GIS Web Services using .NET Framework

by

Ching-Huei Tsou

M.S., Structural Engineering (1997)
B.S., Civil Engineering (1995)
National Taiwan University

Submitted to the Department of Civil and Environmental Engineering
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF ENGINEERING
in Civil and Environmental Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 2002

© 2002 Ching-Huei Tsou, All rights reserved.
The author hereby grants M.I.T. permission to reproduce and to distribute publicly paper
and electronic copies of this thesis document in whole or in part.

Signature of Author

Department of Civil and Environmental Engineering
May 15, 2001

Certified by

Kevin Amaratunga
Assistant Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by

Oral Buyukozturk
Chairman, Departmental Committee on Graduate Studies
GIS Web Services Using .NET Framework

by

Ching-Huei Tsou

Submitted to the Department of Civil and Environmental Engineering
on May 15, 2002 in partial fulfillment of the
requirements for the degree of
Master of Engineering in Civil and Environmental Engineering

Abstract

Geographic Information System (GIS) has been extensively used in technical fields in the past two decades. The focus of this thesis is to develop a robust, efficient, and scalable GIS, which is designed as web services and implemented on .NET framework. Wavelet image compression and parallel computing techniques are used in the implementation to achieve this goal.

By building the system as web services, users all around the world can access the GIS through browsers without installing any additional program, and programmers can develop their own applications on top of the proposed web services. By using wavelet theory, the system can highly compress the image data before it is broadcasted over the Internet. However, performing wavelet compression on a web server will affect the performance of the server significantly; hence, a Beowulf cluster and parallel computing is used for the image processing.

A prototype of the proposed GIS was tested in an environmental field study, and it gave researchers in a remote location the ability to influence the sampling process in real-time. The results of image compression and parallel computing also show that the system works efficiently and can be scaled well.

Thesis Supervisor: Kevin Amaratunga

Title: Assistant Professor of Civil and Environmental Engineering
Acknowledgments

To Prof. Kevin Amaratunga and Prof. Ruaidhri M. O’ Connor, who gave timely and
thoughtful advice to help me clarify my thoughts and direction.

To all M.Eng. classmates, especially thanks to members in the STEFS, for helping me
make it through a wonderful year of my life.

To my beloved girl friend, Chia-Chi Chang, for encouraging and giving me strength to do
what I had to do.

And to my parents for their continued support and encouragement through these
challenging years. They brought all the love and strength to my life.
# Table of Contents

List of Figures .......................................................................................................................... 6  
List of Tables and Code Samples .......................................................................................... 8  

Chapter 1 Introduction ........................................................................................................ 9  
   1.1 Motivation ................................................................................................................... 9  
   1.2 Software Tools for Environmental Field Study .......................................................... 9  
      1.2.1 Overview ............................................................................................................. 10  
      1.2.2 Hardware and Software ...................................................................................... 10  
   1.3 Microsoft .NET Framework and Web Services .......................................................... 11  
   1.4 Problem Statement ...................................................................................................... 12  
   1.5 Thesis Organization ..................................................................................................... 14  

Chapter 2 System Overview ................................................................................................. 15  
   2.1 Geographic Information System .................................................................................. 15  
      2.1.1 Overview ............................................................................................................. 15  
      2.1.2 Source of Data of the Proposed GIS .................................................................. 15  
   2.2 Background of Web Services ...................................................................................... 16  
      2.2.1 XML, SOAP, UDDI, and WSDL .............................................................. 18  
      2.2.2 Preview of the clients of the GIS Web Services ................................................. 20  
   2.3 Hardware Architecture ................................................................................................. 23  
   2.4 Software Architecture .................................................................................................. 25  

Chapter 3 Building and Accessing Web Services on .NET Framework ......................... 27  
   3.1 Building a Web Service ............................................................................................... 27  
   3.2 Building a Web Application ......................................................................................... 29
3.4 GIS Viewer - a .NET Windows Application ........................................... 32
3.5 Executing the GIS Viewer in Internet Explorer ..................................... 33

Chapter 4 Image Compression and Parallel Computing ............................ 36

4.1 Wavelet Theory ................................................................................. 36
  4.1.1 Multiresolution Analysis ................................................................. 36
  4.1.2 Wavelets and Filter Banks ............................................................... 39
  4.1.3 Two Dimensional Filter Banks and Image Compression ................. 41
  4.1.4 Quantization and Lossless Compression ....................................... 42
  4.1.5 Image Reconstruction ................................................................... 44

4.2 Parallel Computing ............................................................................ 46
  4.2.1 TCP/IP Socket .............................................................................. 47
  4.2.2 The Beowulf Cluster and Message Passing Interface Standard ...... 49
  4.2.3 The Message Passing Programming Model .................................. 50

Chapter 5 Results and Discussions .......................................................... 52

5.1 GIS Web Services ............................................................................... 52
5.2 Wavelets Image Compression ............................................................. 55
5.3 Speed-up by Parallel Computing ......................................................... 62

Chapter 6 Summary and Future Work ....................................................... 64

6.1 Summary .......................................................................................... 64
6.2 Future Work ...................................................................................... 65

Bibliography ......................................................................................... 67
List of Figures

Figure 1-1 Hardware Architecture of STEFS System ................................................. 11
Figure 2-1 Accessing an Object from Other Codes .................................................. 17
Figure 2-2 Accessing a Web Object from another Machine .................................. 17
Figure 2-3 Accessing GIS Web Services through a Web Application .................. 21
Figure 2-4 Accessing GIS Web Services through a Windows Application – PhotoMap .. 22
Figure 2-5 Accessing GIS Web Services through a Windows Application – TopoMap ... 22
Figure 2-6 Hardware Architecture ......................................................................... 24
Figure 2-7 Software Architecture (with Web Application) .................................... 25
Figure 2-8 Software Architecture (with Windows Application) .......................... 25
Figure 3-1 Registering a Web Service on UDDI using Microsoft Visual Studio .NET .... 29
Figure 3-2 Find a Web Service from UDDI using Microsoft Visual Studio .NET ....... 30
Figure 3-3 Invoking Web Service using a Web Application .................................. 31
Figure 3-4 Invoking Web Service using a Web Application .................................. 35
Figure 4-1 Two Band Analysis Bank ................................................................... 41
Figure 4-2 Two Dimensional Wavelet Decomposition ........................................ 41
Figure 4-3 Three Level Wavelet Decomposition ................................................ 42
Figure 4-4 Quantization ...................................................................................... 43
Figure 4-5 Lossy and Lossless Image Compression ............................................. 44
Figure 4-6 Two Dimensional Wavelet Reconstruction ......................................... 45
Figure 4-7 Compression and Decompression Model .......................................... 46
Figure 4-8 Open System Interconnection 7 Layers Model .................................... 47
Figure 4-9 Client-Server Sockets ......................................................................... 48
Figure 4-10 Splitting an Image into n Strips / processes........................................ 51
Figure 5-1 Basic GIS Web Methods............................................................................ 52
Figure 5-2 Web Methods for Accessing the .NET TerraServer................................. 53
Figure 5-3 Applying the GIS Web Services in Field Study....................................... 54
Figure 5-4 Original Image (Charles River)................................................................. 55
Figure 5-5 Color Intensity Distribution of the Original Image (Red, Green, and Blue)... 56
Figure 5-6 Single Stage Decomposition..................................................................... 56
Figure 5-7 Intensity Distribution in the Red Channel (X: Intensity – Y: Percentage)...... 57
Figure 5-8 Quality Comparison.................................................................................. 59
Figure 5-9 Separating an Image into RGB Channels............................................... 60
Figure 5-10 Separating an Image into YUV Channels.............................................. 60
Figure 5-11 Image Compression Results in YUV (8KB) and RGB Color Space (10KB) 61
Figure 5-12 Speed-ups by Parallel Computing .......................................................... 63
List of Tables and Code Samples

