrieval errors and capacity. A large number of pixels can improve detection performance and lower detection errors. However, this also means a small number of bits information can be embedded in an image, which makes the algorithm impractical. Empirical experiments have shown the 800 pixels are appropriate to embed 1 bit information.

♦ Block size

Basically, the number of pixels to embed 1 bit message determines how large the block should be. The block size should be larger than $\sqrt{800} \approx 29$ pixels. We chose it as 35 pixels.

♦ Brightness modulation level

The larger change of the brightness level will increase the chance of correctly retrieving the watermarks, but at the same time may cause visual distortion of the original pictures. As discussed in Section 3.1, brightness level ranges from 0 to 255 in standard J2SE 1.4 system. In the implementation system, the brightness change is set to 10 in order to maintain visual imperceptibility.

5.1.4 DFT Amplitude Modulation Algorithm

This and the next sections present two algorithms performed in the frequency domain of an image. DFT is performed on the blue component of an image, "which is the one that the human

eye is least sensitive to "[22]. Then, watermarking is performed on either the amplitude or phase component of the DFT.

The first effort is to try to manipulate the amplitude component directly. Similar to the brightness modulation approach in the space domain, the watermarking algorithm is to hide information by establishing a mathematical relationship between amplitude values of a randomly chosen pixel and its neighboring points in the DFT domain. Figure 24 show a pixel and its four neighboring points.

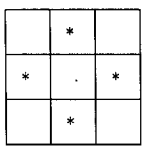

Figure 24 Pixel and Its Neighboring Points in the DFT Domain

Suppose the average amplitude value of these four neighboring points is $\hat{A}_{i,j}$. The Calculation is as follows:

$$\hat{A}_{i,j} = \frac{1}{4}(A_{i-1,j} + A_{i+1,j} + A_{i,j-1} + A_{i,j+1})$$

Where $A_{i,j}$ represents the amplitude value of the point at location (i, j) in the DFT frequency domain.

The embedding process is shown in Figure 25.

If bit "1" is embedded, the amplitude value of a pixel is set to be larger than the average value of its four neighboring points.

If bit "0' is embedded, the amplitude value of the pixel is set to be less than average value of its neighboring points. If the pixel value happens to be much greater than the value of neighboring points, we need to adjust the pixel value smaller and the values of neighboring points larger at the same time.

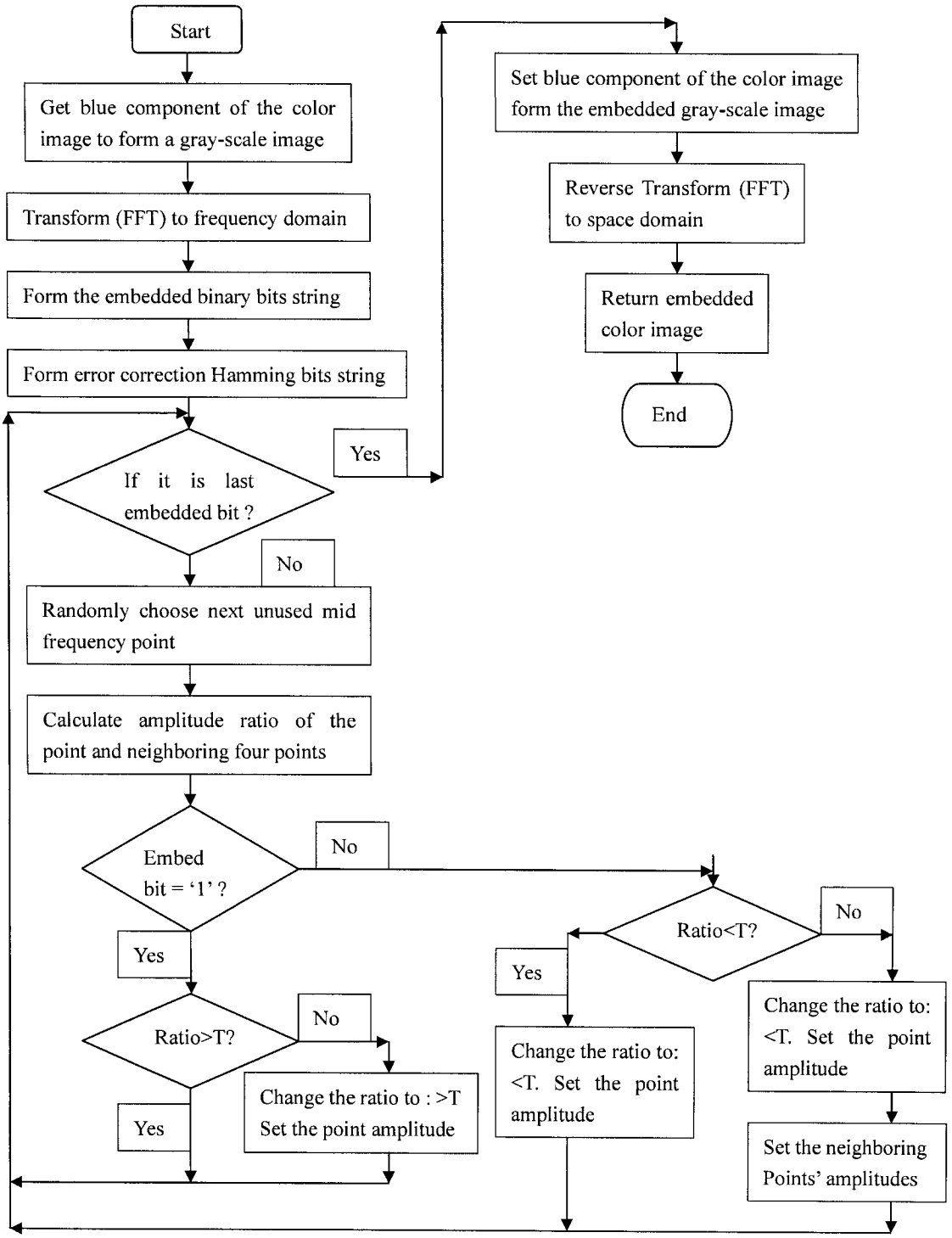

Figure 25 Flow Chart of the Embedding Process for the Amplitude Modulation Algorithm

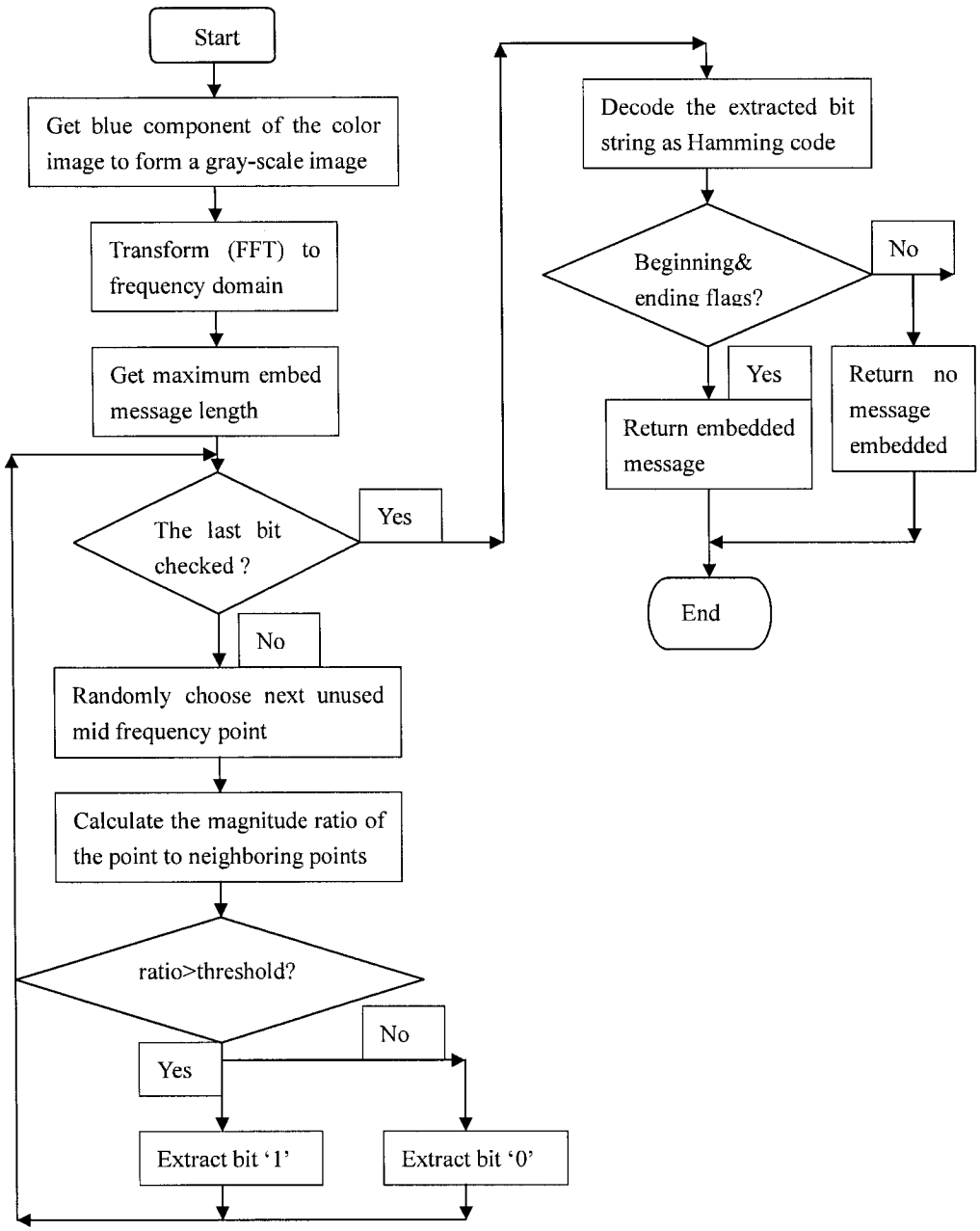

Figure 26 Flow Chart of the Extraction Process for the Amplitude Modulation Algorithm

The retrieval procedure, as shown in Figure 26, is to compare the amplitude value of the central point with the average value of its four neighboring points $A(i,j), and\ \hat{A}(i,j)$. If the former value is greater than the later, bit "1" is judged to be embedded. Otherwise, a bit "0" is retrieved.

- Parameter setting

Parameter setting is an important issue for the algorithm. This is mainly due to the fact that the values set and retrieved for amplitude are not the same using J2SE's image processing package. As an improper parameter setting can lead to greater detection errors, we conducted a number of experiments to choose appropriate values. Table 6 shows the major parameters and procedures for the embedding progress.

Ratio	**Parameters**	**Description**
Embedding bit "1"		
If $A(i,j)/\hat{A}(i,j) >$ DIF_TIMES	DIF_TIMES =1.5	No action taken
If $A(i,j)/\hat{A}(i,j) <$ DIF_TIMES	UPLEVEL=2.2	Adjust $A(i,j)$ up by UPLEVEL
Embedding bit "0"		
If $A(i,j)/\hat{A}(i,j) <$ DIF_TIMES	DOWNLEVEL=8	Adjust $A(i,j)$ down by DOWNLEVEL
If $A(i,j)/\hat{A}(i,j) >$ DIF_TIMES		Adjust $A(i,j)$ down by DOWNLEVEL/5 or DOWNLEVEL/10; Adjust 4 neighboring points up by UPLEVEL.

Table 6 Parameter List for the Amplitude Modulation Algorithm in DFT.

5.1.5 DFT Phase Modulation Algorithm

The second effort in the DFT domain is to manipulate the phase component. The periodicity of the phase component of the DFT leads to conditions on the values the DFT may take. Suppose $F(u,v)$ is the DFT. $A(u,v)$ is the amplitude component and $\phi(u,v)$ is the phase component. Thus,

$$F(u,v) = |F(u,v)| \times e^{j\times\phi(u,v)} = A(u,v) \times e^{j\times\phi(u,v)}$$

If 2π is added to the phase component $\phi(u,v)$, then

$$F'(u,v) = |F(u,v)| \times e^{j\times(\phi(u,v)+2\pi)} = A(u,v) \times e^{j\times\phi(u,v)} \times e^{j\times 2\pi}$$
$$= A(u,v) \times e^{j\times\phi(u,v)} \times (\cos 2\pi + j \times \sin 2\pi) = A(u,v) \times e^{j\times\phi(u,v)} \times (1)$$
$$= A(u,v) \times e^{j\times\phi(u,v)} = F(u,v)$$

Hence, $\phi'(u,v) = \phi(u,v) + 2\pi$

If π is added to the phase component $\phi(u,v)$, then

$$F''(u,v) = |F(u,v)| \times e^{j\times(\phi(u,v)+\pi)} = A(u,v) \times e^{j\times\phi(u,v)} \times e^{j\times\pi}$$
$$= A(u,v) \times e^{j\times\phi(u,v)} \times (\cos(\pi) + j \times \sin(\pi)) = A(u,v) \times e^{j\times\phi(u,v)} \times (-1)$$
$$= -A(u,v) \times e^{j\times\phi(u,v)}$$

Build on Nick Efford's image processing package[11], Empirical results shows that the package supports the phase component $\phi(u,v)$ of the FFT of a gray image is expressed between $-\pi$ and π, that is $-\pi \le \phi(u,v) \le \pi$. If π is added to the phase component $\phi(u,v)$, the amplitude component $A(u,v)$ remains positive value, while the phase component $\phi(u,v)$ change to $\phi(u,v) + \pi$ if $\phi(u,v) \le 0$ or $\phi(u,v) - \pi$ if $\phi(u,v) \ge 0$ in most cases. That is,

$\phi' = \phi(u,v) + \pi \ge 0$ if $\phi(u,v) \le 0$; and

$\phi' = \phi(u,v) - \pi \leq 0$ if $\phi(u,v) \geq 0$.

