WASHINGTON UNIVERSITY
SEVER INSTITUTE OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE

DESIGN AND PERFORMANCE ANALYSIS OF A DISTRIBUTED
IMAGE SPACE NAVIGATOR

by
Kamal Bhatia, B.Tech

Prepared under the direction of Professor Jonathan S. Turner

A thesis presented to the Sever Institute of Washington University in partial fulfillment of the requirements for the degree of

Master of Science

August, 1997

Saint Louis, Missouri
Imaging applications in medicine and satellite surveillance require viewing high resolution images that are too large for presentation on current and near term displays. Systems for viewing such images must operate over high speed networks and support mechanisms for smooth panning and zooming using a distributed client-server framework. Applications supporting these mechanisms need to effectively utilize intelligent caching and presentation techniques and allow distributed multiuser navigation. This thesis explores an approach to the design of such systems called the Image Space Navigator (ISN).

The design issues for a general architecture of the system and implementation of a medium performance prototype based on standard workstations and networks are discussed. Also presented are strategies for organizing and presenting image data for rapid display and real time navigation. Description of the software prototype including the remote viewing tool (ISN Viewer) and distributed image server are presented and a mechanism for collaborative image viewing in multicast groups is described. The thesis describes the structure and design of the distributed image service and presentation modules using the Playground application design environment. State space models of the client and server are presented and evaluation of performance parameters such as network traffic and delay bounds for smooth panning and zooming over high speed networks are discussed.
To my parents and brothers
for their constant support and admiration
and the almighty lord
Contents

List of Tables ......................................................................................... vii

List of Figures ......................................................................................... viii

Glossary ................................................................................................. x

Acknowledgments .................................................................................... xiii

1 Introduction ........................................................................................ 1

1.1 Motivation ....................................................................................... 1

1.2 Components of the Image Space Navigator ....................................... 3

1.3 Basic User Interface (ISN Viewer) .................................................... 4

1.4 Examples ......................................................................................... 6

1.4.1 Geographic Map Navigation ......................................................... 6

1.4.2 Study of Medical Images .............................................................. 7

1.5 System Design Issues ...................................................................... 8

1.5.1 Estimate of Network Bandwidth and Cache Size ....................... 8
1.5.2 Space and Cost Requirements for Large Images ........................................ 9
1.5.3 Caching at Server and Clients ............................................................... 10
1.5.4 Server Processing Requirements ........................................................... 10
1.5.5 Network Bandwidth ................................................................................ 10
1.6 Related Work ............................................................................................ 11
1.7 Overview .................................................................................................... 11

2 Organization and Retrieval of Image Data .................................................... 13

2.1 Multilevel Data Representation on Disk ...................................................... 13

2.1.1 Multiresolution Image Pyramid and Tiles ............................................... 15

2.1.2 Server Storage Subsystem for Prototype ............................................... 16

2.2 Data Caching at Server ............................................................................. 16

2.2.1 Just-in-Time Delivery of Image Tiles ...................................................... 18

2.3 Data Representation at the Client .............................................................. 19

2.3.1 Example of Incremental Window Redrawing ........................................ 20

2.4 High Speed Network Support .................................................................... 21

2.5 Related Work ............................................................................................ 22

3 Software Design of ISN ............................................................................... 23

3.1 ISN Viewer User Interface ........................................................................ 23

3.2 Playground Distributed Design Environment .......................................... 25

3.2.1 Playground Design Model .................................................................... 25
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.8.2 Disk Section</td>
<td>45</td>
</tr>
<tr>
<td>4.8.3 Server Application Section</td>
<td>45</td>
</tr>
<tr>
<td>4.8.4 Network Section</td>
<td>46</td>
</tr>
<tr>
<td>4.9 Simulation Results</td>
<td>46</td>
</tr>
<tr>
<td>4.10 Summary</td>
<td>48</td>
</tr>
<tr>
<td>5 Conclusion and Future Directions</td>
<td>49</td>
</tr>
<tr>
<td>5.1 Collaborative Image Viewing</td>
<td>49</td>
</tr>
<tr>
<td>5.1.1 Software Structure</td>
<td>51</td>
</tr>
<tr>
<td>5.2 High Speed Image Data Transfer using APIC and ATM</td>
<td>51</td>
</tr>
<tr>
<td>5.3 Priority Queues within ISN Server</td>
<td>52</td>
</tr>
<tr>
<td>5.4 Conclusion</td>
<td>52</td>
</tr>
<tr>
<td>Appendix A Color Figures</td>
<td>53</td>
</tr>
<tr>
<td>Appendix B ISN Viewer Reference Manual</td>
<td>56</td>
</tr>
<tr>
<td>B.1 ISN Control</td>
<td>56</td>
</tr>
<tr>
<td>B.2 Mouse and Key Controls</td>
<td>56</td>
</tr>
<tr>
<td>References</td>
<td>59</td>
</tr>
<tr>
<td>Vita</td>
<td>63</td>
</tr>
</tbody>
</table>
List of Tables

4.1 Range and Typical values for the Parameters ....................... 35
4.2 Variation of Network and Server Bandwidth with Number of Clients ... 41
4.3 Summary of Results ........................................... 43
4.4 Observations from Time Log for Single User Navigation ............. 43
4.5 Simulation Parameters ........................................ 45
4.6 Simulation Results (N = 8) .................................... 46
4.7 Variation of Delay with Network Speed ............................ 47
4.8 Variation of Delay with Navigation Parameters ..................... 47
4.9 Variation of Delay with Server Latency ............................ 48
4.10 Variation of Delay with Disk Hit Rate ............................ 48
# List of Figures

1.1 Variation of Display Cost against Resolution .................................. 2  
1.2 Components of the Image Space Navigator ....................................... 3  
1.3 ISN Viewer User Interface with Component Description ....................... 5  
1.4 Navigating through a Geographic Map (SF Bay Area) .......................... 6  
1.5 Collective Radiological Study of MRI Images .................................... 7  
1.6 Data Requests for a Pan Operation within Image Space ....................... 9  
2.1 Multilevel Representation of Images on Disk ................................... 14  
2.2 Structure of the Image Pyramid ..................................................... 15  
2.3 Server Storage structure for Image data ......................................... 17  
2.4 Incremental Update of Zoom Window using Image Tiles ....................... 19  
3.1 Elements of the ISN Viewer and User Interface ................................ 24  
3.2 Overview of Playground Connections for ISN ................................... 27  
3.3 Connection Oriented ISN Interface for Client and Server ..................... 28  
3.4 Navigation Cycle with Window States ............................................ 31
4.1 State Transition Diagram for the Client (Image Viewer) .... 34
4.2 State Transition chain for Move states .................. 36
4.3 Variation of Bandwidth with Transition Probability .......... 39
4.4 Variation of Bandwidth with State Holding Time \( (p_m=0.8, s=512) \) ... 40
4.5 Probability Distribution of Number of Clients in Move state .... 41
4.6 Probability (Log Scale) of Number of Clients in Move state ..... 42
4.7 Server Simulation Model .................................. 44

5.1 Collaborative Image Viewing ............................... 50

A.1 San Francisco Bay Area .................................... 53
A.2 Medical Imaging ...................................... 54
A.3 High Quality Digital Newspaper .......................... 54
A.4 Satellite Imagery ...................................... 55

B.1 Elements of ISN Controller .............................. 57
Glossary

API - Application Programmer Interface.

Application Management - Coordination of information used for end-user documentation, informed module launching and configuration of distributed applications.

ATM - Asynchronous Transfer Mode. Data communication protocol for connectionless high speed data transfer over networks using packet cellification and virtual circuits.

APIC - ATM Port Interconnect Controller. Dedicated hardware controller with interface to connect computers and peripherals directly to ATM networks with direct memory access and I/O control for high speed data transfer.

CSA - Client Server Application

Client - Refers to end user application or process running in a distributed environment. In our case refers to the viewers in ISN and the user application that communicates with the server by sending messages and requesting data for image display.

DMA - Direct Memory Access.

DSP - Digital Signal Processors.

Distributed Application - An application consisting of multiple concurrent processes usually running on separate workstations that communicate over the network. In playground and ISN, an application is the client or the server consisting of independent modules with logical connections among published variables.

Eavesdropping or Group Viewing Mode - A mode of image viewing where two or more clients connect and establish a link with another client to share the same image space.
This establishes a master and slave relationship, with the master being the client connected to the server controlling the data requests and state of the common view of the image.

I/O Abstraction - A connection oriented model of interprocess communication in which independent variables interact with an abstract environment to maintain consistent data states.

ISN - Image Space Navigator.

ISN Viewer - The user interface for navigating the image space using pan and zoom windows. Also refers to the associated menu control for connections to a server, requesting specific file data and switching between single and group viewing modes.

GUI - Graphical User Interface.

Launcher - A playground API to automatically configure and start modules based on their identity and connection information published by the application.

Logical Connection - The communication specification between two published variables. Connections can specify either bidirectional or unidirectional communications.

Module - A component of a distributed application with certain attributes, as published variables or registered functions visible to the environment.

Multicast Mode - Mode of operation in which data is sent out to multiple connected modules or clients from a single source simultaneously.

PACS - Picture Archive and Communication System.

Playground - A set of software tools for writing interactive distributed applications, by providing an abstraction to separate the concerns of low level communication protocols from programming language and application implementation.

Published Variable - A data structure, such as integer or array, which is exposed from the module to the external environment. In playground, published variables connected to other similar variables are updated when one of them changes via the logical connection.

RPC - Remote Procedure Call.
SHD - Super High Definition. Refers to the class of high quality digital images and video of resolution above 2048x2048 pixels and frame rates above 30 fps.