Code Sample 1 - A Simple Web Service (C#).............................................................. 28

Code Sample 2 - A Simple Web Application (C#)....................................................... 31

Code Sample 3 - Executing the GIS Viewer in Internet Explorer (HTML).................. 35

Table 5-1 Comparison of the clients of the GIS web services ..................................... 54

Table 5-2 Single Stage Decomposition Result .............................................................. 57

Table 5-3 Two Stage Decomposition Compression Result ............................................. 58

Table 5-4 Four Stage Decomposition Compression Result .......................................... 59
Chapter 1 Introduction

1.1 Motivation

Many efforts have been made to bring new technologies into traditional environmental field study. Instead of using pen and paper, researchers now are able to collect field data via mobile computing devices integrated with automated sensors, and the information collected can be transmitted live to be processed. These tools save considerable time on data collection and transcription. This is great, but not enough. The rapid development of Internet has provided a new form of data exchange. If all these information can be streaming to a network system, then they can be shared between researchers, displayed world-wide on the Internet. Moreover, it will allow project managers to have a real-time influence on the project from a remote location. To achieve these objectives, a web based Geographic Information System (GIS) is proposed in this thesis.

1.2 Software Tools for Environmental Field Study

Software Tools for Environmental Field Study (STEFs) is a joint environmental engineering and information technology project in the Civil and Environmental Engineering Department in Massachusetts Institute of Technology. This Masters of Engineering project involved water resources and information technology team members to bring computing power to environmental field study. The GIS web services presented in this thesis are initially part of the project; therefore, to understand the role and the importance of the GIS web services, let’s first take a look at the whole integrated STEFS
1.2.1 Overview

The STEFS system is intended to be an innovative change to the traditional field study tools. The rapid developments of mobile computing and wireless communications in recent years have enabled us to build up new tools to assist field researchers in improving the data collection process. The system consists of multiple hardware components and software applications, and several wireless and Internet technologies are used for transferring and displaying collected data while still in the field. It allows researchers to use many different sensors to collect environmental data, to input them into mobile computing devices automatically or through graphical user interfaces (GUI), and to share the data between researchers immediately. The GIS web services presented in this thesis are used for displaying these data cartographically on the Internet. A prototype system has been completed and tested in field trials in Boston and Newcastle, Australia. Field studies demonstrated the noticeable gains achieved with the use of the data collection system. Commercialization potential exists for environmental, utilities and asset management, including the emergency disaster response.

1.2.2 Hardware and Software

The primary components of the STEFS system are an embedded Visual Basic application running on a handheld PC (Compaq iPaq PocketPC) equipped with a Hydrolab water quality sensor, a global positioning satellite receiver, and an 802.11b wireless PCMCIA card. Besides, there is a laptop used as a field server, which receives data from PocketPC wirelessly and stores it in a Microsoft SQL server database, and transmits the data through a GSM (Global System for Mobile communications) data enabled cell phone to a
web server. The web server then broadcasts the information over the Internet using GIS web services which are built on Microsoft .NET framework. Figure 1-1 shows the basic architecture of the STEFS system and the primary software used.

![Hardware Architecture of STEFS System](image)

**Figure 1-1 Hardware Architecture of STEFS System**

### 1.3 Microsoft .NET Framework and Web Services

The Microsoft .NET Framework has two main components: the common language runtime and the .NET Framework class library. The common language runtime is the infrastructure of the overall .NET platform. It acts as an agent that manages code at execution time, providing memory management, thread management and other core services. The class library is a comprehensive, object-oriented collection of reusable types that can be used to develop many kinds of applications including Web Forms and XML Web services. One major goal of .NET is to provide a platform for building and delivering web services on the Internet, i.e., .NET is trying to make the Internet to be the
basis of a new operating system, and the key ideas are web services.

Web services are pieces of software which are available over the Internet and act as blocks in building distributed applications. They dramatically simplify the way of sharing data by using open standards. Further, applications which are constructed using web services can take advantages from various sources regardless where they are and how they were implemented. As a result, web services are ideal for building platform independent applications focus on communication, collaboration and integration.

Since our goal is to provide a robust application for sharing information over the Internet, web services and .NET framework are chosen as the format and platform to implement the system.

1.4 Problem Statement

As described in the previous sections, a web based GIS plays an important role in the integrated STEFS system. GIS has been extensively used in technical fields in the past two decades, by definition, a GIS is a combination of elements designed to store, retrieve, manipulate, and display these geographic data. There are three main purposes of creating a GIS for the STEFS project:

1. Makes data available live on handheld PCs from field server.

Since usually it is necessary to have more than one team doing field tests in different positions at the same time (there were three groups in the Newcastle, Australia field test),
it is very helpful that each group can see other groups’ data live through a graphical interface.

2. Makes data available live in laboratory from field.

Researchers who sit far from the field are able to see what is going on immediately, and decide where to go and what to do next. That is, it will allow project directors to have a real-time influence on the project from a remote location.


By setting up a web server and web services, one can publish the field test results to the Internet, and people all around the world can gain useful data by accessing the web services.

Further, because the graphic information is going to be sent on the Internet, and the bandwidth of network is limited, it is better to compress the data before they are transmitted. However, image compression operations will affect the performance of the web server significantly especially when it is accessed by many people simultaneously. The performance of the system is another important issue which has to be addressed.

To satisfy the data sharing requirements, the GIS web services using the Microsoft .NET framework is proposed. To highly compress an image and maintain reasonable quality, a wavelet compression process is introduced into this system. Also, to maintain the performance of the system, a parallel computing enabled cluster is used for performing the compression.
1.5 Thesis Organization

Chapter 1 gives an overview of the STEFS project, and motivation of the thesis, and the background and the role of the present GIS web services.

Chapter 2 describes the standards used in developing XML web services, and explains the hardware and software architectures in the GIS web services system.

Chapter 3 explains how we implement GIS web services in the Microsoft .NET, and how to build a web application so that users can access our web services through browsers. Alternatively, a windows application viewer is built which can also access the web services and without the restrictions of displaying only certain image types.

Chapter 4 introduces image compression and parallel computing into the system. Also, the wavelet theory behind the image compression process and the scheme of performing this operation in a parallel computing cluster are explained.

Chapter 5 shows a prototype of the GIS used in a field test, the benefits one can gain from performing the wavelet compression, and the performance increases one can have by doing parallel computing.

Chapter 6 gives the summary and the states the future work.
Chapter 2 System Overview

2.1 Geographic Information System

2.1.1 Overview
A GIS is a combination of elements designed to store, retrieve, manipulate, and display geographic data, i.e., information about places. It integrates common database operations such as query and statistical analysis and displays the results in a graphic manner. By providing visualized information, a GIS enables users to explore and analyze data by location; therefore, it is much easier to reveal the patterns, relationships and trends behind the data. In brief, a GIS is a tool to help people change the way to manage information.

The earliest GIS research work can be traced back to the transportation studies at the University of Washington in the 1950s [TW91]. On the other hand, modern GIS software has been developed primarily during the past two decades. The most popular commercial GIS software is ArcInfo and ArcView, made by ESRI (Environmental Systems Research Institute). They are initially used in the STEFS project as the only GIS software [SARTLK02]. However, in order to be better integrated with the other STEFS applications, to fulfill our own requirements, and to have more control of the system, we design and implement our own GIS in web services using the Microsoft .NET.