These relationships imply that we can change the sign of the phase component by adding π to the phase component. Thus, the author deduced the algorithm as follows.

When bit "1" is embedded, the phase value is set to be greater than 0 ($\phi(u,v) > 0$): add π to $\phi(u,v)$ if $\phi(u,v) \leq 0$; leave the $\phi(u,v)$ unchanged if $\phi(u,v) > 0$.

When bit "0" is embedded, the phase value is set to be less than 0 ($\phi(u,v) \leq 0$): add π to $\phi(u,v)$ if $\phi(u,v) \geq 0$; leave the $\phi(u,v)$ unchanged if $\phi(u,v) \leq 0$.

In the extraction process, a bit "1" is extracted if the phase is greater than 0, and a bit "0" is retrieved in other circumstances.

The following Figure 27 and Figure 28 show the detailed embedding and extraction processes.

There are still some boundaries conditions, which for unknown reasons and incomplete information posted for the package, do not comply with these rules. For example, when $\phi(u,v)$ is too small to be nearly 0 or too large to be nearly π, the change of phase does not comply with above rules. These all lead to future work for a better result.

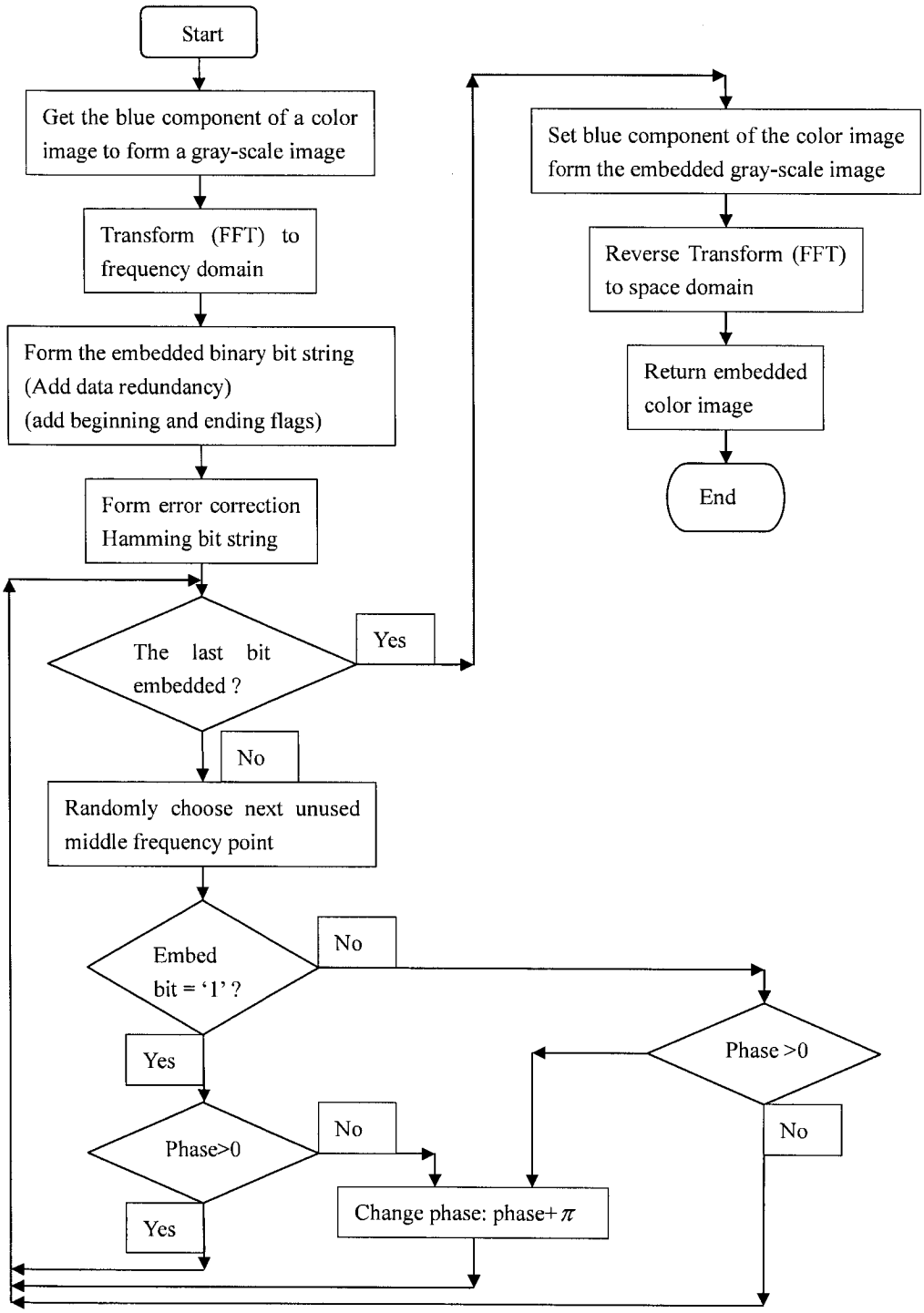

Figure 27 Flow Chart of the Embedding Process for the Phase Modulation

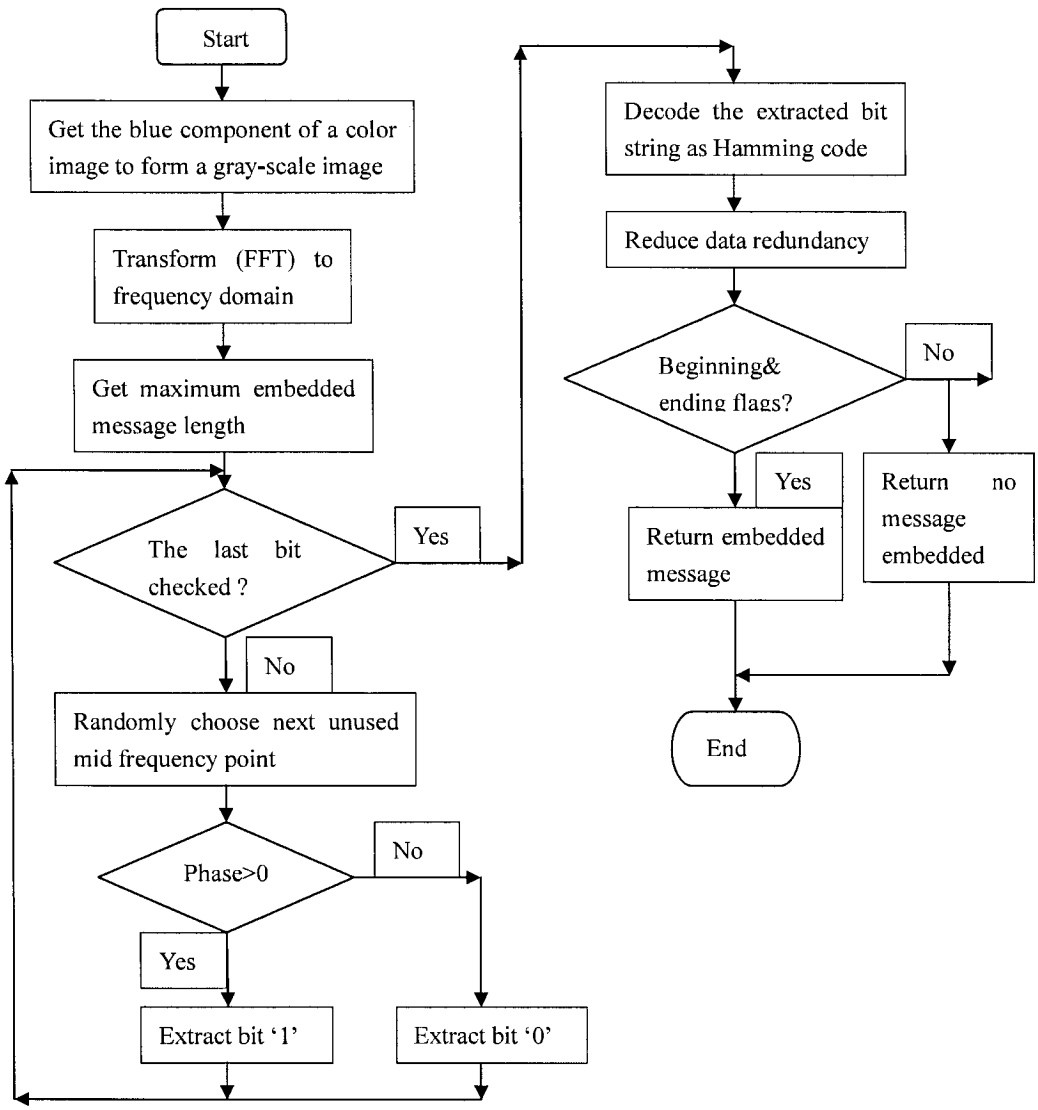

Figure 28 Flow Chart of Extraction Process for the Phase Modulation Algorithm

- Parameter issues

Due to the uncertain change of a few phase value after setting and saving it in an image using Nick Efford's image processing package, same messages are embedded in the image for several

times. Currently the replication number is set to 3. The larger the number is, the lower the retrieval error is. However the higher data redundancy will not only lead to a long processing time for both the embedding and extraction process, but also a poor performance (eg, visual distortion) in a picture with small candidate phases.

The following specifically summarizes our efforts to achieve the requirements of imperceptibility and robustness in the DFT amplitude and phase modulation algorithms.

1. Imperceptibility

Besides manipulating the amplitude and phase on the blue component of an image, we will also consider the following:

- In order to avoid visual distortion, the DFT's negative symmetry rules on the amplitude and phase components must be obeyed in the modulation processes.

  For the amplitude modulation in section 4.4.5,

  $$A'(u,v) = A(u,v) + r$$
  $$A'(N-u,N-v) = A(u,v) + r \quad ,$$

  here $A(u,v)$ is the amplitude component, r is the adjustment level.

  For the phase modulation in Section 4.4.6,

  $$\phi'(u,v) = \phi(u,v) + \delta$$
  $$\phi'(N-u,N-v) = \phi(u,v) - \delta \quad ,$$

  here $\phi(u,v)$ is the phase component, δ is the adjustment level.

- By studying the amplitude spectrum of the blue component of an image, the author found that an image has its energy centralized on the origin and two axes. Changes made to these points will lead to greater perceptible distortion in the original image. Therefore, we exclude these points from the embedding process.

2. Robustness

In order to make the watermark survive when an image is cropped, the watermarking of both the amplitude and phase modulations are performed in the middle frequency part of the spectrum, which contains most important information about the image. The watermark can survive if the image is slightly cropped.

In order to make the watermark survive when an image changes from color to gray, the two algorithms are performed in the blue component of a color image or in the gray component of a gray image.

In order to make the watermark survive when an image rotates by 90,180 and 270 degrees, three rotation formulas are used. These formulas are the same as those used in the Patchwork algorithm in Section 4.4.3. The rotation is performed on the whole image, rather than in a small block of image.

5.2 Evaluation of the Algorithms

This subsection briefly presents a trade-off analysis, and discussion of attack strategies and performance.

5.2.1 Trade-off Analysis

In the analysis and implementation of each algorithm, we had been focusing on achieving the basic properties of watermarks. Figure 29 illustrates the four basic requirements. However, imperceptibility, robustness, reliability and capacity may not be achieved at the same time. In our system, algorithms were designed based on the trade-off principle. Imperceptibility is the first priority. An algorithm which cannot achieve imperceptibility is not a good candidate for usage. We also intended to improve the reliability and lower the error rate of detection by using Hamming codes or by embedding a watermark message several times. Some efforts were made to achieve the robustness of the watermark and they are described in the next subsection 5.2.2.