Server - The repository computer which maintains a data store to service requests from clients. Typically, contains an archive or maintains an image database and continuously runs a daemon looking for external requests and servicing data back to connected clients.

View Window - Refers to the current or selected window for panning and zooming within the base image. The idea is to focus on a particular window for "viewing" at one time although multiple windows might be present.

Virtual Mouse - State of the screen pointer when it is shared with between member clients of a multicast group. The actual state is maintained on the server in the common memory space and copies sent out to each connected member for updating individual states.

Window - Refers to the region of image confined within a square area on the screen of the viewer. For the user interface, size of windows are either 512x512, 256x256 or 128x128 pixels.
Acknowledgments

Professor Jonathan Turner has been an excellent advisor, providing insight, advice and innovative ideas for enhancing the overall quality of this thesis and the software system. It was a very interesting insight Jon had about designing such a system based on our initial experience with the SHD system. Jon has been always available for guiding me in the right direction and acted as the mentor to bring the idea to practice. Prof. Jerry Cox and Prof. Jim Blaine have been instrumental in providing constant feedback and excellent suggestions.

I wish to personally thank Tatsuya Fuji, who guided me very well both as a fellow researcher and friend during his stay at Washington University. Tatsuya’s initial tutorial help on Xwindows and Xlib were the founding stones for the design of the viewer. I would like to thank my teachers at Washington University, Prof. Guru Parulkar, Prof. Mark Franklin, Prof. Douglas Schmidt, Prof. Paul Min, Prof. Robert Rouse and Prof. Donald Snyder who introduced me to the subtle and interesting principles in networking and distributed system design and analysis. My teachers at IIT, Kanpur deserve special gratitude for laying the foundations without which I would never have completed my work here. I personally wish to thank Prof. Joseph O’Sullivan for giving me the opportunity to attend Washington University initially.

I wish to thank Joe Hoffert for ideas about the connection manager and Paul McCartney for insights from his user interface design experience. Special thanks to my friends, Srihari and Inder and my roommates, Paritosh and Navin who provided me with the necessary subsistence and late night school commute support to complete my work in time. Myrna Harbison deserves special thanks for fitting me in Jon’s busy schedule whenever I needed to in time. Finally thanks to Vivin for arranging our initial meetings and providing medical images for testing the system.

My foremost and deepest respect to my parents, Subhash and Ratna, who have been instrumental in guiding me to the best in life and providing immeasurable love and support all the way. My brothers, Pankaj and Ankit, have always encouraged me and provided deep affection which has guided me to a successful and very satisfying educational career. No
words can express my heartfelt gratitude and reverence for my parents and my grandmother, Laxmidevi for all their sacrifices for me.

This research was supported by a grant from Nippon Telephone and Telegraph Company, Japan under the Visiting Researcher program in the Advanced Networking Group of Applied Research Lab at Washington University.

Kamal Bhatia

Washington University in Saint Louis

August 1997
Chapter 1

Introduction

In medical imaging and satellite imagery applications, the problem of storage and fast retrieval of high resolution images is an important one. With the advent of faster computers and larger and cheaper memories, such large image applications have become practical and it has become easier to implement such systems in an economic manner. The support from high speed networks has provided the necessary backbone to implement applications of such high bandwidth nature in an efficient manner. However the display technology for large resolution screens has not advanced enough for displaying large images (i.e. 4096x4096 pixels or more) in a cost effective manner. To solve the problem, we present an approach for viewing high resolution images using normal workstations in a distributed environment using high speed networks.

The objective of this thesis is to explore the design of systems for viewing large images on limited resolution displays in a networked environment. A particular system, called the Image Space Navigator (ISN) was developed as a proof of concept and as a vehicle for examining the issues associated with the design of such systems.

1.1 Motivation

Simple examples of high resolution images includes digital maps or geographic images and high resolution medical images. Advances in the fields of high quality digital imagery and high resolution layered satellite data has brought with them an increasing need for managing and viewing large images. Applications in medicine require viewing images at the finest
resolution and the process of acquiring and storing with very fine grain resolution has been made possible with high quality digital scanning equipment. Typically, mammographic or X-Ray images are digitally stored with resolutions of approximately 70 Kpixels/square-inch in order to detect fine physiological features of interest. Maps and geographic data require large resolutions in order to map large physical areas while maintaining information about individual entities. For example, a map of the St. Louis metro area, an area of about 80km x 80km would require a resolution of 8Kx8K to provide a 10 meter resolution.

![Graph showing relationship between display cost and image format](image)

Figure 1.1: Variation of Display Cost against Resolution

Advances in display technology have been slow to develop, compared to development of processing systems and networks. Further, the focus of display technology development has been on improving representation characteristics of displays such as flicker rate or color mixing rather than developing cost effective and scalable displays.

Figure 1.1 shows the relationship between cost and display size. The cost of a system for viewing images of size 2048x2048 pixels is around $43,000 [11]. This suggests that high resolution displays are too costly for routine use, and in any case lack the resolution needed for maps and satellite imagery. In addition to display costs, data storage and network bandwidth are also constraining factors in the management and presentation of
large images. However, the rapid improvement in the cost of disk storage and network bandwidth makes these less difficult to deal with.

This thesis tries to address the problem using an image viewing GUI. The idea is to provide interactive and responsive control of images or an image viewed at different resolutions. The next section describes the basic structure of the Image Space Navigator.

1.2 Components of the Image Space Navigator

Figure 1.2 shows the components of the Image Space Navigator using a server machine, high speed network and image display units.

Figure 1.2: Components of the Image Space Navigator
The main image viewing tool or application is called the ISN Viewer. The viewer provides essential user interface functions including mechanisms to allow the user to conveniently view very large resolution images on limited resolution displays. The remote image server or ISN server is an application running on a remote computer which services requests from the clients across the network. The Image files are stored on the server's local disk. The disk (or possibly a tape drive for larger image archives) stores the images in an indexed manner for speedy retrieval. To provide rapid response to user requests, both the viewer and the server include caches to hold image data in anticipation of requests. These will be discussed further in chapter 2.

1.3 Basic User Interface (ISN Viewer)

The user controls the system using the ISN Viewer. The basic paradigm for image viewing in the ISN application is to have a base image which fills the screen and shows a low resolution version of the entire image. Higher resolution zoom windows can be created within this background image and dragged across the image using the mouse. The effect is like moving a magnifying glass over a printed image. The key performance challenge for the system is to support smooth real-time panning with no pauses or interruptions due to network or processing delays. The background image provides context for the user and helps him or her understand the relative position of the zoom windows within the overall image space. The user interface for the viewer also allows the user to select a remote image server and then select an image file to view from the server’s database.

Figure 1.3 shows a snapshot of the ISN Viewer GUI with several overlapping zoom and pan windows of various sizes. In this example the user is viewing a digital newspaper and it can easily be seen that the original font at screen size resolution is barely readable, whereas the zoom windows provide for a very nice see-through lens to clearly read the print and also selectively view areas of interest.

By clicking on the left mouse button, the user is able to zoom one level down or in (to higher resolution) and by clicking on the right mouse button, the user can go up or out one level. The window position is tied to the cursor or the mouse and as the user slides the mouse the window follows it smoothly to the new position. The action for resizing and zooming is controlled by using the mouse keys and the arrow keys on the keyboard.

The important thing is to maintain responsive behavior when the user pans through the image or zooms in to higher or lower resolution data. Since the data has to be obtained from
Figure 1.3: ISN Viewer User Interface with Component Description
a remote server, the important issue is to send minimal size requests for data so that data is available "in time" for redrawing at the new position within the window. Also requesting data very frequently from the server for small data sizes is not advisable because of the minimum time involved in sending and receiving data from the remote server.

![Map of Hayes Valley and San Francisco Bay area](image)

Figure 1.4: Navigating through a Geographic Map (SF Bay Area)

1.4 Examples

This section presents some of the applications which would use ISN tool and the key techniques implemented in the ISN system.

1.4.1 Geographic Map Navigation

One of the most important applications for large image viewing occurs for geographical map navigation. The following example shows a road map example of a small portion of the San Francisco Bay area. It can be seen that for a relatively small size image (4096x4096 pixels),
the information density is very high. Clearly for any relevant examination of the details, a high resolution subset of the map larger than the screen size needs to be viewed.

Figure 1.4 shows the map of the upper bay area of San Francisco City. The multiple overlapping windows provide different resolution views of the map in windows of 512x512 and 256x256 pixels. One of the windows shows the finest resolution view at 4096x4096 pixels and the road and the street details are very clear. The intermediate resolution windows at 2048x2048 pixel resolution shows an overview of the neighborhood and specific features as parks, bridges etc. This approach of viewing the map is much more informative than the traditional zoom and view approach implemented in map viewers as MapQuest [23] and Yahoo [24].

![MRI Images](image_url)

Figure 1.5: Collective Radiological Study of MRI Images

1.4.2 Study of Medical Images

Medical images such as MRI and CAT are acquired at very high resolutions and the process of digitization leads to image sizes of around 3000x3000 pixels. The challenge is to view and study a group of images together to extract relevant physiological information from it. The problem is further complicated by the fact that medical images need to be interpreted.
in collective groups and particular attention needs to be given to detailed patterns in the images.

Figure 1.5 shows a matrix of MRI data of the human brain for a radiological study obtained from the SHD-NTT data bank. The actual images have already been subsampled to create a mosaic of 2048x2048 pixels. It can be seen that the anatomical features of some of the images are visible only within the higher resolution windows. Since the application supports multiple windows the user can create multiple zoom windows and do a comparative study of the images. This interface is simpler and more intuitive than the interface available in complex image handling tools for viewing medical data.