2.1.2 Source of Data of the Proposed GIS
Data in GIS comes in two parts: geographic data and attribute data. Geographic data
represent the physical places such as cities, rivers, lakes, and usually come in the form of maps. Attribute data are used for describing the characteristics of the places. In the present project, attribute data come from the field study, and geographic data are fetched from USGS (U.S. Geological Survey) database by using the Microsoft .NET TerraService.

**U.S. Geological Survey**

USGS was founded in 1879, and it has provided 585 digital linear tapes containing 14 TB of uncompressed Digital Ortho Quadrangle aerial imagery data and 950 CDROMs containing 530 GB of compressed Digital Raster Graphics topographical map data to the project. [Website-1]

**Microsoft .NET TerraServer**

Microsoft TerraServer .NET Web Service (TerraService) are application interfaces used to access the Microsoft TerraServer imagery database using the .NET technologies. The TerraServer use the image data from USGS and store all these data in six 1.5 TB SQL Server 2000 databases. Each database server runs on a separate server in a four-node Windows 2000 Server cluster. [Website-2]

**2.2 Background of Web Services**

Web services are pieces of software that make themselves available over the Internet and uses a standardized messaging system. If you are familiar with any object-oriented (OO) programming language, you can understand the concepts of web services very easily.
When writing an OO code, one encapsulates related data and methods into class, instantiate an object from the class, and then access the data in the object through the interface it provides. A client in this case is other pieces of code. See the figure below:

![Single Machine Diagram]

**Figure 2-1 Accessing an Object from Other Codes**

In the web services case, an object becomes a server which the services sit in. The web services themselves are the interface to access the resources provided by the server, and a client can be a program running in any machine which is connected to the internet. See the figure below:

![Machine A and Machine B Diagram]

**Figure 2-2 Accessing a Web Object from another Machine**

To put it simply, web services are classes with web front ends. Web services allow classes to be written and run on a server end. Therefore they can use the data provided by the
server, such as look up information in a database or perform an intensive calculation, and then return the results to the client. A web service works exactly the same as a method call in traditional OO programming language: it takes parameters from a client, performs the operations, and returns a value, except that the web service and the client usually run in different machines and pass the data via the Internet.

The idea of web services is simple. To implement web services, there are four essential standards we have to know: XML, SOAP, UDDI and WSDL. Those standards are briefly described in the following section.

2.2.1 XML, SOAP, UDDI, and WSDL

XML

Extensible Markup Language (XML) is a World Wide Web Consortium (W3C) proposed recommendation for a file format to easily and cheaply distribute electronic documents on the World Wide Web [W3C-1]. The design goal of XML is to provide a human readable, machine readable and document-ready document format. Besides, XML is extensible and well-formed, so it is much more useful in describing data than the display-oriented HTML. Also, XML documents contain the rules to which its data must conform, therefore it is self-describing and can be validated.

Because of these properties, XML is an ideal format for data exchange, and XML has become a standard information exchange mechanism over internet recently.
SOAP

Simple Object Access Protocol (SOAP) is a standard protocol which was proposed by Microsoft for the exchange of information in a decentralized, distributed environment [W3C-2]. SOAP is essentially XML and HTTP, and it consists three parts: (1) Envelope: defines a framework for describing what is in a message and how to process it, (2) Encoding rules: for expressing instances of application-defined data types, and (3) RPC convention: for representing remote procedure calls (RPC) and responses.

A simple way to think of SOAP is that it packs the XML data into an envelope, embeds the envelope as a header into HTTP, and then uses HTTP as the protocol for transferring the data over the Internet. Both XML and HTTP are all in text rather than binary, so it’s very easy to interoperate across machines without worrying about security issues. Since SOAP is lightweight, efficient, and secure and introduces no new technology beyond what is already being used, it becomes a trend for machine to machine communication.

UDDI and WSDL

UDDI stands for Universal Description, Discovery and Integration. A UDDI Business Registry (UBR) is a global public directory of businesses and services. Over 80 companies -- including business leaders like Merrill Lynch and Cargill; technology leaders like IBM, Microsoft, and Sun Microsystems; and innovative B2B companies like Ariba, CommerceOne, and VerticalNet -- have all signed on to support UDDI. [WebSite-3] It uses SOAP to publish, edit and browse the directory, and a user can send queries to UBR to find out published web services and the information to use them.

WSDL stands for Web Services Description Language. WSDL is an XML format for
describing where a web service is offered and how to use it [W3C-3].

To deploy a web service, one has to develop the methods, generate a WSDL file and then publish it to UBR. To use a web service, one can query the UBR and discover the WSDL file for that service, add a web reference to the client application, and then one can transfer data between client and server in XML format via SOAP according to the WSDL file.

2.2.2 Preview of the clients of the GIS Web Services

Before diving into the detail design of the system, let’s take a look at the graphic user interface and how to use it.

Web Application

A web application is a program running on the web server. It accepts inputs from clients, searches the database through our GIS web services, and then sends the requested information back to the users. To use this web application, the only requirement is a browser. Also, the output of this web application is only standard HTML and JavaScript, therefore any browser can be used as a client. Figure 2.3 is a screenshot of the web application which shows the result of the pH values along the Charles River in the Boston / Cambridge area.
To access our GIS web services, simply input the area of interest (a database file), and the system will fetch pieces of images from the Microsoft .NET TerraServer (according to the latitude, longitude, and resolution recorded in the database file) and compose a suitable map for the area, and list all the parameters available in that database. Then a user can select any parameter which he or she is interested from the drop-down menu, and the system will draw the corresponding data on the map. Also, our services can be used to locate a certain point on the map, and allow users to view data tables directly. Besides, if a user clicks on the map, the web application will invoke a web method to zoom in the map.

**Windows Application**

In addition to use a browser as the client of the web services, a stand alone viewer is also provided for accessing the services. Figure 2-4 and 2-5 are two screenshots from the .NET windows application, a GIS web services viewer.
Figure 2-4 Accessing GIS Web Services through a Windows Application
- Photographic Map

Figure 2-5 Accessing GIS Web Services through a Windows Application
- Topographic Map
It has all functions that the web application has, but with an improvement in data transmission (this will be discussed further in chapter 4). However, a .NET windows application has to be executed on a platform supporting the CLR (common language runtime), which is .NET framework only for now. Microsoft has sent their C# and CLI (common language infrastructure) to ECMA (European Computer Manufacturer’s Association), and on December 13, 2001, the ECMA ratified the C# and CLI specifications into international standards. (ECMA-334, 335). This means vendors all around the world can design their own common language runtime (in which the .NET code is running) target on any platform. To date, Microsoft has released a beta version of .NET Compact Framework on Windows CE based platform, and a Shared Source CLI distribution is available on FreeBSD. A .NET application will become platform independent eventually. Thus it will eliminate most of its limitations, and still keep all the benefits.

Alternatively, one can modify part of the code in this windows application and compile it as a windows control. A .NET windows control can be executed in the Internet Explorer as an applet, i.e., users can run this program inside a browser and need not to download the windows application manually. (This will be discussed more in Chapter 3.)

2.3 Hardware Architecture

Figure 2-6 shows the hardware architecture of the entire GIS web services system. In the system, all attribute data from field study are stored in a local database of the server, and
the geographic data (a suitable map) are taken from Microsoft .NET teraserver automatically when a specific area has been chosen by users. The system will then combine the two formats of data into a whole, and broadcast it over the Internet. The Beowulf cluster is used for image compression, and will be discussed more in chapter 4.