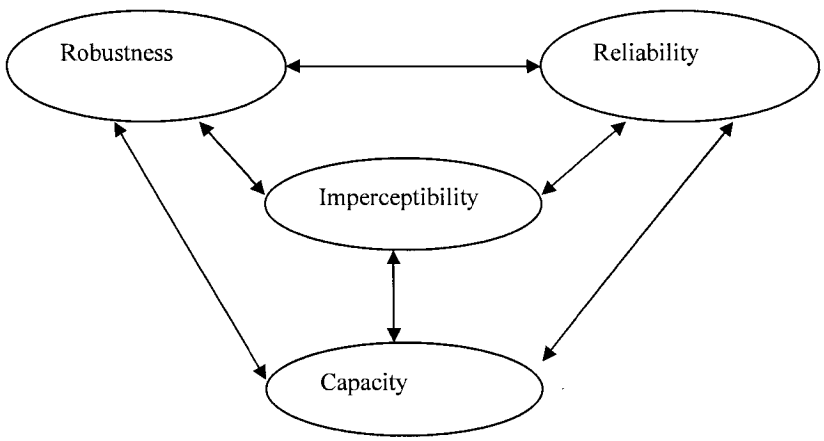

Figure 29 Trade-off Analysis

5.2.2 Discussion of Attack Strategies

As stated in the previous subsection, robustness is a requirement for a watermarking. Theoretically, a good watermarking scheme requires the embedded watermark can be recovered if the image is distorted. These attacks are classified into two categories: either performed in the space domain (eg, cropping, scaling and rotation) or in the frequency domain (eg, low pass/high pass filtering). As our focus is on providing the framework for watermarking algorithms, testing of algorithms, we did not take an in-depth investigating of any particular attack-resistance methods. Rather, we demonstrate the ideas we deployed in our system.

To make the watermark survive a cropping attack, we divide an image into small blocks for algorithms performed in the space domain. Each block embeds the same watermarking message.

To make the watermark survive rotations in 90, 180 and 270 degrees, we design rotations formulas for all algorithms.

To make the watermark survive converting form a color to gray image, we embed the

watermark bits in the three color components of a pixel for a color image.

Table 7 summarizes the attack-resistance results

	LSB	**Brightness Modulation**	**Patchwork**	**Amplitude Modulation**	**Phase Modulation**
Space Domain					
Cropping or Image Partially Damaged	Yes	Yes	No	Depends	Depends
Rotation in Four Degrees	Yes	Yes	Yes	Yes	Yes
Color to Gray	Yes	Yes	Yes	Yes	Yes

Table 7 Analysis of Several Attacks in the Space Domain

5.2.3 Experimental Results

The following pictures show the original and the watermarked image using LSB, Patchwork and DFT phase modulation algorithms. They are used as samples to illustrate how the watermarks survive cropping, rotation and color to gray transforms.

The original image is shown in Figure 30. The embedded message is "my little girl" and the password is "111111". Figure 31, Figure 32, Figure 33, Figure 34 and Figure 35 demonstrate the extracted images and messages using the LSB method. Figure 31 shows that we can extract the watermarked message "my little girl" using this method. Figure 32 and Figure 33 exhibit that we can also extract the embedded message after cropping part of the watermarked image or rotating the watermarked image. Figure 34 and Figure 35 shows that the watermarked message can be detected even if the image is transformed to a gray image and cropping is performed in the transformed gray image.

Figure 30 Original Image

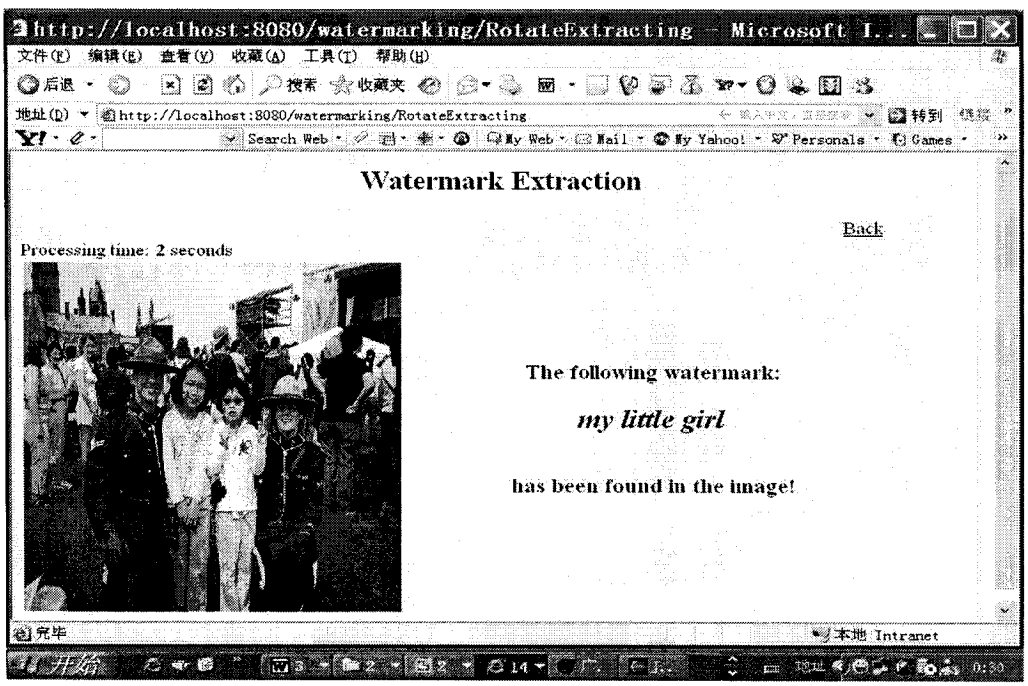

Figure 31 Extraction Process of the LSB Method

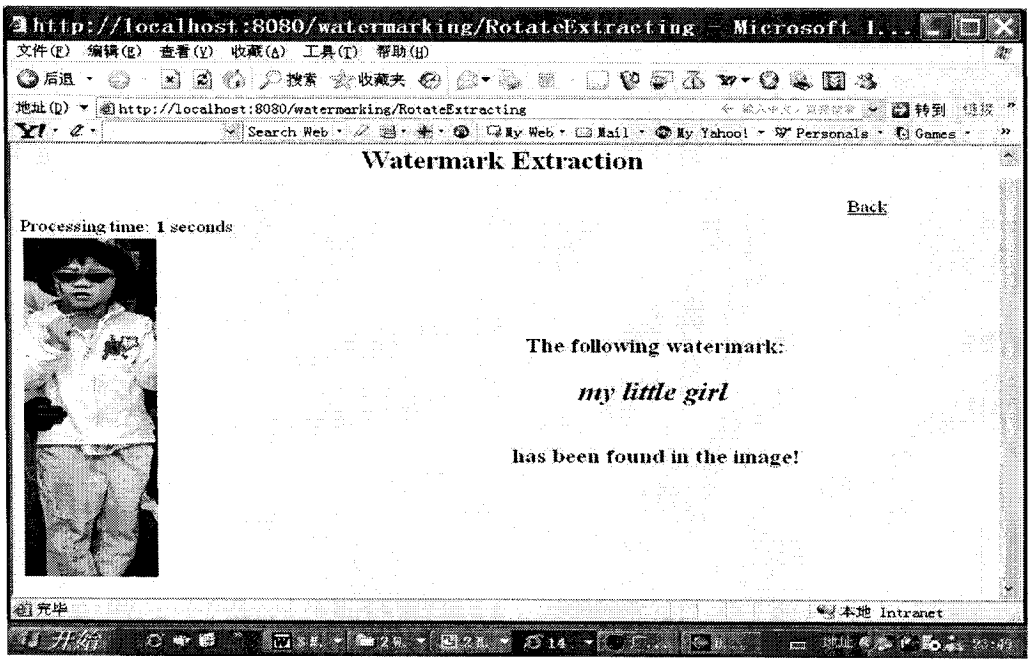

Figure 32 Extraction Process of the LSB Method--Cropping

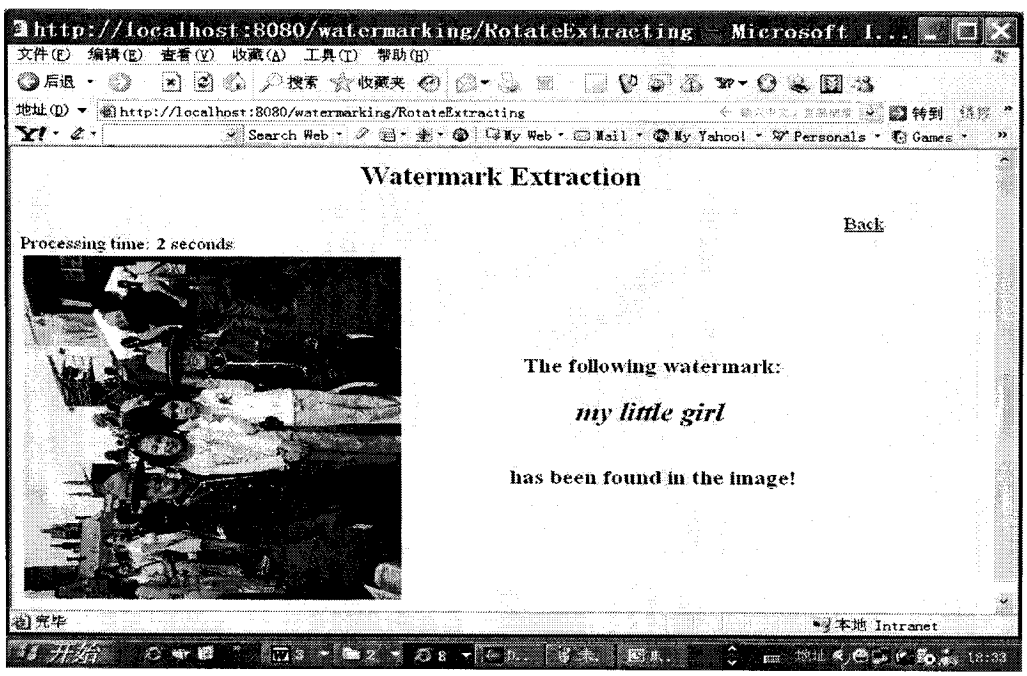

Figure 33 Extraction Process of the LSB Method -- Rotation

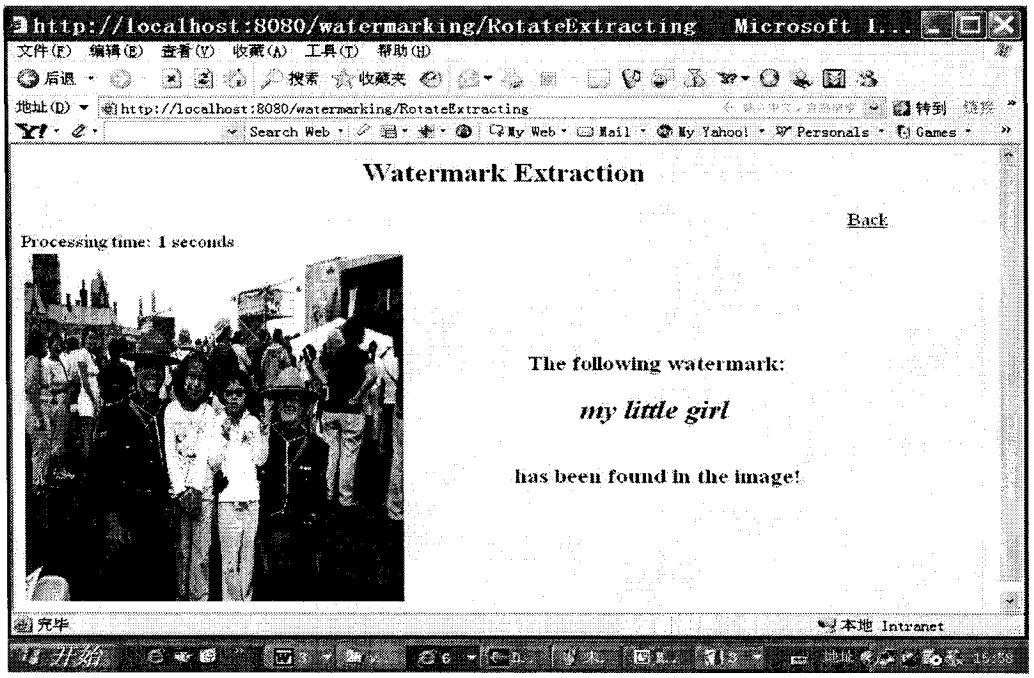

Figure 34 Extraction Process of the LSB method – Color to Gray Transform

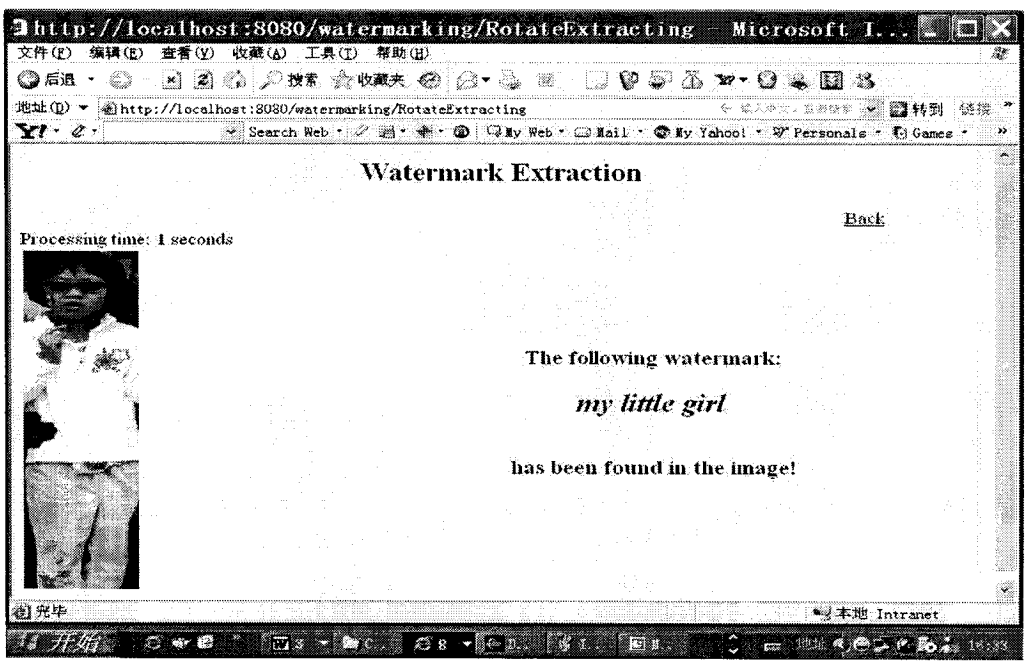

Figure 35 Extraction Process of the LSB method – Color to Gray Transform & Cropping

Figure 36 illustrates the extraction result using the revised patchwork algorithm. We can notice that the extracted watermarked message is "my nittle girl" instead of "my little girl". This is due to the statistical error based on the algorithm itself (Analyzed in detail in Chapter 5.1.3), despite that we have used Hamming code and added data redundancy in the embedded messages.