1.5 System Design Issues

The basic system design issues relate to efficient storage and retrieval of images and using the network bandwidth for responsive interactive navigation of images.

1.5.1 Estimate of Network Bandwidth and Cache Size

A typical navigation scenario involves approximately 6 to 16 users, normally with workstations or PC's which have screen resolutions of about 1180x980 pixels. For purpose of illustration and simplicity of analysis all original screen sizes shall be taken as 1024x1024 pixels, assuming a depth of 1 byte/pixel giving us 1 MB of raw image data for the original screen. The largest zoom/pan window is 512x512 pixels and to allow smooth panning through the image, a screen size (i.e 1024x1024) neighborhood of image data centered around the current window is cached locally on the client. This means that the client caches another 1 MB of data per zoom window. Assuming upto 5-6 independent pan and zoom windows for each user, the client would require 5-6 MB of extra memory for cache.

Figure 1.6 shows an example of the pan operation performed by the user within an original image of 16Kx16K pixels. Let us assume that the user takes about 1 second to pan a zoom window at the highest resolution to the upper right corner of the screen from the lower left hand corner of the screen. This is a fairly demanding example, but is useful for gaining an understanding of the performance requirements for the system. The diagonal shaded area in the figure contains about 31.25 MB of data and so the panning operation could require the transfer of data at rates of about 250 Mb/sec. More typically, we could expect users to pan at rates less than 1024 pixels/sec (rather than 16x1024 pixels per sec) so typical data
Figure 1.6: Data Requests for a Pan Operation within Image Space

rates for panning users will be under 20 Mb/sec. For more ambitious applications, with images of size say 64Kx64K bandwidth goes up by factor of 4, and for larger screens as in the SHD system, bandwidth also goes up by a factor of 4.

1.5.2 Space and Cost Requirements for Large Images

The limited speed of accessing information from disk and the size of semiconductor memory are major constraints in the design of large data access programs. In most applications, retrieving image data is often a time consuming task and the problem can be partly solved by advanced memory management techniques. The storage costs for disks is within reasonable limits to store large images. Disks are now a few hundred $/GB so a 16Kx16K image be stored for about $100 on-line. The more important factor is the disk access speed on the server for data I/O. The disk access rates of 10-20 ms and data transfer rates of 10-20 MB/sec creates a bottleneck for servicing clients within a given time delay. To service a
2Kx2K block of image data via disk I/O requires over 400 msec which might not be tolerable in certain instances.

1.5.3 Caching at Server and Clients

The user should be able to smoothly pan and zoom within the image space irrespective of size of the image. The redraw and data structuring operations are performed locally to the display for rapid interactive response, with incremental on-demand data retrieval from the server. Caching at the server reduces the service time since no disk I/O is needed when data is accessed from the cache. Caching at the client helps in smoother navigation since most panning requests can be handled locally, without waiting for the arrival of data from the server. Typical cache sizes at the client would be 1-2 MB (for a 1024x1024 displays, 1 MB of cache stores a full screen of image data) and we could expect the server cache to be 20-30 times larger since it would be servicing multiple clients. It would be reasonable to have four times as much cache on the server compared to the client (i.e 4 MB of cache is used per client when the client cache is 1 MB in size).

1.5.4 Server Processing Requirements

The efficiency of navigation is tightly coupled to delays in server processing time including time for data I/O and time required to send data to clients. The server also has to process image data into proper data structures and this requires processing cycles at the server.

1.5.5 Network Bandwidth

As in most distributed applications, network bandwidth plays an important role in the implementation and performance of the application. In the case of ISN, the network bandwidth and latency are of primary concern for rapid response for navigation. The factors affecting the bandwidth would be the maximum number of users requesting data and the number of different servers using the same network path.
1.6 Related Work

One of the approaches presented in the literature (Lieberman [20] and Donelson [10]) is to display the various levels of an image on the whole display at one time. The problem with this is that this causes the user to lose the context of the overall image and limits views to a single region of interest. Typical applications in medicine (Gee [12]) require simultaneously viewing a collection of images and contrasting features of interest to extract relevant information. Lieberman [20] and the TerraVision group [19] present an important step in solving the problem by allowing dynamism in viewing images. The approach is to have control over any portion of the image and being able to do so in a fast and interactive manner. This translates to supporting smooth or real time panning in applications dealing with static images.

Dal Degan [8] identifies and describes some of the issues for implementing a distributed imaging system with high data bandwidth requirements. Lowe [22] states that the main concern for large image applications is not the compression algorithm since the actual implementations offsets the efficiency of good compression and storage. In actual implementations of image processing applications, 70% to 90% of the image retrieval time is taken up by the network kernel switching, Xwindows [16] communications and other data packing tasks for network transfer which are independent of the specific image representation. This means that any implementation of an ISN not only needs to consider the limitations due to physical elements such as network and cache sizes but also look very carefully at the software system and must minimize data switching and presentation times. Ligier et al. [21] explains, for distributed PACS, the need for developing the imaging application in a modular architecture so that there are no explicit dependencies on hardware or certain physical systems or particular networks.

1.7 Overview

The remaining chapters of this thesis are arranged as follows. Chapter 2 discusses the organization and storage of image data for the server and the client. Described in this chapter are multi-level data tiling of image data on disk, data representation on client for zoom and pan windows, data caching on the server and block segmentation of image for efficient incremental updates. Chapter 3 discusses the software design of the image space navigator including ISN Viewer and ISN Server. The focus of the chapter is on software
design of the server and client, design of the ISN viewer interface, intermodule communication using Playground and the navigational mechanisms supported by the ISN viewer. Chapter 4 discusses the performance of the ISN prototype and quantitative client and server models. Included in this chapter are the client model of navigation and a simulated server model. Network bandwidth requirements and delays are evaluated and compared with the values obtained for the prototype implementation. Chapter 5 concludes with a summary of contributions and discussion of future research directions. A Reference manual for the ISN Viewer is presented in the Appendix.
Chapter 2

Organization and Retrieval of Image Data

This chapter presents the specific image storage and retrieval strategies implemented for the Image Space Navigator. The performance of the ISN is strongly coupled to the management and retrieval of image data and caching implemented in the client server system. While most implementations of large image handling systems are implemented using some kind of database software, consideration of image database design has been omitted here since our focus is on network and data representation issues (using caching and disk storage) rather than queries or searches within large images.

2.1 Multilevel Data Representation on Disk

Disk storage allows construction of image servers capable of holding large amounts of data (tens or hundred of gigabytes). Unfortunately, data retrieval from disk is slow with typical access times of 10-20 ms and transfer rates of 10-15 MB/sec. When data needs to be delivered to the clients, the disk I/O speed can become a bottleneck.

To allow rapid retrieval of image data, images are stored at all resolutions and are stored without compression. This is illustrated in Figure 2.1. The multiple levels of images are stored in a logical fashion from screen size (i.e. 1024x1024 pixels) to the size of the original image. For example, for an original file of size 16Kx16K the directory contains image files 1024x1024, 2048x2048, 4096x4096 and 8192x8192 pixels in size, in addition to the original
Figure 2.1: Multilevel Representation of Images on Disk
image. This method of storing data in multiples of 2, may lead to problems for images which are of non-standard sizes. We address this by rounding up the next to the next power of 2 from the original highest resolution image.

2.1.1 Multiresolution Image Pyramid and Tiles

An Image Pyramid is a data storage structure that represents an image as a set of multiresolution layers (Figure 2.2) stacked over each other in an orderly manner. The bottom of the pyramid contains the full resolution image with intermediate layers representing smaller resolution pixel versions resized to intermediate sizes. In addition, each layer is subdivided into a mosaic of image Tiles. A tile is a small square region of contiguous pixels in an image. We have used tiles of sizes 32x32 pixels (total size = 1 KB) which form logical blocks of separation within the image. Tiles provide "region of interest" access by allowing selective request of image areas.

![Image Pyramid Diagram](image)

**Figure 2.2: Structure of the Image Pyramid**

The image pyramid uses at most 33% more data space than the original image. \( X = \text{Pixel width/height of original image. Total file size} = X^2 + \frac{X^2}{4} + ... = \frac{4}{3}X^2 \) compared to the original size of \( X^2 \) pixels).

This process of multiple storage can easily be applied even if the original data is compressed using some compression algorithm. The lower resolution levels would be compressed or coded in a similar fashion in a pyramidal structure. In the ISN server, the data is stored in raw byte format for most images, either in 8-bit or 24-bit coding depending on the depth of the display screen. This is done to allow for standard ways of storing and retrieving data.
2.1.2 Server Storage Subsystem for Prototype

Figure 2.3 explains the hierarchy of data storage at the server and the client. Depending on the number of connections being handled by the server all image files being handled would be transferred from the disk to the cache. After the file descriptor goes stale either because the client connection was broken or a different file was requested, the file is removed from the cache in order to minimize on disk space. In case of the prototype implementation, the file descriptor is referenced to a different image with all the images being on the same server disk. In cases of low load, or based on intelligent frequency of use, certain files might be stored on the disk most of the time, so that connection and access requests from the client can be handled with shorter processing time. Typically disks would manage one disk file per active connection. The disk data would contain all the levels of the current image, stacked on each other with a pointer in memory associated with the currently active resolution level.

2.2 Data Caching at Server

If client requests always had to be serviced from disk, response time would be unacceptably slow. Consequently, the server maintains a cache for each client and attempts to serve most requests from the cache, going to disk only to update the cache.

The aim is to minimize the latency of image blocks sent to the connected clients. The server maintains separate caches for each connection to the client. For each client, typically a cache of 4 MB or 2048x2048 pixels per zoom window is stored centered around the last access point from the client. This allows the server to get data for any 1024x1024 neighboring region around the last section sent to the client from the server. This cache is maintained for all active connections and is allocated once a connection has been established via initial handshaking.