In this prototype, both web services and web application are running in the same web server. However, because the GIS functions are provided as web services, there is no need to restrict the web application and web services to the same machine. For example, since end users usually have a lower speed network, we can have a central web services server in Boston and many web application servers sit in different cities to handle the requests from local users. This is one of the main advantages of building our system as web services. Besides, programmers all around the world can develop their own systems by taking advantages of using the web services as part of their code, as the Microsoft .NET TerraServices are used to fetch geographic data in the present system.
2.4 Software Architecture

As mentioned before, two different ways to access the GIS web services are provided. The software architectures of both schemes are shown in Figure 2-7 and 2-8.

![Software Architecture (with Web Application)](image_url)

Figure 2-7 Software Architecture (with Web Application)

In the first scheme, a web application is used as an agent between the web services and clients, and any browser can be used to access the web services through this web application.

![Software Architecture (with Windows Application)](image_url)

Figure 2-8 Software Architecture (with Windows Application)
The primary differences between the second scheme and the previous one are:

- It uses an image processing unit to compress images before they are sent to the clients.
- The client application can talk to the web services directly so there is no need to have a web application.

The details of how to implement these schemes will be explained in the following chapters.
Chapter 3  Building and Accessing Web Services on .NET Framework

In this chapter, the concepts of how to build a web service on Microsoft .NET frameworks by using Microsoft Visual Studio .NET, how to create a web application as the front end of the web service, and how the clients can access the web service through the web application are discussed. Throughout this chapter, a simple example is used to demonstrate the concepts since the actual programs are too lengthy to show in here.

3.1 Building a Web Service

As mentioned before, our GIS web services can fetch a suitable map from the Microsoft TerraServer and combine it with our attribute data sampled from field studies. To simplify, a fixed map and data is used to demonstrate the concepts of building a web service. Here is the code sample in C#:

```csharp
[WebMethod]
public string drawRect(string picName, int X, int Y)
{
    // read an image from file
    const string path = @"C:\Inetpub\wwwroot\GISWebService\Pics\";
    Bitmap map = new Bitmap(path + picName + ".jpg");

    // draw a white rectangle on the image
    for (int i = -3; i < 3; i++)
        for (int j = -3; j < 3; j++)
            map.SetPixel(X + i, Y + j, Color.White);
}
```
Code Sample 1 - A Simple Web Service (C#)

To build a web service, first the keyword [WebMethod] is used to declare that it is a web method. Then we read a fixed image from file, draw a white rectangle on it, save the image as a new file, and return the full location of the new file. A simple web method is very close to a regular method, except the keyword used in the beginning.

After creating a web method, we have to deploy it on the Internet so people can find the service though UDDI. For example, using the integral tool in Microsoft Visual Studio .NET, one can simply go to the “XML Web Services” section on the start page, chose “Register a Service” (or go to the Microsoft UDDI register page on WWW directly: http://www.gotdotnet.com/vs/registerwspage.aspx), follow the instructions and input the necessary information as shown in Figure 3-1:
3.2 Building a Web Application

Web services are pieces of code available on the Internet; they can be accessed by programmers using the standard protocols very easily, but not for end users. Therefore, to make the web services a useful tool for every user, one needs to build a web application for accessing the services. The role of a web application in the present system is that it acts as a front end for the web services; people can access the web services through the
GUI without having any knowledge of programming.

To use web services in a web application, first search the UBR for existing services, and then one can add a web reference to the program. For example, if “GIS” is used as the keyword to search the UBR, one can find the service we have just registered before, as shown in Figure 3-2. Now the web reference can be added into the application.

Figure 3-2 Find a Web Service from UDDI using Microsoft Visual Studio .NET

A web reference is a link from the web application to the existing web service; it acquires the WSDL file from UDDI, and the WSDL will tell the programmer how to invoke the web service.

After adding a web reference pointing to the web service, the service can be used as a web class, i.e., one can create an object of the web service:
private void Go_Click(object sender, System.EventArgs e)
{
    // create a web service object
    WebReference1.Service1 rs = new WebReference1.Service1();

    // read inputs from user
    int x = Convert.ToInt32(X.Text);
    int y = Convert.ToInt32(Y.Text);
    string pic = fileName.Text;

    // display the image
    map.ImageUrl = rs.drawRect(pic, x, y);
}

Code Sample 2 - A Simple Web Application (C#)

Then one can use the web object to invoke a method in the web service (the drawRect method is a web method built earlier in chapter 3.1). Here is a screenshot of this web application (the GUI is created by using the visual tools in Microsoft Visual Studio .NET)

Figure 3-3 Invoking Web Service using a Web Application

So far, we have explained how to build a web application for accessing the present web services. A web application has a lot of benefits. First, the output of a web application is
standard HTML and JavaScript, which can be interpreted by almost all platforms and browsers, i.e., the system is platform independent, and a user can use the GIS without installing any software or plug-ins. Second, if the GUI or the system is upgraded in the future, a user can benefit from those improvements immediately without doing anything.

3.4 GIS Viewer - a .NET Windows Application

Using a browser as the client has many benefits; however, there are still some downsides of it. For instance, a browser can not be integrated with other application. It makes the GIS a separate part of the whole STEFS system. Moreover, for security reason, a web application is actually running on the server side, but sometimes we do want to use the computing ability on the client side, for example, decompressing our own image format.

To resolve these problems, a windows application version of GIS client in built. By developing our own application instead of using a browser, the wavelet theory can be used to compress an image (which is usually better than JPEG) in the server side before it is sent to the client, and then decompress the image in the viewer, so the time spends in data transmission can be reduced. The detail compression process will be discussed in the next chapter, and now we will focus on how to change our GIS web application to a windows application.

The beauty of .NET is that everything is running on the same framework. Although a web application and a windows application look quite different, both of them are executed on the same common language runtime and share a large amount of .NET framework
libraries. In our case, moving from a web application to a windows application, there are only three major changes we have to notice:

- The user interface is created by using HTML in a web application, and created by Windows Forms in a windows application.
- The base class of a web application is inherited from the “System.Web.UI.Page” class, but the base class is inherited from “System.Windows.Forms.Form” for a windows application.
- Graphics cannot be done in a browser directly. In a web application, drawing on an image behind is done behind the scene and then the image is displayed in a browser by passing its URL to the browser. However, in a windows application, an image can be draw and display on the screen directly.

A windows application based GIS viewer can be done by making the above changes to the existing web applications, and the user interface of the viewer has been shown earlier in figure 2-4 and 2-5.

3.5 Executing the GIS Viewer in Internet Explorer

As mentioned in section 2.2.2, a .NET windows application has to be downloaded and installed manually, and that is inconvenient for the users. Beside, it is difficult to update the application once it has been released. One way to address this issue is to recompile the code as a windows control, and then it can be invoked in Internet Explorer [E02]. By doing this, users do not need to manually install anything, and programmers can modify
the code or add new features to the application at any time.

There is only a few changes need to be done before one can recompile a windows application as a windows control:

- For a windows control library, the base windows form is inherited from the higher level “System.Windows.Forms.Control” class instead of the lower level “System.Windows.Forms.Form” class.
- A form can be used as the starting class in a windows application by placing a method called Main in the class, and add the [STAThread] attribute to the Main method in order for the form to run. However, a windows control cannot be executed directly so there is no Main method and [STAThread] for a windows control.