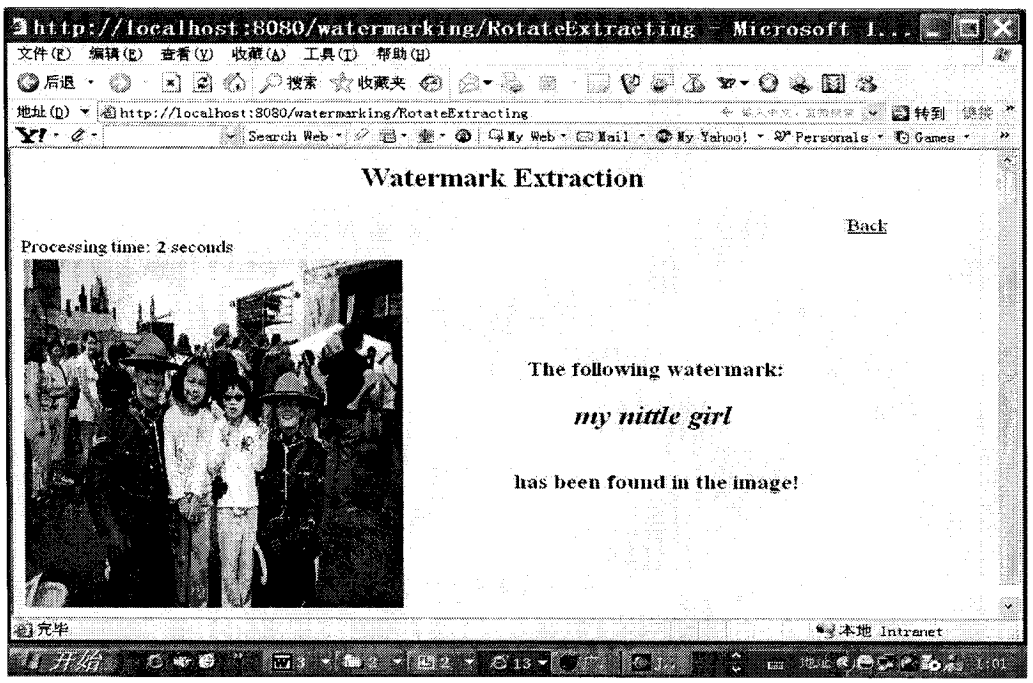

Figure 36 Extraction Process of the Revised Patchwork Algorithm

Figure 37, Figure 38, Figure 39 and Figure 40 illustrate the extracted images and messages using the DFT phase modulation method. Figure 37 shows that we can extract the watermarked message "my little girl" using this method. Figure 38 and Figure 39 exhibit that we can still extract the embedded message after subtracting part of the watermarked image which is not "important" in the image. However, Figure 40 shows that the watermarked message could not be detected if the image is subtracted "too much", as too much information is lost in the subtracted part.

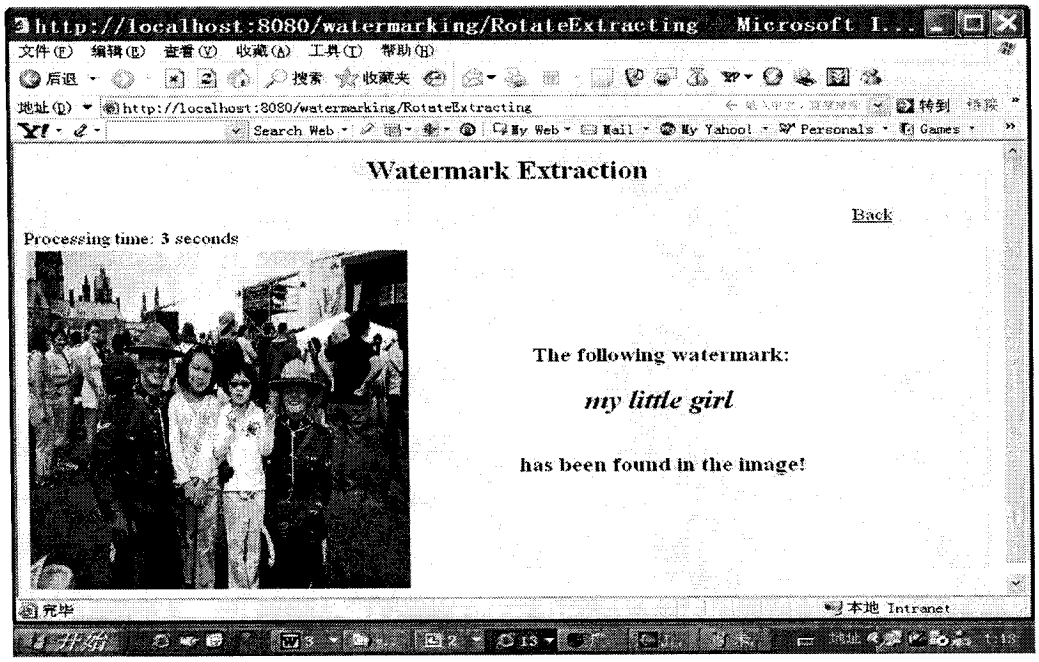

Figure 37 Extraction Process of the DFT Phase Modulation Method

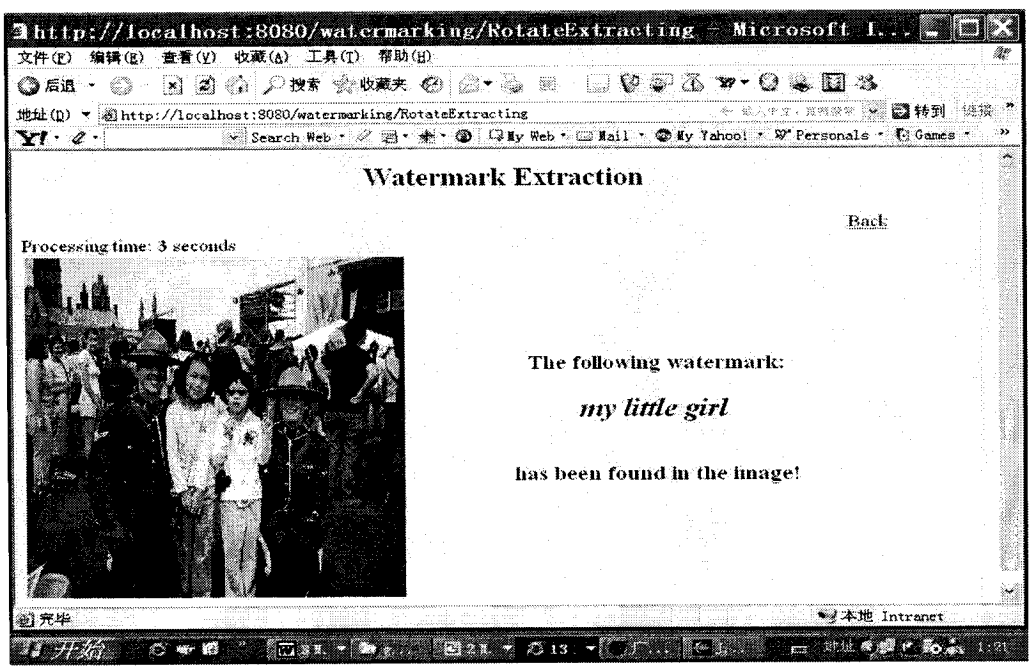

Figure 38 Extraction Process of the DFT Phase Modulation Method—Subtracting (1)

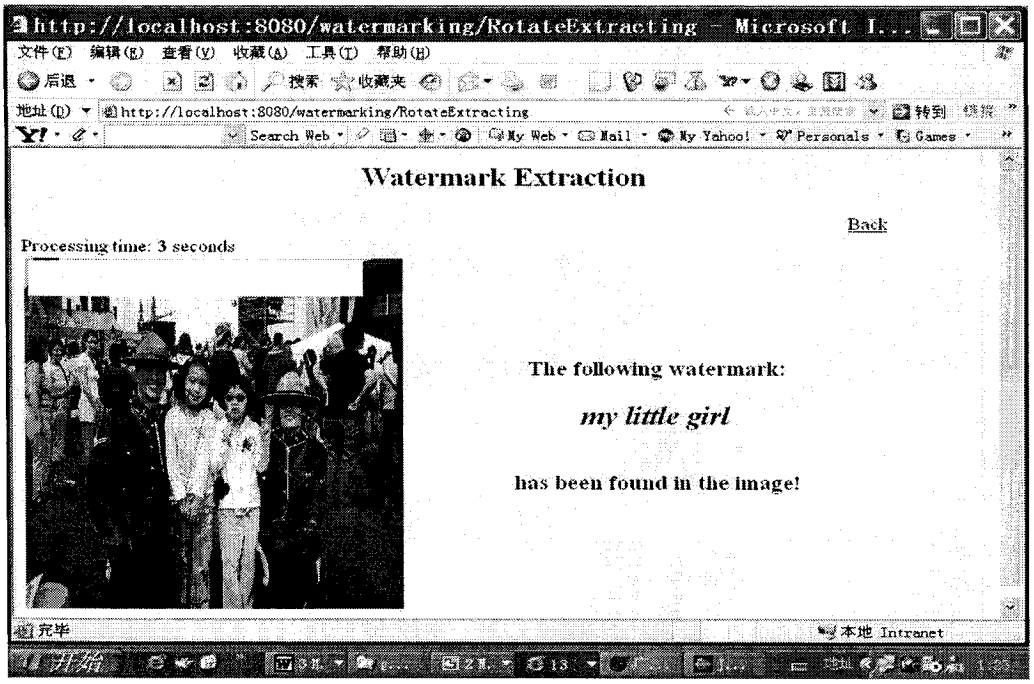

Figure 39 Extraction Process of the DFT Phase Modulation Method—Subtracting (2)

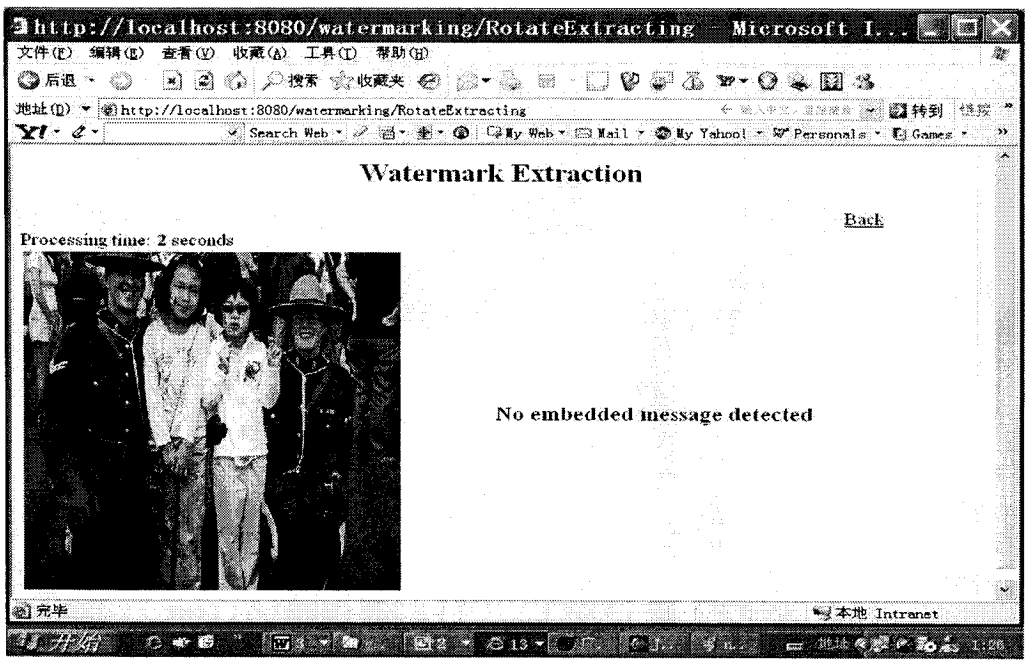

Figure 40 Extraction Process of the DFT Phase Modulation Method—Subtracting (3)

In summary, our implemented system acts as a manager to "show" rather than "tell" the story of digital image watermarking techniques. It visualizes the watermarking processes and makes some obscure terminologies easy to understand.