With 256 MB of cache on the server, we can cache data for 64 clients if there is 1 zoom window/client. To make adaptive changes in the memory used by each of the client, we would keep a count of the number of connected clients and allocate the total memory uniformly between clients which allows better performance. In such a situation the server must dynamically update the cache limits for each connected viewer as clients leave and join the server. This approach would fully utilize the cache and we can also allocate more than the usual limit of cache per client if it is available. However as soon as more clients
Various files stored in database or image archive on the server disk drive

Archived Data → Image Disk Data

Multi-level stack of images at different zoom levels stored on server disk.
- Single server may service various clients with different image requests on the same server machine
- Images stored in ascending order with levels from 1024x1204 to 16Kx16K

Disk → Server Cache

Image tiled data active in memory, in case of multiple files, portion of this image might be stored.

Figure 2.3: Server Storage structure for Image data
join, we would need to allocate an updated fair share of the server cache to the newer clients also.

The state of the cache is managed by an application which takes care of separate memory partitions for each connection from the client. The cache is updated dynamically based on the requests from the clients. The difficult part in servicing requests is to implement an intelligent look ahead strategy to cache data before the next pan request is sent over by the client, avoiding unnecessary delays in service request time. In a multi-threaded implementation of the server, request handling and disk access can be multiplexed for better performance.

### 2.2.1 Just-in-Time Delivery of Image Tiles

The server transmits the image tiles to the client via the network, and the client reassembles them them into the requested area of the window. The client maintains a cache of 1024x1024 for each zoom window, centered on the window position. Each level within the image pyramid is independently tiled and the image tiling is performed when data is read in from the disk. This approach of fetching data as the user needs it is a *just-in-time* model of data delivery for panning. To give a subjectively smooth panning effect, incremental updates of the image data are sent out.

Figure 2.4 shows the structure of the data and shows the data used to repaint or redraw portions of the image within the ISN Viewer. At all levels within the image, the image is tiled into 1024 pixel size tiles and the blocks are globally indexed. This allows the server to only send out the blocks needed to redraw and allows the clients to quickly adjust the data within the zoom window while panning.

Figure 2.4 shows the original image with tile no. 6..8, 10..12 and 14..16 in the server cache. With repositioning of the zoom window, the tiles 6,7,10 and 11 need not be requested, and the new blocks of pixels replace the earlier ones. Note that this repositioning only needs shift operations within the memory. This can easily be achieved using linked lists within the image tile mosaic, with each block just re-addressed to a different location. Given a tile index, we can always use the same algorithm to access any portion of the image because all tiles are connected to their adjacent tiles using memory pointers.
2.3 Data Representation at the Client

To allow rapid response to panning requests, each client includes a local cache. We would like to limit the client cache requirements to under 4 MB per application to reduce the number of requests at the client machines which means we can support the base image and 3 other windows. Data is retrieved from the server in the form of 32x32 image tiles. This limits the amount of data that must be transferred and reduces display flicker that could result from redrawing larger blocks when panning.

In order to support smooth panning, a neighborhood of 1024x1024 pixels of data is cached at the client and the ISN viewer can access this portion of the data directly. Since the maximum size zoom window is 512x512 pixels, this extra data provides space in which the user can navigate without requiring new data from the server. Requests for new image tiles are sent to the server in order to keep the cache in synchronization with the zoom window’s position. Since tiles are 32x32 pixels, a mouse movement of 32 pixels generates a request for
a horizontal or vertical slab of tiles containing 32x1024 pixels. If the user pans diagonally, the requests are sent for 32x1024 Bytes horizontally and 31x1024 bytes vertically, a total of 63 KB of data.

If the user moves the mouse at a rate that is too fast for the system to keep up with, the system jumps ahead to the current mouse position rather than following behind the mouse.

2.3.1 Example of Incremental Window Redrawing

The following simple cycle shows how data is cached on the client and gives an overview of the process of memory allocation and deallocation for the windows in the actual implementation.

1. The ISN Viewer connects to the server and after initialization, a base image of 1024x1024 pixels is requested. The image is mapped to 1 MB area on the client cache and is displayed as the base image.

2. The user clicks at (200,600) in the (1024, 1024) space. A simple zoom window is created and is initialized with the contents of the 512x512 sub-image centered at the mouse. At this point, no data is requested since this zoom window is at the same zoom level as the base image.

3. User zooms in to 2x magnification. This causes the client to allocate 1 MB of memory cache for the window, the data requested is from the 2x image coordinates at (200x2-512, 600x2-512) = (-112, 688) to (912, 1712). The server sends only data for the regions (0, 688) to (912, 1712) and the remaining portion is white space which was initialized when the memory was allocated.

4. As the user pans across the image space, the next data fetch occurs when the mouse crosses the data boundaries of the cached tiles. Note this bounding box window is actually 512x512 pixels in size because of level 2 zoom. As we zoom further, the bounding box size shall decrease, although we keep the amount of data received at 1 MB. Each operation results in a request for 32 KB (when panning horizontally or vertically) or 63 KB (diagonally). The requests are sent after every 32 pixel tile boundary is crossed.

5. Another zoom operation to level 4 would flush the same 1 MB memory block and reload it with another 1024x1024 block. Since we are at level 4, or a 4096x4096 resolution level the window data on the viewer encompasses an area of 256x256 pixels.
with respect to the background image. Filtering of mouse data is done so that not every pixel motion corresponds to redraw operations. The understanding is that the human visual system discerns frame rates of 16 frames/sec as smooth and hence if the user moves the mouse by 100 pixels in 500 msec. then we need to draw only 8 times in that period, which means average of 12 pixels separation in redraws.

6. When the user resizes the window, the data associated with the window does not change in cache, only the portion of the window visible is remapped and set to the size of the bounding box.

7. When the user deselects the window, the memory cache is maintained at the state the user left it. This allows the user to select the same window again and start navigating from the same point the window was deselected. If the user creates another window, new memory area is allocated for it. If the window is removed the memory area is deallocated.

2.4 High Speed Network Support

What if the user moves his mouse extremely fast along the diagonal and requests a large amount of data from the server? Can we intelligently filter requests and still maintain smooth panning? Does the server and the client and the network have the capacity to give a good subjective response? Support from network is very important since the performance of the system depends on it as much as on the server and client caches. If the network cannot support large bandwidth requests, the "responsiveness" in navigation shall be severely affected.

The performance of the ISN is limited by the speed of networks over which the servers and the clients communicate. Typical applications of large image spaces have bursty requests for high volumes of data. For reasonable numbers of independent clients (10-12) using the network, the amount of traffic could easily lead to congestion over medium speed networks (5-20 Mb/sec). This would lead to degraded performance at the viewers for panning and zooming. For this reason, high speed networks (at least 100 Mb/sec) are preferred.
2.5 Related Work

Extensive work has been done for image storage and efficient retrieval of images from archives or databases. Most papers or implementations in the literature deal with applications for astronomical image processing or displaying large images on viewers as "small thumbnails" with fixed zoom increments within a single window. Leclerc and Lau [19] describe the TerraVision system which allows the user to view, in real time a synthetic creation of a landscape using elevation data and large number of aerial images. The storage structure used is pyramidal tile coding which is the multiresolution representation of terrain data divided into equal sized elements and quad encoded [15] images.

Stephenson and Voorhess [35] describe a large scale implementation of the IMACS: Interactive Multi-terabyte Image Archive. This work presents a realistic treatment for such large scale problems with emphasis on optimizations of storage and database management for images. This paper describes the functional architecture, data caching model and object management techniques for queries and retrieval. Their approach of implementing an "all softcopy" environment for the problem is very similar to our idea of using a client-server model. Most of the approaches discussed in literature have approached the problem by optimizing caching strategies and presenting hierarchical image storage for faster access. This thesis has used a modified version of the ideas suited for supporting multiuser multiwindow navigation through large images.
Chapter 3

Software Design of ISN

This chapter describes the structure and architecture of a prototype software implementation for the Image Space Navigator. General software design structure and implementation details of the ISN Viewer and the ISN server, including the service and control modules are discussed. The application is an X-Windows based image viewer and image server using C++ and Playground [13]. Playground is a distributed application design environment developed at Washington University. The user interface implemented in TclTk [37] provides the interface for controlling server connections and requesting specific image files from the remote image archive.

This chapter briefly describes the Playground design environment with particular reference to our application. The user interface and the underlying connection and control manager are presented. Also presented is the server application which controls the service requests to the clients for image viewing. Description of the Event Processing algorithm implemented for ISN viewer is presented. Data presentation and managing server requests for image navigation is controlled by the event handler.

3.1 ISN Viewer User Interface

ISN viewer gives the user the interaction and control facility to navigate and establish connections to the image server. The Figure 3.1 shows the GUI presentation with the components needed for the ISN application. Briefly described below are various features of the user interface:
Figure 3.1: Elements of the iSN Viewer and User Interface
The GUI allows the user to connect to any image server running the service manager for ISN. For flexibility the user should be able to address the server based on its global IP address, or using server names. The user should also have a list of available servers to choose from, thus eliminating the need to remember addresses. Once the user has specified an IP Address, the program connects to the remote server and gets the directory of available image files. The user interface allows the user to select an image or picture to view from the server database, and allows it to do so using a browse menu list of available files.

The user interface's main purpose is to allow smooth navigation within the image space. The user can select zoom, pan and resizing operations using the mouse or the arrow keys. This approach is a more efficient approach of navigation than the traditional approach of selecting actions from a pull-down or pop-up window menu.