After these minor modifications of the code, one can recompile the program as a windows control library, and generate a .dll windows form library. To execute the library inside a HTML file, one needs to embed an object, and specify a unique “classid”. A classid consists of two parts: the path of the control library and the fully qualified name (namespace plus the name of the form) of this control, separated by a pound sign. An example is shown in code sample 3, and the result is shown in figure 3-4.

```html
<html>
<head>

    <title>Executing the GIS Viewer in Internet Explorer</title>

</head>

<body bgcolor="#ECE9D8" text="#000000">

<font size="5">Executing the GIS Viewer in Internet Explorer</font>
<object id="myControl"  
</body>
</html>
```
By wrapping our windows application as a windows form control, one can execute the view through the browser and no additional software needs to be installed in the client side. Therefore, user can always use the most updated viewer without downloading any application manually. However, currently this alternative only supports Internet Explorer, and version 6 or higher is required.
The size of data sent from a GIS server is usually quite large (in the form of an image). Since the bandwidth of the network is limited, it is better to compress the data before it is sent from the server, and then uncompress it in the client side. Hence, a wavelet image compression algorithm is introduced into our system in this chapter. Besides, to reduce the impact of performing this extra operation in the web service server, a Beowulf Linux cluster and parallel computing technique is used for presenting the image compression.

4.1 Wavelet Theory

Wavelet analysis is a relatively new mathematical method concerned with efficiently representing information in a hierarchical way. It has been widely used for continuous function analysis and discrete signal processing. The early work started in the 1980's by many researchers, and after the paper by Daubechies [D88] that caught the attention of the larger applied mathematics communities in signal processing, statistics and numerical analysis [BGG98]. Although the whole theory behind wavelets in the mathematical form is complex, the key can be interpreted in a very simple way.

4.1.1 Multiresolution Analysis

One of the central ideas of wavelets is multiresolution analysis where the decomposition
of a signal is in terms of the resolution of detail. In the signal processing area, one is most concerned with the \( L^2(R) \) function space. The \( 'L' \) means a Lebesque integral, and the \( '2' \) denotes the integral of the square of the modulus of the function, and the \( 'R' \) states the variable of integration is a number over the whole real line. This is the space of all functions with a well defined integral of the square of the modulus of the function, i.e., with finite energy. Any finite energy function can be expressed as a combination of basis functions for the \( L^2(R) \) function space. This is,

\[
f(t) = \sum_i c_i \cdot \phi_i(t)
\]  

(4.1)

If all the functions within one space, \( V_0 \), are also in another function space \( V_1 \), it is said that \( V_0 \) is a subspace of \( V_1 \). Then a multiresolution decomposition for \( L^2(R) \) can be constructed as a series of subspaces \( V_j \) such that

\[
V_{-\infty} \subset \cdots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots \subset V_{\infty}
\]  

(4.2)

where

\[
V_{-\infty} = \{0\} \quad and \quad V_{\infty} = L^2(R)
\]  

(4.3)

If we chose

\[
\phi_{0,k}(t) = \phi(t-k), \quad k \in \mathbb{Z} \quad and \quad \phi \in L^2(R)
\]  

(4.4)

as the basis set of \( V_0 \), the basis set of \( V_j \) can then be defined as a set of scaled and shifted version of \( \phi(t) \):
\[ \phi_{j,k}(t) = 2^{j/2} \cdot \phi(2^j t - k) \] (4.5)

For \( j > 0 \), the \( \phi_{j,k}(t) \) is narrower than \( \phi_{0,k}(t) \) and shifted in smaller steps, thus the resolution of \( V_j \) is larger than \( V_0 \). On the contrary, the resolution is smaller when \( j < 0 \).

If a proper scaling function \( \phi(t) \) is chosen, i.e., it satisfies the dilation equation

\[ \phi(t) = 2 \cdot \sum_n h_0(n) \cdot \phi(2t - n) \] (4.6)

then equation (4.2) can be followed. These spaces are embedded and complete [SN97].

The functions are called scaling functions since one can change the size of the subspace spanned by altering the number of \( j \) in the functions. Also, if we define the difference between \( V_j \) and \( V_{j+1} \) as a new space \( W_j \), such that

\[ V_j + W_j = V_{j+1} \quad \text{and} \quad V_j \cap W_j = \{0\} \] (4.7)

that is written as

\[ V_j \oplus W_j = V_{j+1} \] (4.8)

then the functions in \( W_j \) will satisfy

\[ w(t) = 2 \cdot \sum_n h_t(n) \cdot \phi(2t - n) \] (4.9)

since \( W_j \subset V_{j+1} \), and equation (4.9) is called the wavelet equation. Also, the basis
functions for $W_j$ can be written in the form as the basis of $V_j$ in equation (4.5)

$$w_{j,k}(t) = 2^{j/2} \cdot w(2^j t - k)$$ (4.10)

and $w(t)$ is called the mother wavelet. Because every subspace $V_{j+1}$ can be decomposed into a sum of a finer subspace $V_j$ and a wavelet subspace $W_j$, the whole $L^2(R)$ space can be written as,

$$L^2(R) = V_0 \oplus W_0 \oplus W_1 \oplus W_2 \oplus \cdots$$ (4.11)

This is the multiresolution representation of the $L^2(R)$ space in terms of difference information. Any function $f(t)$ in $L^2(R)$ space can be expressed as

$$f(t) = \sum_k c_0(k) \cdot \phi_{0,k}(t) + \sum_{j=0}^{\infty} \sum_k d_j(k) \cdot w_{j,k}(t)$$ (4.12)

If equation (4.12) is truncated so that $j$ equals to a finite number, then a lower resolution representation of $f(t)$ is obtained.

### 4.1.2 Wavelets and Filter Banks

The basic concept of multiresolution analysis has been explained in the previous section, but how can we implement that theory into code? Assume we have a function $f(t)$ in the $L^2(R)$ space, and we project it onto a subspace $V_j$. If the basis set are orthogonal, we can get the coefficients
\[ c_j(k) = \int_{-\infty}^{\infty} f(t) \cdot \phi_{j,k}(t) \, dt \] (4.13)

Similarly, it can be projected onto wavelet space \( W_j \), and the coefficients are

\[ d_j(k) = \int_{-\infty}^{\infty} f(t) \cdot w_{j,k}(t) \, dt \] (4.14)

Substituting equation (4.5) and (4.6) into equation (4.13), we have

\[ c_{j-1}(k) = \sqrt{2} \sum_n h_0(n-2k) \cdot c_j(n) \] (4.15)

A similar relation can be derived in wavelet space

\[ d_{j-1}(k) = \sqrt{2} \sum_n h_1(n-2k) \cdot c_j(n) \] (4.16)

These two recursive equations are the bridge of wavelet and filter banks, since the meaning of these equations are in fact a convolution followed by a down sampling. Actually in image compression, we never have to directly deal with the scaling functions and wavelets. Only the coefficients \( h_0(n), h_1(n) \) in equation (4.6) and (4.9) and \( c(k), d(k) \) in equations (4.13) and (4.14) need to be considered, and they can be viewed as filters and signals respectively [GB92]. The filter banks representation of equation (4.15) and (4.16) is shown in Figure 4-1.
4.1.3 Two Dimensional Filter Banks and Image Compression

For our GIS web services, the input signal is an image, and it can be treated as a two-dimensional array. Every row and column of the array is a one-dimensional data set, so the filter banks can be extended for this case easily by operating on each dimension separately, as shown in Figure 4-2 (L and H refer to the low pass and high pass filters).