5.3 Issues in the System Infrastructure

The implementation of the digital watermarking system demands multi-disciplinary knowledge. It involves in-depth analysis of watermarking algorithms, deploying Web technology, exploring open source resources and testing feasible parameters. A few key technical aspects are summarized as follows

1. Code Reuse

"There's no sense in reinventing the wheel."[40] This is a common saying in the Java world. Code reuse has already formed a "reuse culture" prevailing in the technology world. Benefits from code reuse include reducing cost, increasing developers' productivities and improving product quality. It is especially beneficial for FLOSS developers, since it is a goal for participants to share their products with others.

Either available entirely or partially, reusable codes can be employed via libraries, packages or subroutines. The author has adapted a few Java source codes from open source resources under various licenses which allow redistribution for non-commercial usage. The following lists the origins of the software and packages used.

- J2SE API

J2SE API provides a bunch of powerful Java class libraries and packages that greatly benefits the development. For example, java.util.Random, a Java class, supports implementing the private password mechanism in the system. The extensive java package javax.imageio supports more image encoder/decoder operations than the standard package java.awt.image.

- Uploading and downloading information

The file upload package com.oreilly.servlet.multipart [40] supports uploading files from the local

browser to the application server. The source code is available under a non-commercial License, with commercial redistribution rights available [40]. Other software almost certainly exists that is available under more general licenses. The users can download images by simply right-clicking over the image and then selecting the "Save Image as" function. Javascript in the web page could also be used to provide this functionality in a button.

- Image processing package by Nick Efford [11]

We need the FFT tools to perform operations in the frequency domain. We found that image processing package by Nick Efford provided several classes to manipulate images and we are using its FFT class to support the DFT amplitude and phase algorithms. Unfortunately, this package is not open source software and is only free for educational reference. If we want to register our system as open source software under a generally usable license, we need to find an open source replacement or write a Java class by ourselves.

2. Open Source Tools

The following Table 8 lists the Open Source products and their licenses used in the implementation system.

Software	License	Remarks	Reference
J2SE 1.4.2	Binary Code License Agreement (BCL)	Sun grants you a non-exclusive, non-transferable, limited license without fees to reproduce and distribute the Software, provided that you distribute the Software complete and unmodified	http://java.sun.com/j2se/downloads/
	Sun Community Source Licensing (SCSL)	powerful combination of the best of the proprietary licensing and the more contemporary open source technology licensing models	http://www.sun.com/software/communitysource/j2se/java2/download.xml
Eclipse 3.0	Common Public License version 1.0; Eclipse Public License –v1.0	Acknowledged by OSI	http://www.eclipse.org/org/index.html
Lomboz	GNU LGPL license, Version 2.1	Acknowledged by OSI	http://www.objectlearn.com/legal/license.jsp
Apache Jakarta Tomcat server (embedding J2EE)	Apache Software License (Apache License, Version 2.0)	Acknowledged by OSI	http://www.apache.org/licenses/
com.oreilly.servlet.* packages	A non-commercial license (copyright and owned by Jason Hunter.)		http://www.servlets.com/cos/license.html

Table 8 System Components and Their Licenses

OSI--Open Source Initiative

3. OO Design—Object-oriented Design

The embedding and extraction algorithms of the individual example watermarking algorithms are designed as Java classes. Whenever an algorithm is accessed, the corresponding object is instantiated. This method encapsulates the watermarking algorithm in one object and isolates watermarking technology from the rest of the system framework.

4. Session Management

Session is an advanced design for web applications. As HTTP is a "stateless" protocol, each time a browser sends a request to the Web server, the Web server does not automatically maintains the contextual information about the request. Session in servlets can associate a browser request with a unique Session object supported by the Web container, creating and storing unique session information about a request. Sessions make sure

- A user visiting a Web application in many requests can maintain data integrity. Data stored in the session object can be accessed by these requests from the same browser.
- Various users accessing the web application can have their own unique instance reference and session object (session id for example) to prevent data reentry.

5.4 System Features and Limitations

The implemented system has a few unique features, and also some limitations.

1. System Features

- Oblivious watermarking

No original image is needed to be stored and needed to be used in the extraction process.

- Support for both gray and color images

Unlike majority systems that only support gray-scale images, the implemented system supports both gray-scale and color images.

- Support for multiple image formats

The system is capable of embedding watermarks in images of JPEG and PNG formats and

generating embedded images in the PNG format.

- System portability

As Java is portable, the system can be ported to other platforms which run J2SE and J2EE.

- System extendibility

As the system uses the J2SE technology and object-oriented design, other workers should be able to follow the code structure and program flow to implement other methods that can be added to the system.

- System processing capacity

Due to the processing capacity of the Apache Jakarta Tomcat server under the default setting, the maximum image size appears to be limited to 1024*728 pixels. The watermarking system will break down and show an "out of memory" error when processing an image larger than the size. In order to make the system capable of processing large images, we need to change the default setting. In Windows XP, the setting is as follows:

In Menu Start→Apache Tomcat 5.0→Configure Tomcat→Java, set

Initial Memory Pool: 256 MB

Maximum Memory Pool: 256 MB

2. System Limitations

- Limited watermarking methods support

The current system supports five algorithms. We hope we can have more and robust algorithms added in the system.

- Limited image formats and watermarked image formats supports

The current system support input images in JPEG and PNG formats and output images in PNG formats. More formats can be supported if we find the appropriate input/output filters.

5.5 Summary

In summary, our research involves multi-disciplinary knowledge and efforts. The time and effort involved are required by the need to explore digital image watermarking techniques, digital

image processing, information hiding, computer science and cryptography. This demanding characteristic may suggest a difficult task for us to build the system.

- Specific concerns are the need for knowledge of digital image processing as a pre-requisite for understanding of digital watermarking technology and inventing of new algorithms.
- Recognition, from the review of many academic papers, research results and websites, suggests that there are many challenges in the research of digital image watermarking technology. There exist a lack of protocols, standards, benchmarking and comprehensive mathematical theory and no watermark can survive all attacks, etc. Thus there are still many opportunities for further exploration and research.
- The need, from the perspective of the computer science technology, is to deal with Web technology. We needed to design the Web application structure, to choose from Web servers and application servers. We needed to learn Java, JavaScript, HTML, XML, J2SE technology, Servlets and JSPs of the J2EE technology.
- Information hiding and cryptography technologies also contribute to the building of a watermarking system. Certain knowledge in error coding is also touched upon in the implementation.

Overall, we built a working solution that provides a prototype open source watermarking testbed system that implements several example watermarking algorithms. Our system design has made it simply to extend the system and integrate more watermarking methods and more features such as more image formats. Our research goal is not the implementation of new watermarking algorithms themselves, but the creation of a tool for exploring feasible models for setting up an extendable integration system using Free Software tools. Moreover, I was able to try out a few ideas, for example, the DFT amplitude and phase modulation methods, which are my own attempts at a digital watermarking method.

Chapter 6 Conclusions, Open Questions and Future Work

Future work includes adding more practical watermarking algorithms, efforts to render algorithms more robust and the addition of some management functions to the system. We want to make our system available and usable for others to be able to share in the future activities. We have already commenced the process of building the SourceForge project for this system which tentatively has the name "WaterBed". This will be the public repository of the codes and documentation. They will be available under the GNU General Public License.

As the digital watermarking system and technology evolves, we want to add in more algorithms such as the spread spectrum, Discrete Cosine Transform (DCT) and Mellin-Fourier Transform algorithms. Furthermore, we hope to improve the robustness of the algorithms, such as making the algorithm resist geometric transforms in more degrees. We also hope to support multi-language versions by adapting Unicode.

One particular consideration is to integrate database management in the system so that valuable and manageable information can be collected. This will improve system efficiency and communication in the FLOSS community.

In conclusion, the current research work and the prototype watermarking system have successfully established a basis and framework for me and others to explore ideas in this field.

References

Web references, except where noted, were verified by visiting them on June 10th.

[1] Alpvision company website: *www.alpvision.com/signit/signit_rates.html*

[2] Apache Software Foundation Home Page: *www.apache.org*

[3] Baxes, Gregory A., *Digital Image Processing principles and applications* New York: Wiley & Sons, Inc, 1994

[4] BEA Company Home page: *www.bea.com*

[5] Blue Spike company Home page: *www.bluespike.com*

[6] Borland Software Corporation Home page: *www.borland.com*

[7] Digital WatermarkingWorld Forum: *www.watermarkingworld.org*

[8] Digimarc Corporation Home page: *www.digimarc.com*

[9] Doo, Gun Hong.,et al. "A public key audio watermarking using patchwork algorithm",

http://www.kmutt.ac.th/itc2002/CD/pdf/17_07_45/WA2_PI/7.pdf

[10] Eclipse Foundation Home page: *www.eclipse.org*

[11] Efford, Nick., *Digital Image Processing a practical introduction using Java™* , Harrow, England: Pearson Education Limited, 2000

[12] Free Source Foundation Home page: *www.fsf.org*

[13] Fridrich, Jessica (Ed.), *Information Hiding 6th International Workshop, IH2004 Toronto, Canada, May 2004, Revised Selected Papers*, New York: Springer, 200

[14] Hall, Marty, *Core Servlets and JavaServer Pages* (electronic version)

[15] Hall,Marty, *more Servlets and JavaServer Pages*, USA: Sun Microsystems Press, 2002

[16] Harris,Lesley Ellen., *Canadian copyright law,* 3rd ed., Canada: McGraw-Hill Ryerson Limited, 2001

[17] IBM Corporate Home page: *www.ibm.com*

[18] JBoss Developer Zone website: *www.jboss.org*

[19] Johnson, Neil F., Zoran Duric and Sushil Jajodia. *Information Hiding steganography and watermarking- attacks and countermeasures,* Massachusetts: Kluwer Academic Publishers, 2001

[20] Katzenbeisser, Stefan., Fabien A.P.Petitcolas et al. *Information hiding techniques for steganography and digital watermarking,* Massachusetts: Artech House, Inc, 2000

[21] Kim, Hyung Cook, Ogunleye Hakeem, Guitart Oriol and Delp, Edward J., *The Watermark Evaluation Testbed (WET), www.datahiding.com/spie2004wet.pdf*

[22] Kutter,Martin, et al. "*Digital Signature of Color Images using Amplitude Modulation*", Proc. of SPIEEI97, Feb. 1997, pp. 518-526, *http://citeseer.ist.psu.edu/142846.html*

[23] Linux Home page: *www.linux.org*

[24] Microsoft Corporation Home page: *www.microsoft.com*

[25] Moskowitz, Ira S. (Ed.), "Information Hiding 4th International Workshop, IH2001 Pittsburgh, PA,USA, PA, USA, April 2001 Proceedings", New York: Springer, 2001

[26] MyEclipse Home page: *www.myeclipseide.com*

[27] MYSQL Home page: *www.mysql.com*

[28] Netbeans Organization Home page: *www.netbeans.org*

[29] NEC Research Institute CiteSeer Home page: *citeseer.nj.nec.com*

[30] Onur Mutlu Project report "*An Overview of Image Watermarking Algorithms*",2001

[31] ObjectLearn Home page: *www.objectlearn.com*

[32] Open Source Initiative Organization website: *www.opensource.org*

[33] Perl Foundation website: *www.perl.org*

[34] Petitcolas, Fabien.A.P. (Ed.), *Information Hiding 5th International Workshop, IH2002 Noordwijkerhout, The Netherlands, October 7-9, 2002 Revised Papers*, New York: Springer, 2003

[35] PNG Home Page: *libpng.org/pub/png*

[36] *O'* Ruanaidh, J. J. K., Dowling, W. J., and Boland F. M., "*Watermarking digital images for copyright protection*" in Proc. Inst. Elect. Eng. Conf. Vision, Image and Signal Processing, vol. 143, Aug. 1996, pp. 250--256, *http://citeseer.ist.psu.edu/ruanaidh96watermarking.html*

[37] *O'* Ruanaidh,J.J.K., Dowling,W.J. and Boland, F.M., *Phase Watermarking of Digital Images,* in: Proc. Int. Conf. on Image Processing, Vol. 3, Lausanne, Switzerland, 1996, pp. 239--242. Oct 18, 2003 *http://citeseer.ist.psu.edu/ruanaidh96phase.html*

[38] Schneier, Bruce, *Applied Cryptography Second Edition : protocols, algorithms, and source code in C"*, New York: John Wiley & Sons, Inc,1996.