3.2 Playground Distributed Design Environment

The Programmers Playground provides a methodology and set of software tools for writing interactive distributed applications. Playground offers an abstraction that serves as an insulating layer between the application code and the low level communication protocols. The basic motivation for using Playground was to use an existing framework of network communication for integrating with the ISN viewer and server modules in a client server system. This allowed us to concentrate on the software design for the ISN Viewer window control and supporting efficient data service and request methods for smooth navigation. The features used from Playground are for is used for the connection management between client and server modules and support for dynamically changing connections using the API.

3.2.1 Playground Design Model

Playground based applications use the I/O abstraction model of distributed computing. Each client and server initializes itself as a Playground module (i.e process) in a system with a set of data structures that may be externally observed or manipulated via the presentation (the external interface). Each module is written independently and modules are then configured by establishing logical connections between the data structures in the presentations. The communication primitives (sending and receiving messages and the operating system to network calls and vice versa) are taken care of by the Playground runtime system.
All variables have to be published explicitly with the public name, data type and access protection information. The data type field is used to ensure data compatibility across connections and also provides the clients and servers with appropriate data structures for image data memory and connection variables. The access protection is an additional level of security mechanism to avoid incorrect connections and to selectively control flow in a preferred direction. For example, the clients are supposed to send "requests" to server for image data. The client and the server use a software design pattern called Reactor (A pattern where a function or object method is registered with a variable, and it is invoked when the variable changes). The reaction functions are invoked when external updates in the direction of access protection are available. The connections are made from the Application Management System which has an automated application launcher. Report [13] discusses the main features of the Playground programming environment.

3.2.2 Service and Connection Model

Figure 3.2 shows an overview of how connections are managed and serviced. The server application continually runs on the server, where the data plug-in's (where external links from remote modules can connect or plug in and request data) poll for requests from the clients. The server works in a time shared fashion, polling for requests from the clients. After a cycle of requests is over, the server application checks the flags for each server cache and updates the caches if the last request required a disk I/O to take place. The bidirectional arrows show that both read-and-write connections are made to the server. The descriptors are registered when the initial handshaking takes place between the clients and the server.

The SRV variables are file descriptors for accessing images in the archive, and used for disk I/O and cache reads when the image data is accessed. The connections are made to the server descriptors, over which the server polls for activity, and when new updates are requested the data transfer is handled by the SRV variables as in figure 3.2. Playground manages these connections for the viewer and the server applications and uses reactive control for servicing data requests.

3.3 Architectural Overview of Application Modules

The distributed infrastructure provided by Playground requires the variables that exchange data and information over networks to be published and declared open for connections.
Error control and streaming of data is taken care of by the run-time system and the clients just request data and wait for it to arrive. A brief description of the modules involved in the ISN are described in this section.

3.3.1 Client Module

The variables published by the client and the corresponding mirror variables on the server maintain state and data for the ISN Viewer. Figure 3.3 shows the Playground connection manager with the connections and variables for ISN viewers running on different clients connected to single image server.

The client maintains an internal state about the position of the the current window and the associated zoom level. The viewer maintains a stack of pan and zoom windows and the state of each window element in the window is stored separately. The window state consists of:
1. Window size of the current window. This refers to the size as a multiple of 128x128 pixels. The possible values are 1, 2, 4 and 8 which refers to window sizes of 256x256, 512x512 and 1024x1024 pixels.

2. Current zoom level within the window. The zoom levels are powers of 2 starting with the base image being at zoom level 1.

3. Relative position of the window within the stack of windows. Any new window automatically becomes the top of the stack and stays there until a different window is selected. If the user clicks on a particular window it moves to the top of the stack and the rest of the windows are rearranged to reflect the correct order of stacking. The issue in such a dynamic situation is to ensure that each of the windows changes independently of the others while maintaining the global context within the base image.

This means that any position or zoom level change in the window should not affect the state of the underlying windows. The challenge in this case is to do this in an efficient and fast manner without redrawing the overlapping windows. ISN stores
the background (the image data behind the window) of each zoom window as an
XImage Map of the windows and the base image. This ensures that while panning,
the viewer doesn’t redraw individual windows but instead paints the currently selected
background from the memory. The background image is a snapshot of the instance
when the window was selected by the user.

4. Name service for the server module ID. The user only specifies the IP address of the
image server, the connections within Playground are made using the unique PGcom-
mid and the client resolves the PGcommID from the IP address supplied by the user.
The PGCommID is a combination of process id, socket number and the IP address
for the server module. This allows multiple servers to use the same machine or IP
address and service connections over different sockets.

The viewer requests and filters mouse motions so that only relevant data is requested from
the server. This involves skipping over intermediate mouse positions when the user moves
the mouse too quickly for the system to track on a pixel by pixel basis.

3.3.2 Server Module

The server runs in a time shared manner to service requests from the clients. To implement
fair allocation of service to all clients, the server application polls the connected clients
for activity on the port and “reacts” to any requests from the clients. The server module
requires the following information:

1. File name for image data to be transferred.

2. Position and list of tile numbers and the zoom level of each new request.

3. Control bits to signal transfer of data and inform the client about the availability of
a new data stream. This allows the server and the client to synchronize data transfer
and allows them to maintain consistent and similar states using these control bits.

3.3.3 Control Module

The control modules are used in the “group viewing” or the multicast mode between various
clients navigating through the same image space. The module maintains a single shared
memory for updates to all the participants in the group. The control module’s published
variables also have information about the virtual mouse, which is essentially the shared
cursor when a multicast view operation is being executed. The cursor shares all the updates
in position and resolution level sent in by the current master and updates are sent out to
all the slaves or the other members in the current multicast group.

3.4 Software Implementation of the Viewer

The main and most time critical component of ISN is the GUI or the viewer. To achieve fast
access to memory and full control of the zoom windows the ISN viewer is implemented
using the X-Windows system [16] under C++. The basic store block for the viewer is the
Window, which stores the state of each window within the image space.

// Define a class to store a Window based pixel information and
// coordinate data to be used for the ISN
class WindowStruct {
    public:
        short X;        // The X-coordinate (center of window)
        short Y;        // The Y-coordinate
        short Zoom;     // The Zoom factor
        short WF;       // The Window factor (size of Window)
        XImage *Image;  // The memory to store XImage data

    // The Constructor for this class follows ...
};

The ISN Viewer maintains a linked list of such window objects and each object is initialized
with some base parameters in the constructor. The memory for each window is allocated
when it is created and is disposed off in the destructor. In this implementation we assumed
a base screen of 1024x1024 pixels in size (that is actually larger in height than the typical
screens), and the zoom windows can be made at 512x512, 256x256 or 128x128 pixel levels
(these correspond to WF levels of 8, 4 and 2) respectively. The base image is part of the
initial theRootWindow, and acts as the background over which the other windows slide and
are selected.
Figure 3.4: Navigation Cycle with Window States
3.4.1 Event Processing Algorithm

The basic event loop processes all the mouse motion and click events generated by the Xserver. The following algorithm outlines the basic components of the Xevent loop.

Get event from Xserver queue  
Examine event  
If (MotionEvent)  
   Flush queue until the last Event -> CurrentEvent  
else  
   CurrentEvent <- Head of the queue (Pop it off)

Case (CurrentEvent)

   EXPOSE : Display base image and Initialize data structures
   ENTRY : Install Local ColorMaps
   EXIT : Uninstall ColorMap

   KEYMAP : Invoke ISN Control User Interface

   KEYPRES : Look for mapping for
      - Quit (Leave Viewer, cleanup and kill)
      - Esc (Delete current window, reset the window queues)
      - Up  (Resize window to Larger size)
      - Down (Reduce window to smaller size)

   LEFTBT : If (!FullScale)
      Request data from server, wait for data
      Flush Window memory and display higher resolution data
      Update the State

   MDLEFT : If (WindowSelected)
      Deselect it
      Update state
      Disengage Screen pointer
   else  
      If (InsideAnyWindow)
         Engage the screen Pointer to the center
         update State
else
    Create new window
    Update window list
    Engage screen pointer

RGHTBT : If (!BaseScale)
    Request data from server and wait
    Reflush window memory and display lower resolution data
    Update the state

MOTION : Update screen pointer
    If (DataNotinMemory)
        Request additional data from server to
        update range of window
        Redraw at new location after XORing the earlier position of window

Loop

3.5 Summary

This chapter described the software design details of the prototype ISN implementation. The motivation for using Playground and the features, including variable publishing and reactor registration were described. The model and description of the client viewer and the server manager, including the strategy used for service requests was explained. The appropriate data structures and the code sections of relevance to the event processing of mouse motion for the viewer was presented. Chapter 5 shall briefly overview the multicast or the group viewing idea for ISN.
Chapter 4

Performance Models and Evaluation

In this chapter, we develop a simple performance model for the ISN and use it to evaluate the system performance so that we can configure the system appropriately. The objective of the performance evaluation is to understand the system from a quantitative standpoint in order to engineer the system elements to achieve the best performance.

Figure 4.1: State Transition Diagram for the Client (Image Viewer)
4.1 Client Model

The client model is shown in Figure 4.1. It is a continuous time Markov chain model with 3 states: view, zoom and move. The zoom state is a transitory state with a zero holding time that models a request to change the zoom level of the zoom window. The view state models the periods where the user is viewing the image but not zooming or panning. The move state models periods where the user is moving the zoom window. For simplicity, we assume that there is just a single zoom window.

The average holding times in the view and move states are \( t_v \) and \( t_m \). The probability that we go from the view state to the zoom state is denoted by \( p_z \). The probability of moving from the view state to the move state is \( p_m = 1 - p_z \). The average panning rate while in the move state is \( s \) pixels/sec.