The wavelet analysis now decomposes the original space into one average subspace (LL) and three detail subspaces (LH, HL, and HH). This process can be performed recursively on the average subspace. In this manner, a multiresolution representation of the original image is obtained (Figure 4-3).
The filter we used for implementing our system is the Daubechies 9/7 filter, which is widely used in wavelet based image compression and it is also a default filter in the JPEG2000 standard [Website-4]. The Daubechies 9/7 wavelet has 4 vanishing moments, so the low pass subspace contains the exact information up to polynomials of degree 3. For a natural image, most information can be expressed in the LL subspace, and the detail subspaces are generally sparse. The sparse subspaces can then be compressed by many lossless compression algorithms, and this is the basic idea of using wavelet for image compression.

4.1.4 Quantization and Lossless Compression

To further compress an image, a quantization process is introduced. Quantization is a many to one mapping, it represents a number using fewer bits by rounding the less significant digits. It can be easily understood using an example. If we use 8 bits to
represent any integer number between 0 and 255, for instance, 105 will be expressed as “01101001” in the binary form. To quantize this number and use only 2 bits to record it, we divide this number by the length of interval (in this case, we are mapping 256 numbers to 4 numbers; therefore the length of interval is \(255 / 3 = 85\)) and round it to the nearest integer. In the quantized system, the number 105 is expressed by “01”, corresponding to 85 in the original system (Figure 4-4). We can easily see that both 104 and 106 will be mapped to the same number, and we cannot restore the original number after the quantization process. Hence, quantization is a “lossy” process.

For wavelet analysis, the detail subspaces are less important, so representing those regions with fewer bits will reduce the data size without affecting the image quality significantly. For example, if we have a 1024 pixel by 1024 pixel grayscale image (8 bits per pixel), the original size of the image is 1 MB. After performing a three level wavelet decomposition (Figure 4-3), the total size of the 10 subspaces will become 4 MB if we use 4 bytes for each floating point number. As mentioned before, the data in the detail subspaces are quite sparse, and can be compressed very well with any lossless compression algorithm. Thus, the image size generally ends up with a much smaller size even we do not perform a quantization. However, in quantization process, if we use 16 bits for subspace LL3, 8 bits for LH3, HL3, HH3, 4 bits for LH2, HL2, HH2, and 2 bits for LH1, HL1, HH1, then the total size will become 0.36 MB before the lossless
compression. In this example, the quantization process reduces the image to 36% of the original size before the following entropy encoding is performed.

The deflation data compression algorithm is used to perform the lossless data compression in our system. It is based on the LZ77 algorithm and Huffman coding [D96] [Website-5]. This algorithm is used in many general purpose data compression application, and it is built in the Portable Network Graphics (PNG) format as the standard compression method. The block diagram in Figure 4-5 shows the flow of both lossy and lossless image compression.

![Figure 4-5 Lossy and Lossless Image Compression](image)

### 4.1.5 Image Reconstruction

After the wavelet image compression, now we can pass images with much smaller sizes over the Internet. However, the image has to be uncompressed and reconstructed before it can be displayed on the screen. Recall the theory we have derived in section 4.1.2, since
\( V_j = V_{j-1} \oplus W_{j-1} \), the multiresolution decomposition can be written as

\[
\sum_k c_j(k) \cdot \phi_{j,k}(t) = \sum_k c_{j-1}(k) \cdot \phi_{j-1,k}(t) + \sum_k d_{j-1}(k) \cdot w_{j-1,k}(t) \tag{4-17}
\]

Substituting equation (4.6) and (4.9) into (4-17) and integrating will give the synthesis equation

\[
c_j(k) = \sum_n h_0(k - 2n) \cdot c_{j-1}(n) + \sum_n h_1(k - 2n) \cdot d_{j-1}(n) \tag{4.18}\]

That is, we can reconstruct the original fine scale coefficients of the signal from a combination of the scaling function coefficients at a previous coarse level and the wavelet coefficients. In terms of filter banks, a synthesis filter bank corresponding to the one we used in analysis (Figure 4-2) is shown below

![Two Dimensional Wavelet Reconstruction](image)

Figure 4-6 Two Dimensional Wavelet Reconstruction

To summarize, a wavelet decomposition followed by a quantization and a lossless compression process is performed on an image on the server side. An image will end up with a much smaller size but still contain most of the significant information. Then it can
be sent to a user through the Internet efficiently. In the client side, a decompression and wavelet reconstruction process is then performed, and the image can be restored and displayed on the screen, as shown in figure 4-7.

![Diagram of Compression and Decompression Model](image)

**Server Side**

Original Image → Wavelet Decomposition → Quantization → General Data Compression (LZ77 and Huffman coding) → Internet

Decompression → Wavelet Reconstruction → Display

**Client Side**

Figure 4-7 Compression and Decompression Model

### 4.2 Parallel Computing

A web server is usually requested by many people at the same time, and it will be quite time consuming to perform wavelet decomposition and compression on a web server. To reduce the load of server, it is better we can do the image processing in a separate machine. Besides, since we treat an image as a two dimensional array, and each row and column is processed independently during the wavelet decomposition; parallel computing techniques can be applied to the operation effectively.
To do this, first we need to pass images between two systems (in this thesis, we use a desktop with windows XP, IIS 5.1 and .NET framework as our web server, and a Beowulf cluster consists of 16 Linux boxes as the image processing unit), and we have to implement our wavelet compression codes in a parallel manner. In the following sections, we will show you how the TCP/IP sockets and the MPI (Message Passing Interface) standard is used to achieve our goal.

4.2.1 TCP/IP Socket

OSI Seven Layers Model

There are many different protocols that can be used in passing data from one computer to another. The OSI (Open System Interconnection) model defines a networking framework for implementing these protocols in seven layers. The hierarchy and the name of each layer are shown below.

![OSI Model Diagram](image-url)
A message from one computer needs to go through these 7 layers before it can reach the network hardware. In a similar way, messages received at the network level, must go through these same 7 layers before being delivered to another computer. There are many protocols used in each level, for instance, the TCP/IP are the most common layer 4/layer 3 protocols, and the FTP, TELNET, and HTTP are the layer 7 protocols.

In our system, we need to send an image from the web server to the Linux cluster, performing necessary calculations there, and then send the compressed image back to the web server. If we use HTTP to pass the message, an HTTP web server is required on the Linux side (such as Apache) as well as on the Windows side. It is too lavish to setup a web server for this simple function while the Linux system is not a web server at all. On the other hand, TCP/IP is much more lightweight and therefore suitable for our system.

**TCP/IP Socket**

A socket is a loose term used to describe an end point for communication. TCP/IP uses a network address and a service port number to uniquely identify a service. The network address identifies a specific device on the network; the port number identifies the specific service on that device to connect to. The combination of network address and service port is called an endpoint.

![Figure 4-9 Client-Server Sockets](image)

To use TCP/IP protocols, a standard API called Sockets API, also called the Berkeley
Sockets Interface (it was developed at the University of California at Berkeley for the 4.1c release of BSD Unix in the early 1980s) is used. The simplicity of the interface made it common in many other operating system including Linux and Windows.

After we setup a common end point for both client and server, the programs in each side can then send data to or receive data from sockets, i.e., the programs in each side can commute through the sockets.

4.2.2 The Beowulf Cluster and Message Passing Interface Standard

To perform a parallel computing, both hardware and software have to fulfill some basic requirements. A Beowulf cluster and MPI is used in our implementation.