[39] ServerWatch Home page: *www.serverwatch.com*

[40] Sevlets.com website (of Jason Hunter) *www.servlets.com*

[41] Signum Technologies Home page: *www.signumtech.com*

[42] Source forge website: *www.sourceforge.net*

[43] Sun Microsytems Home page: *www.sun.com*

[44] Sysdeo Home page: *www.sysdeo.com*

[45] USENIX, the Advanced Computing Systems Association Home page: *www.usenix.org*

[46] Walker, James S., *Fast Fourier Transforms,* 2nd ed., Florida: CRC Press Inc., 1996

[47] Wang, Sudan., *Thesis-- Issues in adding special purpose functions to scientific or statistical Free Software*, 2001

[48] Wagner, Neal R., *The Laws of Cryptography with Java Code*, 2003 (electronic version)

[49] Wayner, Peter., *Digital copyright protection,* London: AP Professional, 1997

[50] Wayner Peter. *Free for All How LINUX and the Free Software movement undercut the high-tech titans*, by HaperCollins (free electronic version).

[51] W3C World Wide Web Consortium Home page: *www.w3.org*

Appendices

A. Configuration and Installation of the Production Environment in Windows XP

1 Downloads

Almost all websites of Open Source software provide various builds and versions. Usually we can find release builds, milestone builds, nightly builds and demo builds

- Eclipse

  From www.eclipse.org [10], download eclipse release build versions: eclipse-SDK-3.0-win32.

- Java 2 SDK

  In order to develop, compile and run java applications in Eclipse, a Java SDK 1.4 or above is required. It is likely for Eclipse to experience problems if only a JRE is downloaded and installed. From www.sun.com [43], download Java 2 SDK: j2sdk-1_4_2_05-windows-i586-p.exe

- Lomboz 3.0

  From www.objectlearn.com [31], download Lomboz 3 plugins for Eclipse: emf-sdo-runtime-2.0.0.zip & lomboz.301.zip

- Apache Tomcat server 5.0

  From http://jakarta.apache.org [2], download Apache Tomcat server 5.0 release: Jakarta-tomcat-5.0.27.zip

2 Installation and Configuration

- Eclipse installation

  Extract downloaded eclipse build to directory D:\eclipse in Windows. The system will automatically set ECLIPSE_HOME environment variable to this value.

- Java SDK installation

  Install J2SDK on C:\j2sdk1.4.2_05. Verify if it is installed correctly: a directory with the following contents should exist. (See Figure A. 1)

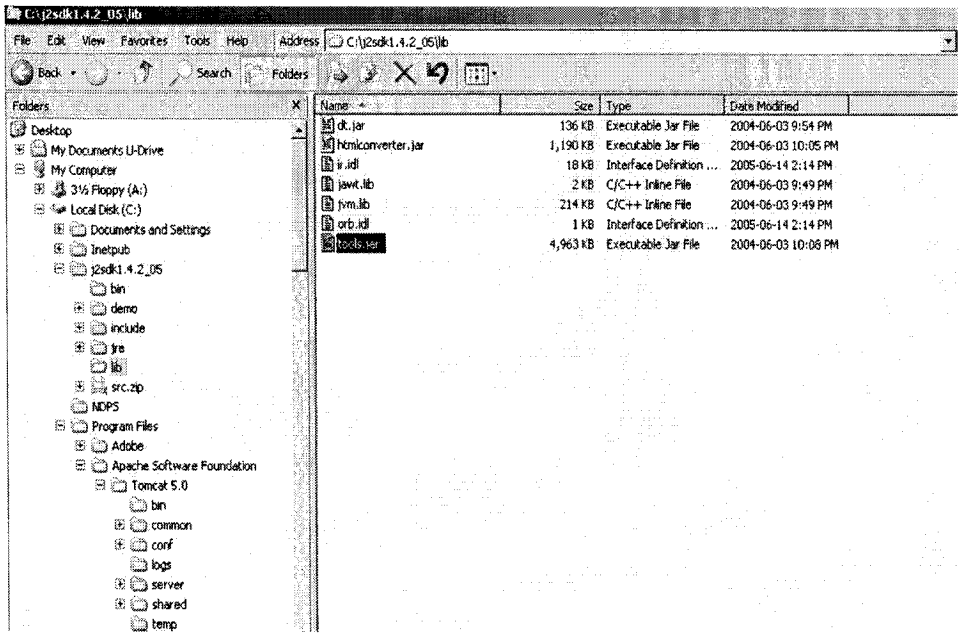

Figure A. 1 Java SDK Installation

- Apache tomcat server installation

  Extract download Jakarta-tomcat-5.0.27.zip to Jakarta-tomcat-5.0.27.exe. Run and install the server in directory D:\jakarta-tomcat-5.0.27. Notice Java installation path is required in the process. Input C:\j2sdk1.4.2_05 as the path, rather than C:\j2sdk1.4.2_05\bin.

- Java in Eclipse

  Eclipse needs proper Java preferences so that it can access Java 2 SDK rather than JRE. After the installation of Eclipse and Java 2 SDK, start Eclipse, visit Java preference and verify if it is using Java 2 SDK. To do this, on the Eclipse menu bar, select Window-> Preference->Java->Build path->Classpath, Check if:

  JDK_TOOLS=C:\j2sdk1.4.2_05\lib\tools.jar.

- Lomboz installation

  Extract emf-sdo-runtime-2.0.0.zip into ECLIPSE_HOME = D:\eclipse directory. Then

extract lomboz.301.zip into D:\eclipse directory.

- Lomboz plugin activation

  Lomboz currently adds actions, wizards and views to the Eclipse environments. To activate these add-ons, start Eclipse, on Eclipse menu bar

  - Select Window->Customize Perspective->Shortcuts->New. Check Lomboz J2EE Wizards. (See Figure A. 2)

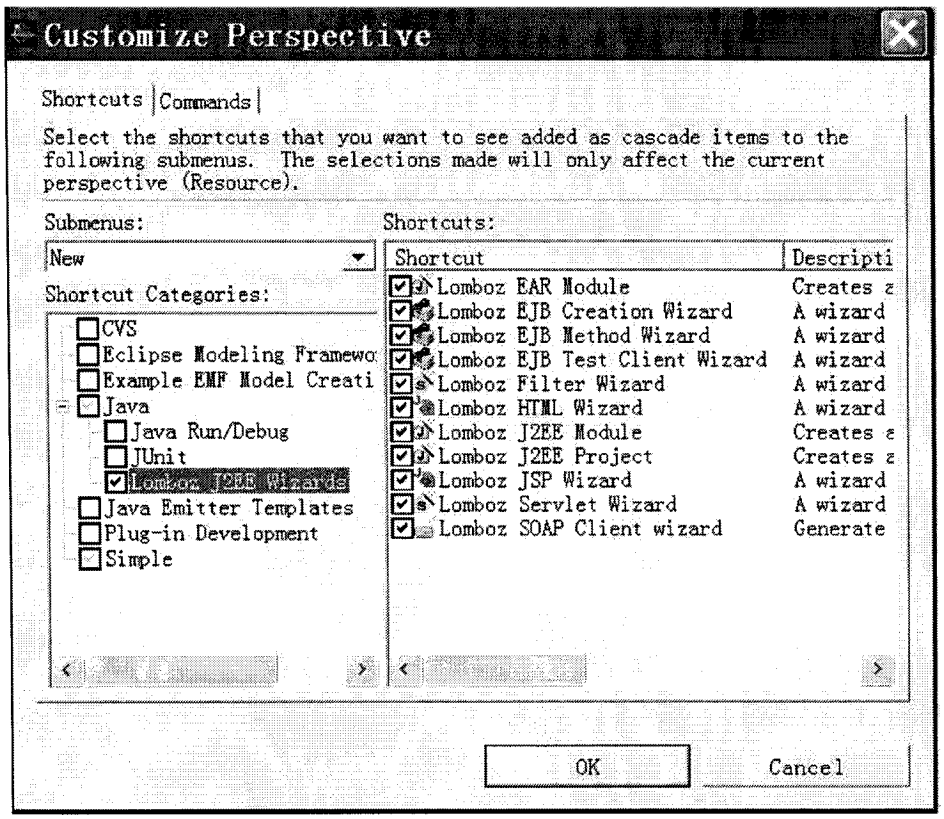

Figure A. 2 Setting of Lomboz J2EE Wizards

  - Select Window->Customize Perspective-> Shortcuts->Show View. Check Lomboz J2EE.(See Figure A. 3)

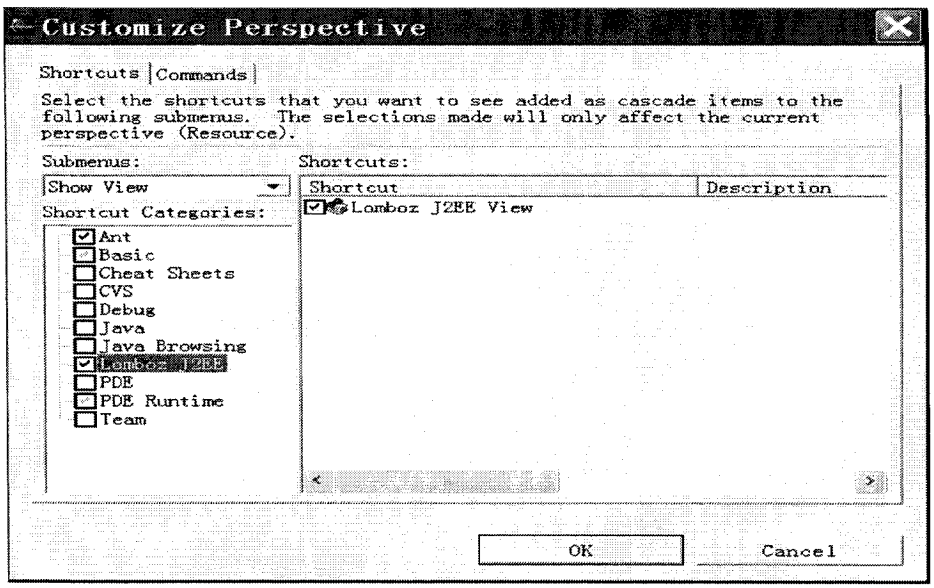

Figure A. 3 Setting of Lomboz J2EE View

- Select Window->Customize Perspective-> Commands. Check Lomboz Actions

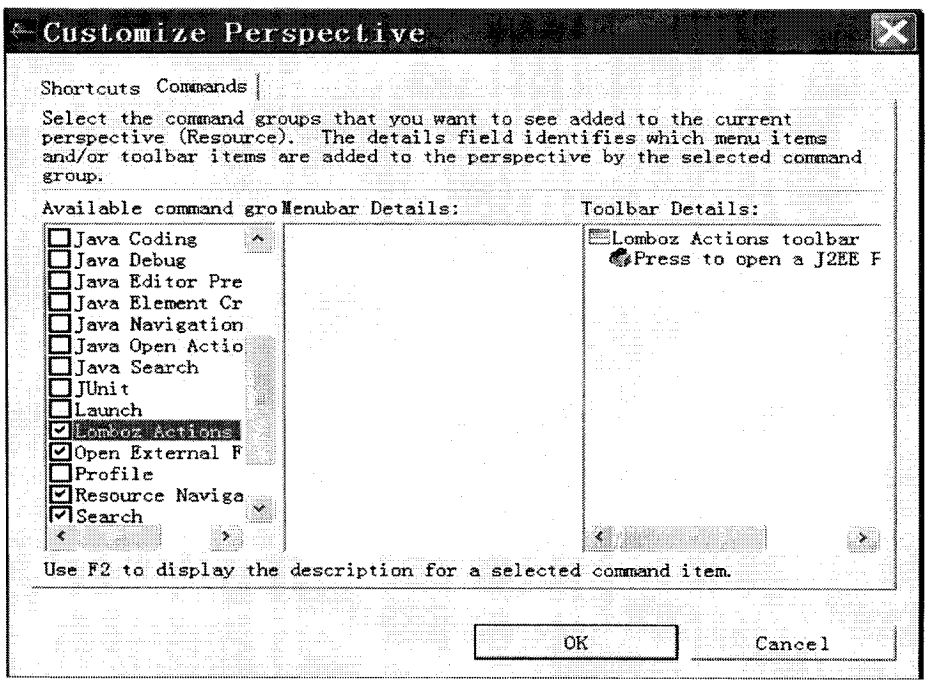

Figure A. 4 Setting of Lomboz Actions

- Select Window->Preferences->Workbench->Label Decorations. Check Lomboz J2EE Decorators.(See Figure A. 5)

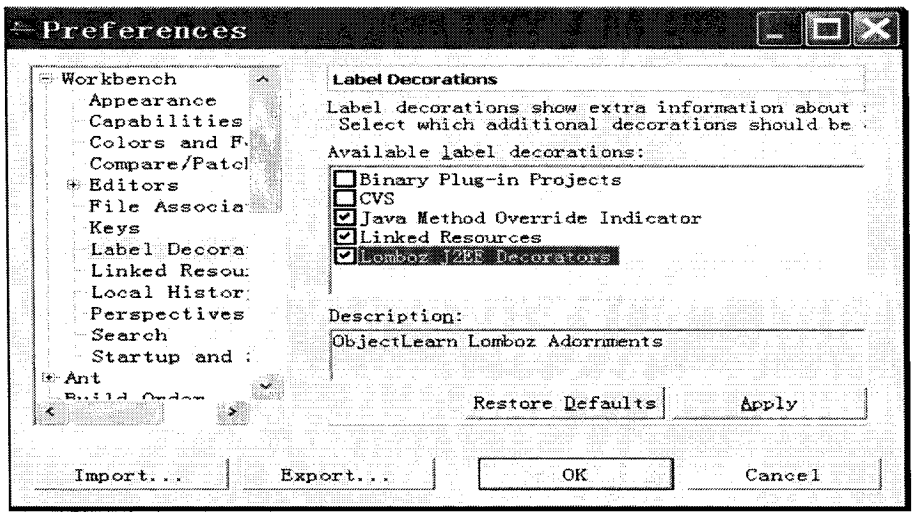

Figure A. 5 Setting of Lomboz J2EE Decorators

- JDT configuration: Define Java projects to have separate source and binary folders.