There are several other variables it is useful to define. Let \( \lambda = \frac{p_m}{t_v} \) be the transition rate from the view to the move state, \( \mu = \frac{1}{t_m} \) be the transition rate from the move to the view state and \( \gamma = \frac{p_z}{t_v} \) be the transition rate from the move state to the zoom state. Table 4.1 presents an overview of the parameter values used in simulation and modeling of the client system.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
<th>Typical</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_m )</td>
<td>0.05 - 0.95</td>
<td>0.8</td>
</tr>
<tr>
<td>( p_z )</td>
<td>0.05-0.95</td>
<td>0.2</td>
</tr>
<tr>
<td>( t_v )</td>
<td>2 - 30 sec.</td>
<td>10 sec.</td>
</tr>
<tr>
<td>( t_m )</td>
<td>1 - 15 sec.</td>
<td>4 sec.</td>
</tr>
<tr>
<td>( s )</td>
<td>20 - 1000 pixels/sec</td>
<td>256 pixels/sec</td>
</tr>
<tr>
<td>( N )</td>
<td>1 - 32</td>
<td>4</td>
</tr>
</tbody>
</table>

4.2 N-Client Model

A set of \( N \) clients are modeled as a birth death process in which the state index is the number of clients in the move state. Since each client has only two non-transitory states the remainder of the clients are assumed to be in the view state. Figure 4.2 describes the state transition chain including the rates of transition in and out of each indexed state. The self loop models changes in the zoom level for clients in the view state.
The rate $\lambda_i = (n - i) \lambda$ is the transition rate for transitions from state index $i$ to $i+1$. The rate $\mu_i = i\mu$ is the rate at which the transitions occur to state $i-1$. $\gamma_i = (n - i) \frac{p_z}{t_v}$ is the rate at which zoom requests are sent when the system is in state $i$.

### 4.2.1 Steady State Equations

In steady state, the rate at which transitions occur out of a state has to match the rate of transitions into a state. Consider the $Rate_{out}$ as the rate of transitions out of state $i$ and $Rate_{in}$ as the rate of transitions into state $i$ from adjacent states. All the rates have to be normalized with the state probabilities to match in a steady state. $\Pi_i$ is the steady state probability of the $i$ clients being in the move state.

\[
Rate_{out} = (n - i) \frac{1 - p_m}{t_v} \Pi_i + (n - i) \lambda \Pi_i + i \mu \Pi_i \tag{4.1}
\]

\[
Rate_{in} = (n - i) \frac{1 - p_m}{t_v} \Pi_i + (n - i + 1) \lambda \Pi_{i-1} + (i + 1) \mu \Pi_{i+1} \tag{4.2}
\]

The following equations are valid for all the state indices except on the boundary of the state transition diagram (i.e. $\forall i \in [1, n - 1]$). The boundary conditions using the finite client model give us the following relations.

\[
\lambda_{n-1} \Pi_{n-1} = \mu_n \Pi_n \tag{4.3}
\]
\[ \lambda_0 \Pi_0 = \mu_1 \Pi_1 \]  

(4.4)

The state probability is given by:

\[ \Pi_k = \frac{\binom{n}{k} \rho^k}{(1+\rho)^N} \]  

(4.5)

\[ \rho = \frac{\lambda}{\mu} \]  

(4.6)

\[ 0 \leq k \leq N \]  

(4.7)

1. \( \bar{N} \) is the average number of clients in the move state. It is \( N \) times the probability that a single client is in move state.

\[ \bar{N} = \sum_{k=0}^{k=N} k \Pi_k = N \frac{\lambda}{\lambda + \mu} \]  

(4.8)

2. The number of pan requests/second due to panning is \( R_m = \frac{s}{32} \bar{N} \), where \( s \) = the scanning rate in pixels/sec and each request is sent for a motion of 32 pixels which is the tile size of the image.

3. The number of zoom requests/second is

\[ R_z = \sum_{i=0}^{i=n} (n - i) \frac{P_z}{t_v} \Pi_i = \frac{P_z}{t_v} \frac{\mu}{\mu + \lambda} \]  

(4.9)

where all the zoom requests are for data of 1024x1024 pixels in size.

4. To find the overload probability at the server, we need to look at the probability that a certain number of users are in the move state. Let \( \Pi^k \) denoting the probability that at least \( k \) clients are panning

\[ \Pi^k = \sum_{i=k}^{i=n} \Pi_k \]  

(4.10)
4.2.2 Average Network Bandwidth

Using the above equations, the average network bandwidth can be computed in terms of
the holding times and the probabilities of transition.

\[ R_z = \frac{p_z}{p_m t_m + t_v} \quad (4.11) \]

\[ R_m = \frac{p_m t_m \frac{s}{32}}{p_m t_m + t_v} \quad (4.12) \]

\[ R_{total} = n[R_z(1024 \times 1024 \times 8) + R_m(1024 \times (32 + 31) \times 8)] \quad (4.13) \]

4.3 Example

In order to get a better feel of the bandwidth required, we look at some values obtained
for the zoom bandwidth and the pan bandwidth. If \( p_m = 0.8 \), \( t_m = 3 \) seconds and \( t_v \)
= 10 seconds. Then the network bandwidth needed to respond to zoom requests \( (R_z) \) is
132 Kb/sec per user and the pan bandwidth is 1561 Kb/sec when the user pans at 512
pixels/sec. The total bandwidth required is 1.7 Mb/sec/user.

If the network links support speeds of 10 Mb/sec, then one can only support 6 users on
average. Increasing \( t_m \) to 5 seconds while keeping everything else the same changes the
total bandwidth to 2.42 Mb/sec per user and we can only support 4 users on this network.
Similarly we can use the birth death model to find out the probability that the network
bandwidth required exceeds the link rate and the average number of clients that can be
serviced simultaneously.

4.4 Variation of Bandwidth with Transition Probability

In the last section, we formulated an expression for the bandwidth requested by the zoom
operation and the pan operation within the image. Both of these bandwidths \( (R_z \) and \( R_p) \)
depend on the state transition probabilities \( (p_m \) and \( p_z) \).

Figure 4.3 shows the variation of bandwidth with the state transition probabilities. As
the zoom probability increases, the bandwidth for zoom also increases. Since the user now
spends comparatively less time in the move state the bandwidth required for pan requests
Figure 4.3: Variation of Bandwidth with Transition Probability

decreases. The interesting observation is that the bandwidth for pan requests falls faster than the rate at which zoom bandwidth increases.

4.5 Variation of Bandwidth with State Holding Time

A larger holding time in the pan state means the user sends more requests for data while panning through the image space. For the view state, increasing the holding leads to lower bandwidth requirements.

Figure 4.4 shows the variation of bandwidth with state holding times. Since the state holding times, $t_m$ in the move state and $t_v$ in the view state are independent a 3-D plot of the bandwidth variation is shown. It can be seen that the zoom bandwidth remains same for large values of the view holding time. However, the zoom bandwidth changes when the view holding time is also small because the user spends a larger percentage of total view time in the move state since the value of $p_z$ is fixed. It is seen that the pan bandwidth is the dominating element of the total bandwidth and the increase of pan bandwidth with
Figure 4.4: Variation of Bandwidth with State Holding Time \( (p_m=0.8, s=512) \)

Increasing move holding time is significant. Note that pan bandwidth is directly related to \( s \). So, a faster average panning rate will lead to a substantial increase in the total bandwidth requirements.

4.6 Probability Distribution of Move State Index

Figure 4.5 shows the distribution of the number of clients connected to the move state. The graphs are indexed by the number of clients. As the number of clients increases the distribution tends to get flatter and has a smaller ratio of standard deviation to mean. The peak of the distribution is located at the most probable number of clients in the move state, which is very close to the average number of clients in the move state. The average number of clients helps us in computing the bandwidth required from the network.

Figure 4.6 shows the distribution of the number of clients plotted on a semilog scale. This is helpful for determining the 99% point for the distribution of the clients in move state. If 8 clients are connected to the server, there is more than 99% chance that not more than
4 clients are in the move state. If we want to service requests at the server within 100 ms for each client we can compute the bandwidth required for server processing. To service 4 clients, with a maximum delay of 100 ms per client requires a server bandwidth of 20.2 Mb/sec, since each pan request is 63 KB in size.

Table 4.2: Variation of Network and Server Bandwidth with Number of Clients

<table>
<thead>
<tr>
<th>N</th>
<th>\bar{N}</th>
<th>\bar{N}_{0.99}</th>
<th>R_{0.99} (Mb/sec)</th>
<th>\tilde{S}_{0.99}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.20</td>
<td>1</td>
<td>1.61</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>0.37</td>
<td>2</td>
<td>3.22</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>0.75</td>
<td>3</td>
<td>4.83</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>1.47</td>
<td>4</td>
<td>6.44</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td>2.94</td>
<td>7</td>
<td>11.27</td>
<td>20</td>
</tr>
<tr>
<td>32</td>
<td>5.88</td>
<td>11</td>
<td>17.71</td>
<td>33</td>
</tr>
<tr>
<td>64</td>
<td>11.76</td>
<td>19</td>
<td>30.59</td>
<td>55</td>
</tr>
</tbody>
</table>

Table 4.2 shows the variation of the number of clients in the move state that contribute to the bandwidth for smooth panning. \( \bar{N} \) is the average number of clients in the move state. \( \bar{N}_{0.99} \) refers to the maximum number of clients in move state with a 99% chance and \( R_{0.99} \) corresponds to the bandwidth for this situation. \( \tilde{S}_{0.99} \) corresponds to the number of pan
requests per second that the server gets with 0.99 probability. An interesting observation is that as the number of clients increases, the average number of clients in the move state is always less than 20% of the total.