A Beowulf cluster is a virtual supercomputer created by linking numerous inexpensive PCs through network connections into a single high-performance cluster. This cluster can provide comparable performance to a traditional supercomputer at much lower cost. The first Beowulf cluster was built at NASA’s Goddard Space Flight Center in 1994, and it consisted of 16 DX4 processors connected by 10 Mbps Ethernet [Website-6]. The cluster we use for this thesis consists of 16 Linux machines, each has a 1.6 GHz Athlon CPU and 512 MB RAM connected by 100 Mbps Ethernet.

On the software side, the MPI standard is used as the standard to implement our parallel program. Although it is possible for us to pass the messages among processors using TCP/IP or vendor supplied libraries that make use of the underlying communications network hardware, the code implemented in this way is not portable and scalable.
Therefore, attaching to a widely accepted standard is necessary. MPI (Message Passing Interface) is a platform independent library that consists of over 129 functions for message-passing [MPI95] [MPI97]. The specification of the MPI standard is the result of several years of research by groups of interested parties from research institutions, academia and industry. Also, the standard has been implemented by many vendors targeting different systems [Website-7].

4.2.3 The Message Passing Programming Model

When creating a parallel program using MPI we are defining a communication environment in which to perform computation. Within this environment we specify the participating processes of the computation, data representation and some other mechanisms. Each process is an independent thread of control and maintains a unique memory address space. In the message passing programming model, a computation usually include one or more processes that communicate by calling library routines to send and receive messages to other processes. A SPMD (Single Program Multiple Data) model is used for implementing our program. This means while our program is running, a fixed set of processes is created at initialization, one process is created per processor, and every processor executes the same program with different data.

For instance, an image can be split into several strips in one machine, as shown in figure 4-10. Each strip can be treated as an independent process and sent to a different machine, and the wavelet decomposition can be performed in different processors simultaneously. After the decomposition, each processed strip will be transferred back to the original
machine, and the results can be combined together.

Figure 4-10 Splitting an Image into $n$ Strips / processes
Chapter 5  Results and Discussions

5.1 GIS Web Services

GIS Web Services

There are two web services with seven public web methods in our system. Service 1 provides the basic functions such as retrieving data, zooming in a map and locating a specific point on the map. The name and brief description of each public web method is listed in Figure 5-1.

Service 1

The following operations are supported. For a formal definition, please review the Service Description.

- getParameters
  Returns the names of available attribute data in a database

- legend
  Generates proper legend for the selected attribute data

- generatePic
  Draws attribute data on a map and generate a new image

- getData
  Returns all data of a specific parameter

- showPosition
  Locates a point on the map according to the latitude and longitude

Figure 5-1 Basic GIS Web Methods

Service 2 is used for accessing the .NET TerraServer, and the name of each method is shown in figure 5-2.
The following operations are supported. For a formal definition, please review the Service Description.

- terraService
- getBoundary

Figure 5-2 Web Methods for Accessing the .NET TerraServer

The terraService method can be used to compose a new map according to a central point (longitude and latitude), scale (meter per pixel), and the size of image (pixels) from the pieces of map stored in the TerraServer. The getBoundary method is used to determine the coordinates (longitude and latitude) of each corner in the new map, and then one can calculate the coordinates of each point on the map.

Field Study in Newcastle, Australia

A prototype of the GIS web services has been tested with the STEFS project field study in Newcastle, Australia on January 2002 [SARTLK02]. A web server with the web services and web application was running in MIT in Cambridge, while the research teams were working in the south hemisphere. All data were transferred back to Cambridge live through a GSM data enable phone and stored in a SQL server database. One could retrieve the cartographic information through the web services with almost no time lag. Figure 5-1 is a screenshot taken from the prototype.
However, the feature of generating a suitable map on demand was added after the field study, and the background of figure 5-3 is fixed and pre-stored in the web server.

Three Schemes: Web Application, Windows Application, and Windows Form Control

Three different schemes for accessing the GIS web services have been discussed in chapter 3. Table 5-1 gives a comparison of these methods.

<table>
<thead>
<tr>
<th></th>
<th>Web Application</th>
<th>Windows Application</th>
<th>Windows Form Control</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Browser</strong></td>
<td>Any</td>
<td>Not required</td>
<td>IE 6 or higher</td>
</tr>
<tr>
<td><strong>Additional Program</strong></td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>Image Compression</strong></td>
<td>Not available</td>
<td>Available</td>
<td>Available</td>
</tr>
<tr>
<td><strong>.NET Framework</strong></td>
<td>Not required</td>
<td>Required</td>
<td>Required</td>
</tr>
</tbody>
</table>

Table 5-1 Comparison of the clients of the GIS web services
5.2 Wavelets Image Compression

Define the compression ratio CR as:

\[
CR = \frac{\text{original size}}{\text{compressed size}}
\]  

(5.1)

Then a larger number of CR indicates a better compression.

The size of a 480 pixels by 360 pixels full color (16777216 colors, 24 bits per pixel) uncompressed image is \(480 \times 360 \times 24 = 4,147,200\) bits (506KB), and if we apply a lossless compression to this image, usually we can get a CR around 1.2 to 2, depending on the actual content of the image. The image shown in figure 5-1 has a CR = 1.85 (using the deflation algorithm).

![Original Image (Charles River)](image)

Figure 5-4 Original Image (Charles River)

We separate the image into 3 color channels (red, green, blue), and the color intensity of each channel is from 0 to 255 (8 bits per channel). The intensity distributions are shown
Applying a single stage decompose to the image, and quantizing the LL subspace using 8 bits, and LH, HL, and HH subspaces using 2 bits, one can obtain the following result:

Also, the intensity distributions of the red channel on each region are shown below:
Figure 5-7 Intensity Distribution in the Red Channel (X: Intensity – Y: Percentage)

Because there are a lot of null values in the LH, HL, and HH channels, those subspaces can be compressed very well. The results are shown in table 5-2:

<table>
<thead>
<tr>
<th></th>
<th>Size (KB)</th>
<th>CR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Original</td>
<td>Compressed</td>
</tr>
<tr>
<td>Original</td>
<td>506</td>
<td>275</td>
</tr>
<tr>
<td>LL</td>
<td>127</td>
<td>105</td>
</tr>
<tr>
<td>LH</td>
<td>31.6</td>
<td>1.22</td>
</tr>
<tr>
<td>HL</td>
<td>31.6</td>
<td>2.63</td>
</tr>
<tr>
<td>HH</td>
<td>31.6</td>
<td>0.95</td>
</tr>
<tr>
<td>Reconstructed</td>
<td>506</td>
<td>110</td>
</tr>
</tbody>
</table>

Table 5-2 Single Stage Decomposition Result
Roughly one can compress the image to 22% of the original size, which is much better than compressing the original image directly (54%). If one performs a two stage decomposition, quantizes the LL2 subspace using 8 bits, and the other level 2 subspaces and level 1 subspaces using 3 bits and 2 bits respectively then a better result can be obtained:

<table>
<thead>
<tr>
<th></th>
<th>Size (KB)</th>
<th>CR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Original</td>
<td>Compressed</td>
</tr>
<tr>
<td>LL2</td>
<td>31.6</td>
<td>26.9</td>
</tr>
<tr>
<td>LH2</td>
<td>11.9</td>
<td>2.03</td>
</tr>
<tr>
<td>HL2</td>
<td>11.9</td>
<td>2.94</td>
</tr>
<tr>
<td>HH2</td>
<td>11.9</td>
<td>2.67</td>
</tr>
<tr>
<td>LH1 + HL1 + HH1</td>
<td>94.9</td>
<td>4.80</td>
</tr>
<tr>
<td>Reconstructed</td>
<td>506</td>
<td>39.3</td>
</tr>
</tbody>
</table>

Table 5-3 Two Stage Decomposition Compression Result

The compressed image is 7.8% of the original size. Similarly, one can perform a four stage wavelet decomposition and keeps 8 bits per pixel for the LL4 subspace, and keeps 5 bits, 4 bits, 3 bits and 2 bits for the other level 4, level 3, level 2 and level 1 subspaces respectively, and then an image with only 4% of the original size can be obtained (table 5-4).
The compressed image is only 4.2% of the original size after a four stages lossy wavelet compression. The results are shown in figure 5-8.