On Eclipse menu bar, select Window->Preferences->Java->Build Path. Click Folders.

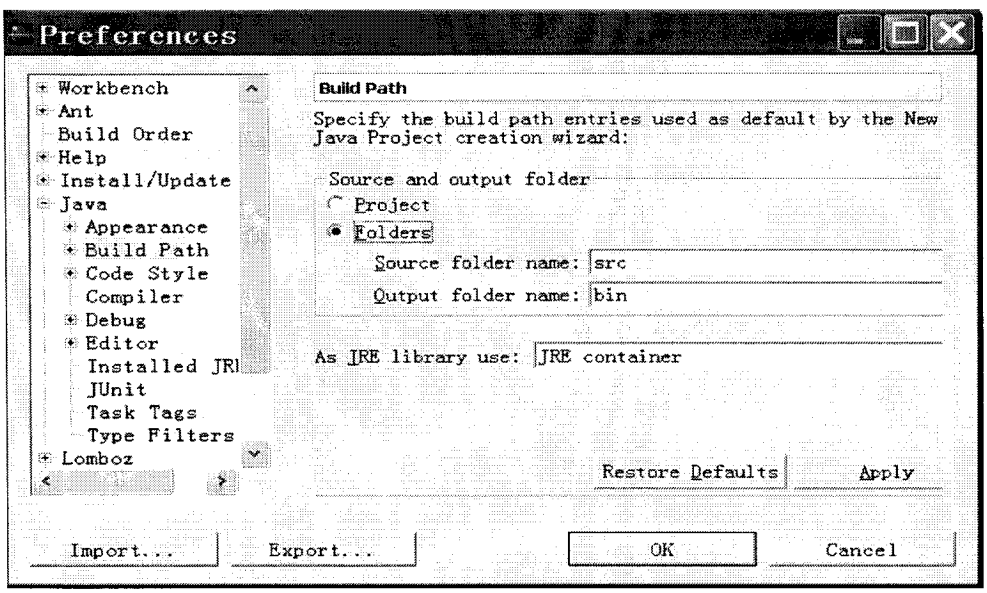

Figure A. 6 Setting of Java Build Path

- JDT and Lomboz

  On Eclipse menu bar, select Window->Preferences->Lomboz, change "JDK Tools.jar" to the actual JDK installation directory: C:\j2sdk1.4.2_05\lib\tools.jar.(See Figure A. 7) This Lomboz preference defines the Java SDK, which will be used by Apache Tomcat application server. This setting also defines the classpath variable JDK_TOOLS to the above value. (Also See Figure A. 1)

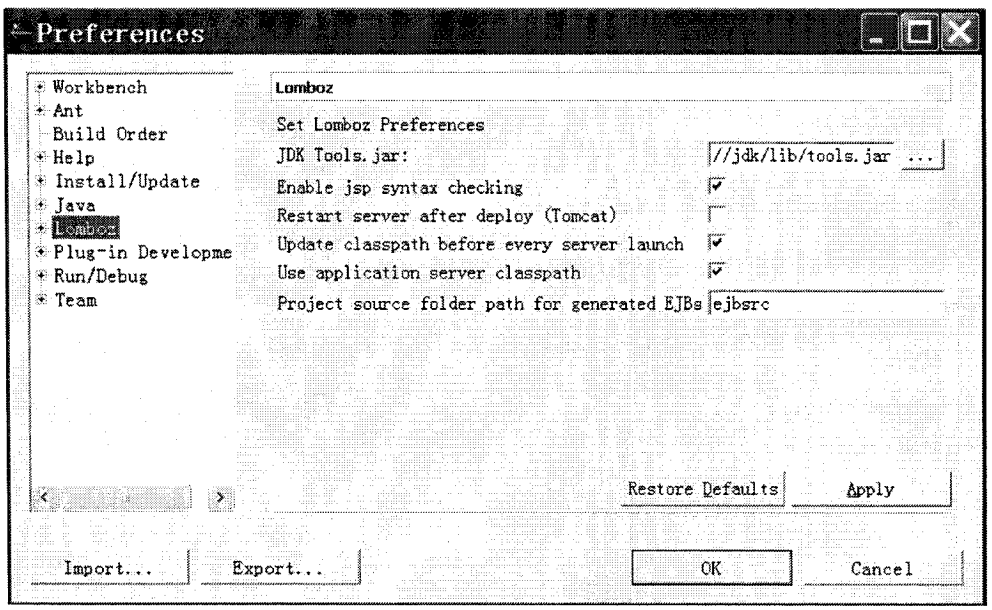

Figure A. 7 Setting of Lomboz

- Server definitions

  Lomboz can work with a variety of Java application servers. On Eclipse menu bar, select Window->Preferences->Lomboz->Server Definition, choose Server types as Apache Tomcat v.5.0.x. (See Figure A. 8)

  In the same window under the "Properties" item, change "Application Server Directory" and "Classpath Variable" to Apache Tomcat server actual installation directory: D:/jarkata-tomcat-5.0.27. (See Figure A. 9) And this setting will automatically change classpath variable—TOMCAT_HOME, as shown in Figure A. 10.

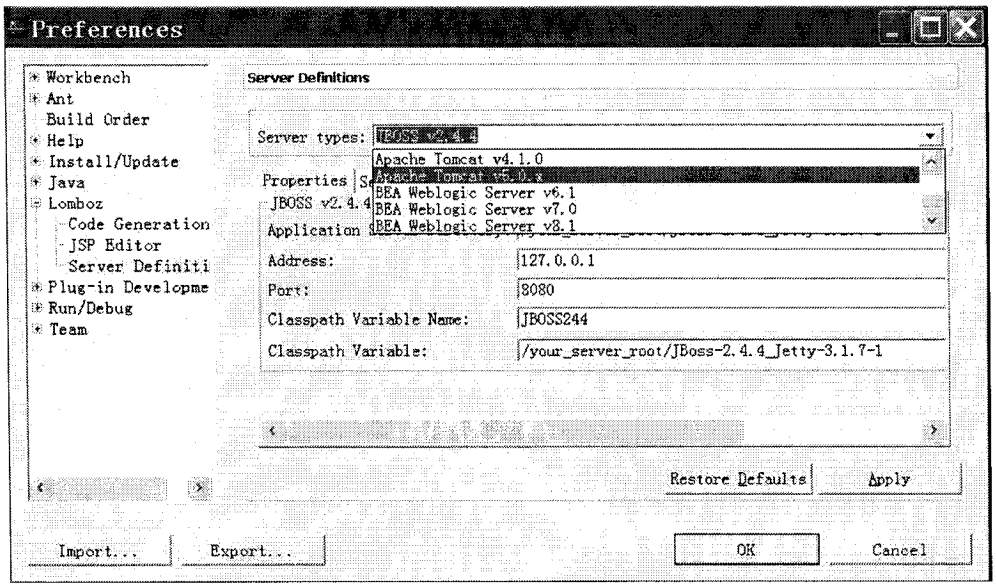

Figure A. 8 Application Server Definition

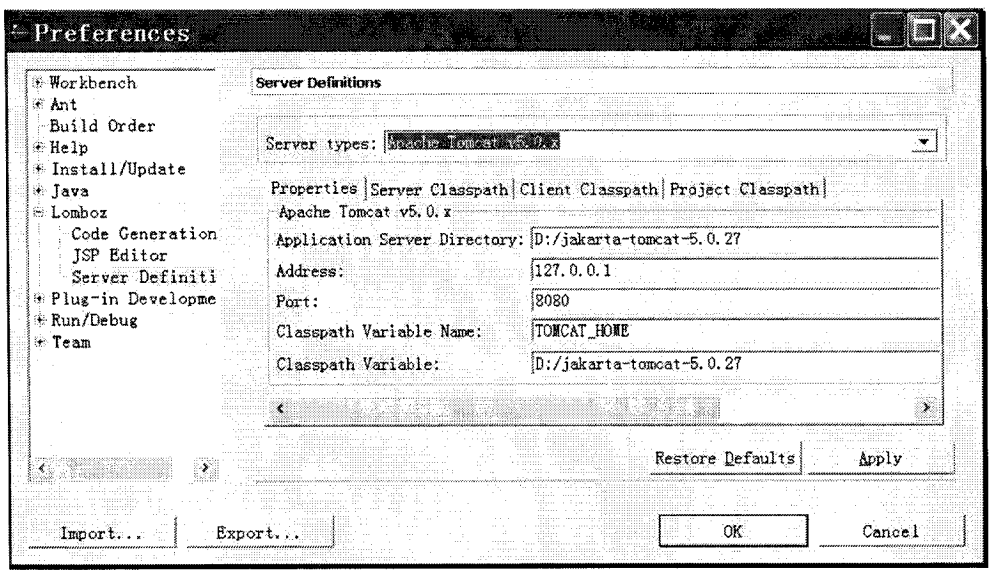

Figure A. 9 Application Server Properties

- Verifying Property

On Eclipse main menu, select Window->Preference->Java->Build Path->Classpath, all Classpath Variables setting can be seen set properly. (See Figure A. 10)

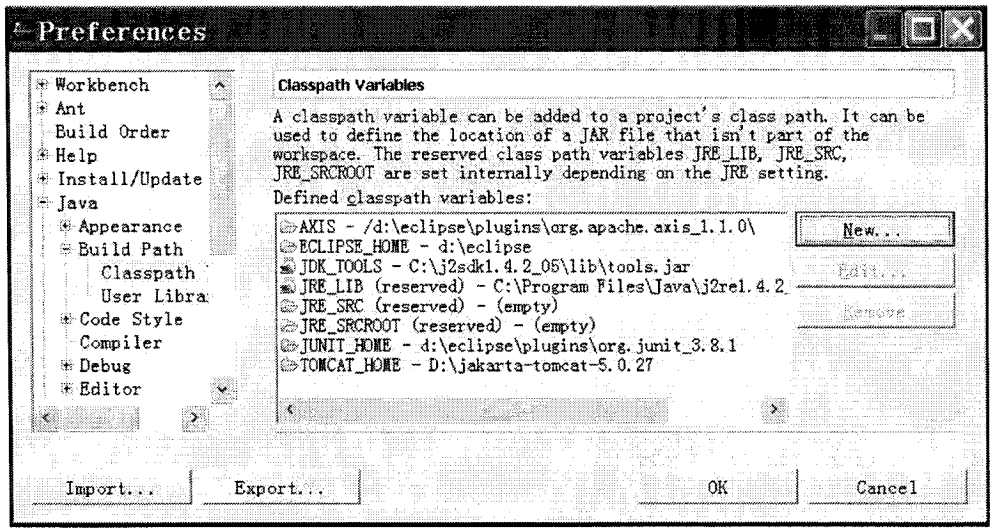

Figure A. 10 Verification of Classpath Variables

- Apache Tomcat server configuration

In order to successfully run/stop, deploy/undeploy apache tomcat server 5.0.27 from Eclipse Lomboz environment, edit file tomcat50x.server in D:\eclipse\plugins\com.objectlearn.jdt.j2ee_3.0.1\servers as follows:

Change

-Djava.endorsed.dirs="${serverRootDirectory}/bin;${serverRootDirectory}/common/endorsed"

to

-Djava.endorsed.dirs="${serverRootDirectory}/common/endorsed"

in the following 'tags':