4.7 Summary of Results

The basic motivation to design a client model and evaluate it was two fold. Firstly, we needed to develop a measure to predict the amount of network bandwidth that would be required for panning and zooming and also evaluate the probability of server congestion in terms of number of clients asking for service. The second reason is to use our knowledge of the client statistics to predict the time delay and latency at the server using simulation model. The server model is presented in the next section.

Table 4.3 summarizes some of the results obtained for the system using the client model.

The medium performance prototype of ISN running over medium speed networks was analyzed using a time log of navigation. The user connects and starts viewing an image and the time log prints the time in milliseconds since the start of navigation with a summary of each command that was executed including requests for pan and zoom data. Table 4.4 shows the values observed for a single user navigating a 4096x4096 pixel image.
Table 4.3: Summary of Results

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transition Probability</td>
<td>$p_m$ 0.6 - 0.9</td>
</tr>
<tr>
<td>Average Move Holding Time</td>
<td>$t_m$ 2 - 5 seconds</td>
</tr>
<tr>
<td>Average View/Zoom Holding Time</td>
<td>$t_v$ 8 - 15 seconds</td>
</tr>
<tr>
<td>Pixel Panning Rate</td>
<td>$s$ 64 - 4096 pixels/sec</td>
</tr>
<tr>
<td>Maximum Number of Clients</td>
<td>$N$ 16</td>
</tr>
<tr>
<td>Average Bandwidth for Panning/User</td>
<td>$R_p$ 0.8 - 1.6 Mb/sec</td>
</tr>
<tr>
<td>Average bandwidth for Zooming/User</td>
<td>$R_z$ 0.2 - 0.35 Mb/sec</td>
</tr>
<tr>
<td>Average Number of Clients in Move State</td>
<td>$\bar{N}$ 4</td>
</tr>
<tr>
<td>$Pr[\text{Number of Clients in Move State} \leq 8]$</td>
<td>$N_8$ 0.997</td>
</tr>
</tbody>
</table>

Table 4.4: Observations from Time Log for Single User Navigation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Navigation Time</td>
<td>72.65 s</td>
</tr>
<tr>
<td>Total Time Spent Panning</td>
<td>20.2 s</td>
</tr>
<tr>
<td>Total Time Spent in Zoom</td>
<td>8.1 s</td>
</tr>
<tr>
<td>Fastest Pixel Scanning Rate</td>
<td>2742 pixels/sec</td>
</tr>
<tr>
<td>Average Pixel Scanning Rate</td>
<td>234 pixels/sec</td>
</tr>
<tr>
<td>Average Bandwidth Observed</td>
<td>4.5 Mb/sec</td>
</tr>
<tr>
<td>Average Delay for Zoom Data</td>
<td>2.82 s</td>
</tr>
</tbody>
</table>

The probability of client being in the pan state was computed as $\frac{1}{\lambda + \mu}$. Using the typical values of parameters for the client model, this comes out to 21%. If we look at Table 4.4 the ratio of the pan to total time is 26% which is reasonably in agreement with our estimate. Also the ratio of view time to total time is 79%. The expectation is that when larger images are navigated the user shall spend more time in panning through the image space and the values assumed for the performance model shall be closer to the observed values. The model predicts a bandwidth of 4.83 for 4 clients which cannot be supported on our network (observed = 4.5 Mb/sec), so to really maintain good performance characteristics we need to use a faster network or use at most 3 clients over this network.
4.8 Server Model

The server model is simulated to compute the service delay for pan and zoom requests observed by the user. The service model based on our implementation is simulated using SimScript [33] discrete event simulation.

Figure 4.7 shows the server model that is simulated. The model is divided into 3 basic sections. The Disk section simulates disk I/O required when client requests cannot be handled from the cache. The direct connection to the Server Application is for data already in the server cache and hence no waiting is required to service these requests. The Server Application simulates the actual process of data packing and sending by the server application to the client. The Network section simulates the transmission delay associated with the data when it is sent back to the client. This model does not include the latency in the network, and more importantly the time used by the client application to request data and the OS to application service time. The actual time to pack the image data in the appropriate data structure and present it on the screen is also not considered. Table 4.5 describes the default values for parameters used in the simulation model.
4.8.1 Arrival Statistics

Let $R = R_z + R_m$ be the total number of requests/sec from the clients from all the $N$ users. Requests are generated independently with exponentially distributed interarrival times with an average of $\frac{1}{R}$. The type of request (zoom or pan) is randomly determined with $R_z$ being the probability that an arriving request is a zoom request and $R_m$ being the probability of a pan request. We computed the average number of zoom request/sec and the average number of pan requests/sec in terms of $p_m$, $t_m$, $t_v$ and $s$ in the last section.

4.8.2 Disk Section

Since all zoom requests require reading data from the server disk, they all go through the disk section. Only when the pan data is unavailable in cache, a disk I/O needs to take place. It also depends on the response time of the disk and it is difficult to get an estimate of the number of pan requests which need a disk I/O. A reasonable estimate would be to assume a cache hit rate of 90% for the pan requests.

The parameter $t_d$ in figure 4.7 has different values for the zoom and pan requests. For zoom requests, $t_d(zoom) = 15 \text{ ms (access time)} + 100 \text{ ms (1 MB data transfer time)} = 115 \text{ ms}$. For pan requests, since the cache needs to be refreshed with current data, the time $t_d(pan) = 15 \text{ ms} + 400 \text{ ms (4 MB data transfer time)}$.

4.8.3 Server Application Section

The server application delay is fixed at 50 ms for time used by the Playground functions and other network application functions. The time taken to pack 63 KB of pan data is
almost 10 ms and for 1024 KB of zoom data is 120 ms. Therefore \( t_s(zoom) = 160 \text{ ms} \) and \( t_s(pan) = 60 \text{ ms} \).

### 4.8.4 Network Section

We consider two types of networks. The first network is a medium speed Ethernet LAN connection which has speeds of 10 Mb/sec. Therefore the times \( t_n(zoom) = 800 \text{ ms} \) and \( t_n(pan) = 70 \text{ ms} \). Using a high speed ATM connection, the time \( t_n(zoom) = 55 \text{ msec} \) and \( t_n(pan) = 4 \text{ msec} \).

### 4.9 Simulation Results

Table 4.6 shows the results obtained for the following set of parameters: \( p_m = 0.75, t_m = 3 \text{ sec} \), \( t_v = 10 \text{ sec} \), and \( s = 512 \text{ pixels/sec} \). The arrivals were simulated using \( N = 8 \) clients. The average delay for each section and the total delay for the request is shown in the following table.

**Table 4.6: Simulation Results (N = 8)**

<table>
<thead>
<tr>
<th>Request Type</th>
<th>Disk</th>
<th>Application</th>
<th>Network</th>
<th>Total Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pan</td>
<td>21</td>
<td>84</td>
<td>102</td>
<td>207 ms</td>
</tr>
<tr>
<td>Zoom</td>
<td>132</td>
<td>184</td>
<td>1107</td>
<td>1423 ms</td>
</tr>
<tr>
<td>Total (Average)</td>
<td>13</td>
<td>88</td>
<td>139</td>
<td>240 ms</td>
</tr>
<tr>
<td>Total (Std. Dev.)</td>
<td></td>
<td></td>
<td></td>
<td>192 ms</td>
</tr>
</tbody>
</table>

Note that the client cache has enough data for 8 pan requests, so if client scanning rate is small enough (\( \leq 8 \text{ requests in 240 msec} \)) then there is no stall in panning. This means the system can support scanning rates \( s \) of \( \frac{256}{0.24} = 1070 \text{ pixels/sec} \). Since zoom requests require a large transmission time and all the requests are implemented using the same queues we see that the pan requests suffer a larger delay. The minimum delay for pan requests would be 135 msec but since zoom and pan requests are handled with the same priority, we obtain larger delays for pan requests.
Table 4.7 shows the effect of network bandwidth on the delays. The results were obtained for the same set of parameters except that an ATM network is used which has much smaller network transmission time. It can be seen that increasing the bandwidth improves the delay by a large amount since that it a big bottleneck in servicing requests, however the server latency is still considerable.

Table 4.7: Variation of Delay with Network Speed

<table>
<thead>
<tr>
<th></th>
<th>Ethernet (10 Mb/sec)</th>
<th>ATM (150 Mb/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Delay</td>
<td>240 ms</td>
<td>112 ms</td>
</tr>
<tr>
<td>Pan Delay</td>
<td>207 ms</td>
<td>87 ms</td>
</tr>
<tr>
<td>Zoom Delay</td>
<td>1423 ms</td>
<td>352 ms</td>
</tr>
</tbody>
</table>

Table 4.8 shows the effect of client model or user navigation parameters. The results shown are for a faster scanning rate through the image and increasing the probability of zoom transitions ($p_z$) to 80%. Increasing the scanning rate clearly increases the queues on all the sections and although the zoom delay does not change by a large factor the total delay does because of higher rate of requests from the clients for pan than zoom.

Table 4.8: Variation of Delay with Navigation Parameters

\[
\begin{array}{|c|c|c|c|}
\hline
& p_z = 0.25 & s = 2048 & p_z = 0.75 \\
\hline
\text{Total Delay} & 240 ms & 402 ms & 195 ms \\
\text{Pan Delay} & 207 ms & 342 ms & 163 ms \\
\text{Zoom Delay} & 1423 ms & 1538 ms & 1203 ms \\
\hline
\end{array}
\]

Table 4.9 shows the effect of server processing speed on the delays. It is assumed that fixed server application latency can be reduced to 10 ms using a very fast server and also the pixel processing time can be almost doubled with a faster server CPU.