Table 5-4 Four Stage Decomposition Compression Result

<table>
<thead>
<tr>
<th></th>
<th>Size (KB)</th>
<th>CR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Original</td>
<td>Compressed</td>
</tr>
<tr>
<td><strong>LL4</strong></td>
<td>2.02</td>
<td>1.70</td>
</tr>
<tr>
<td><strong>LH4 + HL4 + HH4</strong></td>
<td>3.79</td>
<td>2.49</td>
</tr>
<tr>
<td><strong>LH3 + HL3 + HH3</strong></td>
<td>11.9</td>
<td>4.48</td>
</tr>
<tr>
<td><strong>LH2 + HL2 + HH2</strong></td>
<td>35.6</td>
<td>7.65</td>
</tr>
<tr>
<td><strong>LH1 + HL1 + HH1</strong></td>
<td>94.9</td>
<td>4.80</td>
</tr>
<tr>
<td><strong>Reconstructed</strong></td>
<td>506</td>
<td>21.1</td>
</tr>
</tbody>
</table>

Figure 5-8 Quality Comparison
Although the RGB color space is easy to be understood, it is not a suitable color space for image compression. On the other hand, the YUV color space separates an image into a luminance channel (Y) and two chrominance (U and V) channels (Chrominance is the difference between a color and a reference white at the same luminance).

\[
Y = 0.299 R + 0.587 G + 0.114 B
\]
\[
U = 0.492 (B - Y) \quad (5-2)
\]
\[
V = 0.877 (R - Y)
\]

Since human eyes are much more sensitive to the luminance channel than the chrominance channels, the components in U and V channels can be compressed further.

If we separate the sample image into RGB and YUV channels, the following results can be obtained:

![Figure 5-9 Separating an Image into RGB Channels](image1)

![Figure 5-10 Separating an Image into YUV Channels](image2)

It's hard to say which channel is more important in the RGB color space; however, the
details in the color difference channels (channel U and V) are not very clear, i.e., if we use fewer bits to represent the components in the channel U and channel V, it is not easy to be observed by human eyes.

An 8 KB image can be obtained if we perform a four stage wavelet decomposition in the YUV color space similar to what we did in the RGB color space, but use 1 bit less in representing every subspace in channel U and V. If we use the same quantization scheme in RGB color space (using fewer bits in channel G and channel B), a 10 KB image can be obtained but the image quality is much worse than processing in the YUV color space.

![Image Compression Results in YUV (8KB) and RGB Color Space (10KB)](image)

In summary, the more stages of wavelet transforms performed on an image, the higher compression ratio one will get. This is because the more number of wavelet transforms is used, the more high-pass information is extracted out from the original image. Since high-pass subspaces are mostly sparse, a high CR can be achieved. Besides, many details are thrown away during the quantization process, and a higher CR can be obtained in the lossy compression. However, the more high-pass data is dropped, the more blur one will get from the reconstructed image.
YUV space provides better access than RGB color space when dealing with compression. This is because human vision is not very sensitive to color differences. Thus U and V spaces can be further compressed yet our eyes won't notice the changes.

There is one assumption used in the wavelet transform when dealing with the boundary - assuming images are symmetrically extended. Due to this assumption used in the module, there are some errors that exist near the edges of the reconstructed image, and these errors will accumulate if the level of wavelet transform is increased. Although one can have a larger CR if higher level wavelet decomposition is performed on an image, the level has to be restricted in a reasonable range.

5.3 Speed-up by Parallel Computing

The speed-up by parallel computing can be defined as follow:

\[
Overall \ speedup = \frac{Time \ for \ 1 \ process}{Time \ for \ n \ processes} \quad (5.3)
\]

We only care about the parallelized part of code, therefore the net speedup, S is

\[
S = \frac{T_{\text{serial}}}{\frac{T_{\text{serial}}}{n} + T_{\text{comm}}} \quad (5.4)
\]

For an ideal system, the time spent on communication is much less than the time spent on calculation, i.e., \( T_{\text{comm}} \ll T_{\text{serial}} \), so ideally the speedup is n if we have n processor.

Figure 5-12 shows the results of performing parallel wavelet decomposition on a 0.66
MB image and a 8.8 MB image. In both cases, the image is split into several strips and each strip is processed by an independent processor.

Because the more processors are used, the more time is spent on the $T_{comm}$, we cannot actually achieve the ideal speed-up. However, the result is acceptable. Roughly less time is needed if more processors are used. The irregularity of the result depends on the environment variability such as hardware interrupts, hard disk speed, and collisions on the interconnection network.

Figure 5-12 Speed-ups by Parallel Computing
6.1 Summary

1. A web based GIS is ideal for sharing and displaying geographic data live on the Internet, and the system can be easily scaled since it is implemented as web services. Also, these web services can be reused by other programmers, so they can build new systems on top of our services.

2. Web services are the future of distributed computing. In this thesis, we not only build new web services to fulfill our objectives, but also take advantage of the web services provided by other people (the TerraService).

3. A web application can be executed using almost any browser since it is running on the server side and sends only standard HTML and scripts as output to the browser. That means users can access our web services in almost every platform without installing any additional program or plug-ins.

4. To reduce the time in data transmission, image compression is introduced into the system. By performing wavelet compression, one can highly compress image data before it is sent through the Internet. Also, a .NET windows application is used in the client side to decompress and display the image. Alternatively, a windows application can be wrapped as a windows form control and executed inside Internet Explorer 6 or higher. By doing this, one can use the program through a browser without manually installing an application.
5. For a GIS, usually the attribute data are more important than the geographic data, i.e., higher compression ratios with acceptable image quality are usually preferred. A very high compression ratio can be achieved by performing lossy wavelet compression on an image, and the image quality is usually better than the JPEG format with the same compression ratio.

6. For a long filter such as the Daubechies 9/7 we used in this thesis, performing wavelet decomposition is time consuming. Because image processing is inherently parallel, parallel computing is introduced into the system naturally. A speed-up of 2 on performing wavelet decomposition can be obtained if 3 processors are used, and a better result can be obtained if more processors are used.

6.2 Future Work

1. Adding security consideration into the GIS web services.
   For a prototype like the present system, security is not an important consideration. However, if the system is going to be scaled up, one should always keep the security issues in mind. For example, in the web services server, one can deny the HTTP get and HTTP post methods, so people can only access the web services through the SOAP protocol, and reduce the risk of “denied of service” attack. Further, inside the web services, one could add a SOAP header to accept requests only from authorized clients.

2. Caching the images in web server.
   The ability to compose a suitable map on the fly is good; however, it is quite time
consuming since the web service has to fetch pieces of map from the terraServer. If we cache the frequently requested maps in the web server, the performance can be improved considerably.

3. Using the lifting scheme

A convolution based polyphase scheme is used in implementing the wavelet decomposition. However, a lifting scheme is a better way to perform the wavelet transformation if the performance is crucial.
Bibliography

- [TW91] Coppock J.T. and Rhind D.W., "The History of GIS". In Maguire D.J.,