<startVmParameters>-DJAVA_HOME="${jrePath}"

-Dcatalina.base="${serverRootDirectory}" -Dcatalina.home="${serverRootDirectory}"

-Djava.io.tmpdir="${serverRootDirectory}/temp"

-Djava.endorsed.dirs="${serverRootDirectory}/bin;${serverRootDirectory}/common/endors
ed"</startVmParameters>

and

<stopVmParameters>-DJAVA_HOME="${jrePath}"

-Dcatalina.base="${serverRootDirectory}" -Dcatalina.home="${serverRootDirectory}"

-Djava.io.tmpdir="${serverRootDirectory}/temp"

-Djava.endorsed.dirs="${serverRootDirectory}/bin;${serverRootDirectory}/common/endors
ed"</stopVmParameters>

- Setting up Web project/module libraries in Eclipse

  My current working environment is:

  Lomboz J2EE project: test

  Lomboz J2EE module (Web module): watermarking

  According to J2EE conventions, all jar files should be kept in directory WEB-INF/lib of each Web module. When we need to use a .jar supplied by others, we right click on the project name Test and choose Property. Choose Add Jars or Add External Jars to add .jar. (See Figure A. 11)

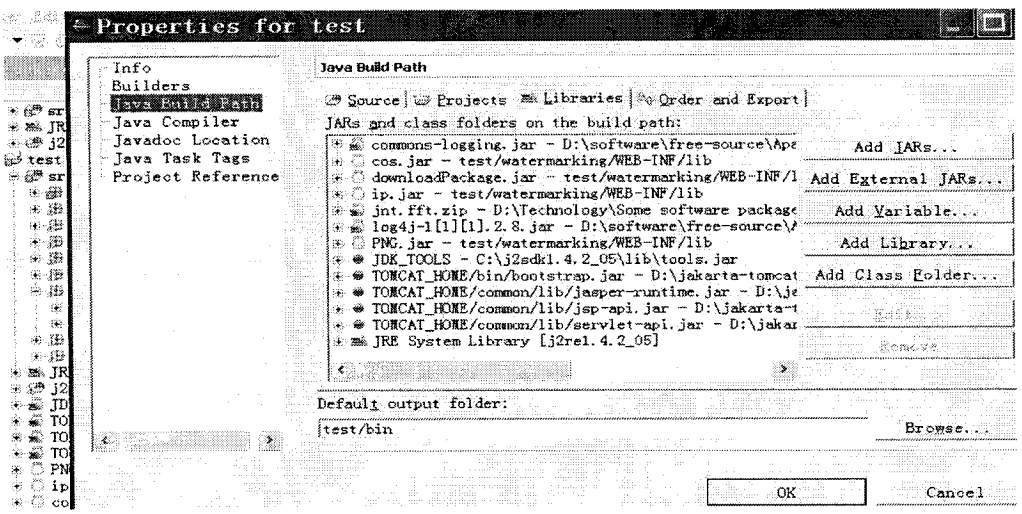

Figure A. 11 Adding Jars or External Jars

Then, right clock web application watermarking, choose Lomboz J2EE-> Add WEB-INF/lib jars to classpath. (See Figure A. 12)

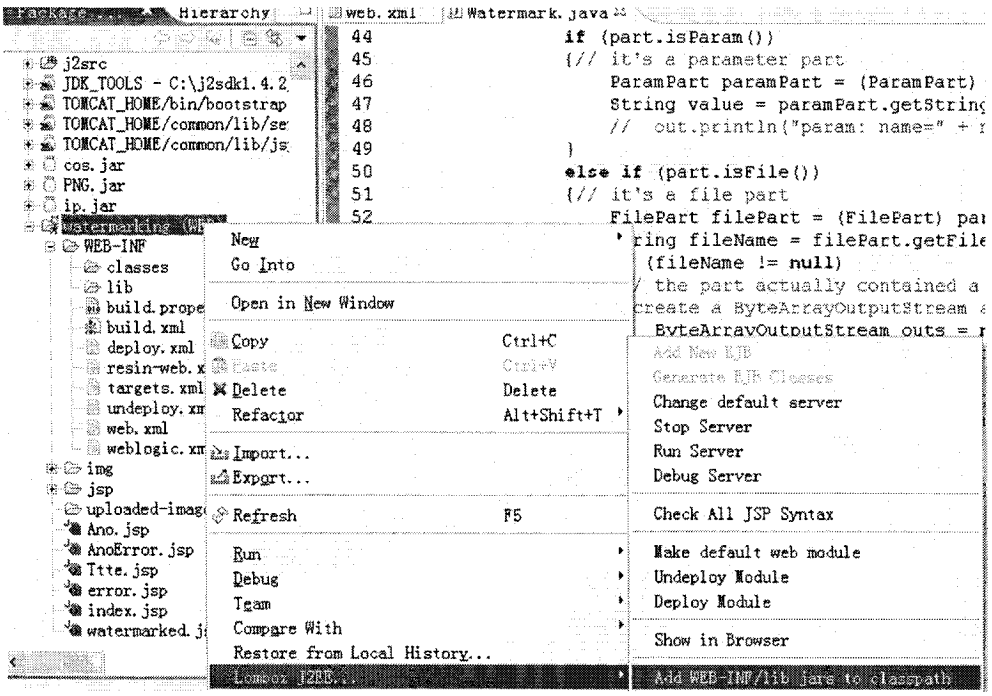

Figure A. 12 Add WEB-INF/lib Jars to Classpath in Lomboz J2EE

B. Setting of the Production Environment in Windows XP

1. Download the latest Java SDK 1.4.2 from [43] and save in directory C:/j2sdk1.4.2_08.
2. Download Apache tomcat server v5.0.27 from [2].
3. Install tomcat server in directory C:/jakarta-tomcat-5.0.27.

   Notice that Java virtual machine directory name is needed in the installation process, which is C:/j2sdk1.4.2_08 instead of C:/j2sdk1.4.2_08/bin.
4. Generate the latest version of the developed watermarking system.

   In Eclipse, deploy the developed Web system watermarking into Apache Tomcat server started by Lomboz. This will generate the latest version of the system in a war file package

called C: \jakarta-tomcat-5.0.27\webapps\watermarking.war in the Tomcat server directory. Then exit Eclipse.

5. Run the standalone Apache tomcat server.

   Go into the subdirectory of the Jakarta Tomcat server installation directory C:\jakarta-tomcat-5.0.27/bin, and run the executable file tomcat5.exe. To test if the server is setup, access the URL http://localhost:8080 on the same machine and see if the contents as shown in Figure B. 1 can be seen.

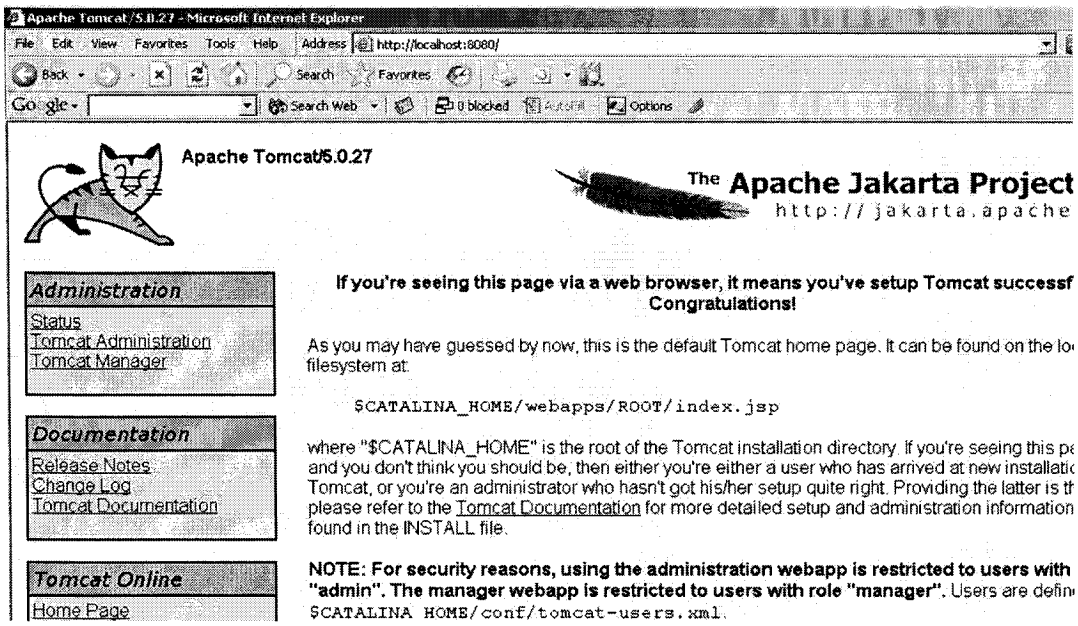

Figure B. 1 Apache Tomcat Server

6. Deploy the web application watermarking.war into the Apache Tomcat server.

   Deploy the war file C:\jakarta-tomcat-5.0.27\webapps\watermarking.war into the server by using the administration tool--Tomcat Manager, as shown in Figure B.2. Then, the web application watermarking can be seen listed in the "Application" directory.

7. Build a working directory.

   Create a directory with the same name defined in the watermarking web application deploy descriptor file: web.xml: D:/uploadImage. This directory is to save uploaded images and

working files on the server.

8. Visit the web application from Web browser by the URL.

   The web application can be accessed by visiting the 8080 port: http://ipaddress:8080/watermarking/

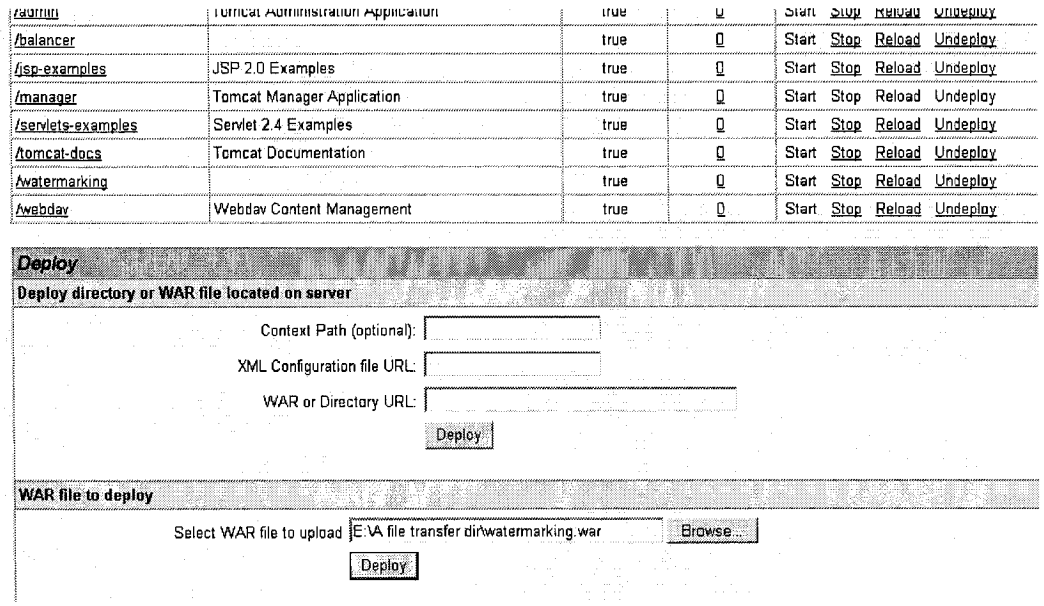

Figure B. 2 Deploy Web Application

C. List of Source-code Programs of the Watermarking System

Source codes are wrapped in kia.jar for other's reference. They are also available in watermarksource.zip file. The following are a list for the source codes

C.1 Servlet Prgorams

EDU.ou.watermark:

- Constants.java
- Hamming4.java
- ImageFile.java
- Utilities.java

EDU.ou.watermark.web:

- UploadDispatch.java
- Patchembedding.java
- Extracting.java

RotateExtracting.java
ImageRetriever.java
ColorToGray.java
FormatTran.java
GeometricTran.java
ShowSpectrum.java
ShowPhase.java

EDU.ou.watermark.algorithm:
Embedding.java
Extracting.java

EDU.ou.watermark.algorithm.lsb:
LSB.java
ExLSB.java

EDU.ou.watermark.algorithm..brightness:
Amplitude.java
ExAmplitude.java

EDU.ou.watermark.algorithm.patchwork:
Patchwork.java
ExPatchWork.java

EDU.ou.watermark.algorithm.FftAmp:
FftAmplitude.java
ExFftAmplitude.java

EDU.ou.watermarki.algorithm.Fftphase:
FftPhase.java
ExFftPhase.java

C.2 JSP pages

index.jsp
embedding.jsp
extracting.jsp
patchwork.jsp
rotatedetector.jsp
formattran.jsp
geotran.jsp
colortogray.jsp
showspectrum.jsp
showphase.jsp
displayexstr.jsp
dispimage.jsp

C.3 Configuration File

web.xml