Table 4.10 shows the effect of server cache hit rate for pan requests on the delays. As the hit rate decreases because of the data access latency and the slower disk data transfer rate,
Table 4.9: Variation of Delay with Server Latency

<table>
<thead>
<tr>
<th></th>
<th>Slow Server (50 ms+ 10 Kpix/sec)</th>
<th>Fast Server (10 ms+ 20 Kpix/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Delay</td>
<td>240 ms</td>
<td>218 ms</td>
</tr>
<tr>
<td>Pan Delay</td>
<td>207 ms</td>
<td>191 ms</td>
</tr>
<tr>
<td>Zoom Delay</td>
<td>1423 ms</td>
<td>1131 ms</td>
</tr>
</tbody>
</table>

Table 4.10: Variation of Delay with Disk Hit Rate

<table>
<thead>
<tr>
<th></th>
<th>80%</th>
<th>90%</th>
<th>95%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Delay</td>
<td>263</td>
<td>240 ms</td>
<td>197 ms</td>
</tr>
<tr>
<td>Pan Delay</td>
<td>238</td>
<td>207 ms</td>
<td>165 ms</td>
</tr>
<tr>
<td>Zoom Delay</td>
<td>1503</td>
<td>1423 ms</td>
<td>1330 ms</td>
</tr>
</tbody>
</table>

the delay is expected to increase. It is reasonable to assume that the hit rate would vary depending on the user requests from 80% to 95%.

4.10 Summary

In this chapter we presented a client performance model and using the modeling parameters we evaluated the model to compute the average network bandwidth. From the results, it was observed that the pan bandwidth is the more important and demanding element and it also affects the responsiveness of navigation. The server model was simulated and it was observed for the fixed set of parameters that we should expect around 250 msec of delay in pan requests under normal navigation conditions whereas using medium speed networks the zoom request service delay was close to 1.5 seconds. Using faster networks helped the most since the majority of delay was because of transferring large chunks of data.
Chapter 5

Conclusion and Future Directions

In this thesis, we have described a medium performance prototype of the Image Space Navigator. Most of the test images for this system can be 8Kx8K pixels in size and one would require a higher performance system to handle larger images effectively. This chapter shall discuss modifications in the design and implementation for a higher performance version and an overview of the collaborative group feature of the Image Space Navigator which is not fully functional yet. In conclusion, the contributions of the thesis towards solving the problem of viewing large images are discussed.

5.1 Collaborative Image Viewing

The idea of the collaborative image viewing (see Figure 5.1) is to allow a group of users to view and navigate the same image. The concept is very similar to having a multicast image viewing conference. The “group viewing” mode of ISN connects various viewers in a shared memory mode at the server. The image updates for pan and zoom are sent out to all the connected clients. This mode of operation increases the amount of messages and the control information that is exchanged between server and the clients to maintain consistent states at all members of the multicast group. The amount of data required is very small compared to the image data transferred. This operation puts stronger constraints on the network delay and speed, since we would like all the connected clients to receive the updates simultaneously. A simple analogy to consider would be the size or the width of a pipe required connecting the clients and the server. In the single connection model, the piped traffic is multiplexed over the various requests from the clients.
In the collaborative image viewing case, a higher speed network is needed, since all updates go out in parallel and they have to serviced simultaneously. However, if we use a network which supports multicast the bandwidth can effectively be distributed over the various channels. This increase in the network bandwidth requirement also is supplemented by having smaller delays in the service request time, since requests from various clients could be closely spaced in time and we would like to service both of them within a upper delay on service request time. The simultaneous update is needed in order to synchronize the states of all the viewers in the group and maintain it during the group viewing mode at all times.

In a group there is a master client who at any instant controls the navigation and as image selection. However, any of the connected clients have the ability to become the master and control the navigation. This involves maintaining a state with the server for all the connections and switching the individual settings. Within the group connections, the "slaves" receive the exact same updates over the connections they needed to establish initially. This facilitates joining and leaving the group dynamically without a large overhead to switch modes of viewing. This allows the transfer of control as a "master" to any of the
clients or viewers in the multicast group by clicking on the menu button in the ISN control menu. The "one-to-many" connection functionality in Playground helps in sending multiple updates out to various clients simultaneously.

5.1.1 Software Structure

Another important support mechanism from Playground is the API for *element-to-aggregate* connections, which provides the basic framework to have multicast or collaborative image navigation connections. The FIFO based semantics for all logical connections guarantees that all states at the client and the server are consistent and match with each other. The original implementation of the ISN Server has a control module which provides the mechanism for maintaining common states. The state of the common mouse or cursor within the shared image known as the *Virtual Mouse* transfers the control of images out of the client to the server.

The tough part of implementation is to force events within the XEvent Queue for each individual client simultaneously and display the exact same information about images simultaneously. The memory and the caches can be updated externally from the server, but the issue is to synchronize and maintain consistent display states on all the clients. For example, when the master moves to a new position and the virtual mouse records and sends the data out to all the connected clients. Since the control is then transferred to the individual XServers running on each of the clients, all the clients redraw and update states at different times. Moreover, The XEvent queue is not available to the user as an object but is maintained internally by the Xwindows application. These are some of the tougher issues for implementing collaborative image viewing.

5.2 High Speed Image Data Transfer using APIC and ATM

The Playground APIC interface would allow an easy transition to a direct memory mapped interface using the APIC [9]. APIC is designed to support high speed application to network interface connection and data transfer using ATM [3] switches without time consuming operating system intervention for memory management. APIC is expected to support zero copy and secure space DMA which would allow the clients to directly read image data off the network into to Ximage structure from which it can be drawn on the screen. As we saw in chapter 4 the major service delay component was the data transfer speeds of the network. Using ATM networks and APIC in a desktop environment we can achieve data.
transfer speeds of 1.2 Gb/sec and this would bring the transfer time down by a factor of 100 to 10 msec or so and support much more responsive zooming and panning.

5.3 Priority Queues within ISN Server

The current implementation of ISN server implements simple data request service in a single thread of control. The requests from the clients cannot be accessed arbitrarily since the Playground runtime system maintains strict FIFO queues to enforce consistent states. We saw in the server model discussed in chapter 4 that pan requests suffer a large service delay because the zoom requests which have a larger service time and large network transfer time have the same priority. It would be logical to implement a separate service queue for pan requests since they can be serviced quickly and the frequency of pan requests from the clients is much larger than zoom requests, because they request smaller amounts of data incrementally. This reduction in service shall improve the smoothness of panning especially with larger images (say 64Kx64K) where the actual physical motion of the mouse corresponds to larger motions within the original image space.

5.4 Conclusion

The implementation of the ISN Viewer and ISN Server has shown that it is possible to design a system that is affordable and works with present day networks and workstations and solves the problem of viewing large images. The performance evaluation of the system helped us understand the bottlenecks in performance and gave us insight about engineering the system. The collaborative image viewing feature of ISN is still under development and it would be very interesting to use in a conference group of image navigators.
Appendix A

Color Figures

Figure A.1: San Francisco Bay Area
Figure A.2: Medical Imaging

Figure A.3: High Quality Digital Newspaper
Figure A.4: Satellite Imagery
Appendix B

ISN Viewer Reference Manual

ISN Viewer is the software tool for viewing large images and also includes the TclTk GUI for access to remote file servers. This manual is meant to be as a reference for using the tool for viewing images of a currently running image server.

B.1 ISN Control

Figure TKGUI shows the basic elements of the ISN Controller. When ISN Viewer is started the controller widget appears and then one has to select the server address. This is needed before the file data or any other information can be obtained. This is shown in figure as step 1. The dialog box appears and if the user enters an invalid IP address the widget flashes an error message.

In step 2, the client connects to the server and accesses the directory of the remote file server, and the user selects an image from the directory to view. After this the connections to the server are established and ISN Viewer starts up.

B.2 Mouse and Key Controls

ISN Viewer is implemented as a window in Xlib and hence it can be selected and iconized as normal Xwindows. The ISN controller can only be accessed when the user has left ISN
Viewer. Following are the key and mouse mapping for image navigation. The Window once selected shall follow the mouse to any location with the image.

1. **Q Key**: Exit the ISN Viewer

2. **Middle mouse key**: Draws a new window at the current position. If the key is pressed within a currently stacked window, it is selected and made as the current active window.

3. **Left mouse key**: Zoom in to the next higher resolution of the image. The new data is displayed within the bounding box of the current window. Has no effect if there is no currently active window.

4. **Right mouse key**: Zoom out of the current resolution within the window to the next lower resolution level. Has no effect if the user is either at the base image resolution level or no window is currently active.
5. **Up arrow key:** The current window is resized to half its size and the current image displayed within it. Has no effect if the original window size is the smallest possible (128x128 pixels).

6. **Down arrow key:** The current window is enlarged to double its present size. Has no effect if already of maximum size (512x512 pixels).

7. **Esc Key:** Deletes the currently active window and has no effect otherwise.
References


Vita

Kamal Bhatia

Date of Birth  
August 7, 1972

Place of Birth  
Kanpur, India

Degrees  
B. Tech. Indian Institute of Technology, Kanpur  
_Electrical Engineering, May 1995_  
B. Tech. (Minor). Indian Institute of Technology, Kanpur  
_Computer Science, May 1995_  
M.S. Washington University in St. Louis  
_Computer Science, August 1997_

Affiliations  
Liaison to the Dean (AGES)  
Sever Institute of Technology

Professional Societies  
Institute of Electrical and Electronic Engineers  
The Computer Society  
Association of Graduate Engineering Students

Publications  
*Washington University Computer Science Technical Report*  
WUCS-97-18

August 1997