Mobile Collaborative Medical Display System

Sanghun Park*
Dept. of Multimedia
Dongguk University
Seoul, Korea

Wontae Kim†
Dept. of Computer Science
Sogang University
Seoul, Korea

Insung Ihm‡
Dept. of Computer Science
Sogang University
Seoul, Korea

Abstract

Because of recent advances in wireless communication technologies, the world of mobile computing is flourishing with a variety of applications. In this study, we present an integrated architecture for a personal digital assistant (PDA)-based mobile medical display system that supports collaborative work between remote users. We aim to develop a system that enables users in different regions to share a working environment for collaborative visualization with the potential for exploring huge medical datasets. Our system consists of three major components: mobile client, gateway, and parallel rendering server. The mobile client serves as a front end and enables users to choose the visualization and control parameters interactively and cooperatively. The gateway handles requests and responses between mobile clients and the rendering server for efficient communication. Through the gateway, it is possible to share working environments between users, allowing them to work together in computer supported cooperative work (CSCW) mode. Finally, the parallel rendering server is responsible for performing heavy visualization tasks. Our experience indicates that some features currently available to our mobile clients for collaborative scientific visualization are limited due to the poor performance of mobile devices and the low bandwidth of wireless connections. However, as mobile devices and wireless network systems are experiencing considerable elevation in their capabilities, we believe that our methodology will be utilized effectively in building quite responsive, useful mobile collaborative medical systems in the very near future.

1 Introduction

Since its foundation in the late 1980s, the field of scientific visualization has witnessed tremendous technical achievements that enable scientists and engineers to effectively extract visual and meaningful information from huge, complicated, and abstract datasets. The field has continued to evolve rapidly in response to the arrival of new computing environments. One of the most attractive challenges in the visualization area nowadays is to exploit wireless computing technologies that may help build a ubiquitous, mobile visualization system, thereby supporting collaborative works.

So far, hand-held mobile devices such as the personal digital assistant (PDA) and cellular phone have been regarded as inappropriate scientific tools for a number of reasons, including small screen size, poor computational performance, limited wireless communication bandwidth, and reduced interaction capability. However, these mobile devices have undergone considerable elevation in

*e-mail: mshpark@dongguk.edu
†e-mail: matt99@grmanet.sogang.ac.kr
‡e-mail: ihm@sogang.ac.kr

their capabilities. These days, it is not difficult to see a PDA that supports a video graphics array (VGA) resolution of 640×480 pixels. The wireless connection bandwidth for mobile devices continues to expand rapidly, which has facilitated the ubiquitous nature of wireless communication. For example, a commercial service for ubiquitous portable Internet access is planned in Korea that provides mobile users with 1 to 3 Mbps of bandwidth. Such changes in the mobile computing environment demand a new paradigm for building a mobile visualization system that supports ubiquitous and collaborative work.

One of the many application fields that could benefit from these technological changes is the telemedicine community. In particular, rapidly advancing mobile technology is expected to offer a critical technological base in building a remote medical care and related information management system that supports an *anytime*, *anywhere*, and *anyone* access. In providing such a telemedicine service, the computer supported cooperative work (CSCW) feature, which enables interactive cooperation among remote users, may increase its usability drastically. For example, doctors may be able to treat their remote patients interactively, sharing a tailor-made virtual working environment through a home care telemedicine system.

In an effort to explore these new technologies for medical informatics, we have developed an integrated architecture for a PDA-based mobile display system allowing ubiquitous computing. Our emphasis is on the development of a medical system that enables users in different regions to share a working environment for collaborative visualization with the potential for exploring a huge volume of medical data, such as computed tomography (CT), magnetic resonance imaging (MRI), or positron emission tomography (PET) data.

In this paper, we deal with technical aspects that must be considered in designing a mobile collaborative medical display system. Our system consists of three major components: mobile client, gateway, and parallel rendering server. The mobile client serves as a user interface and enables the user to choose the display and control parameters interactively and collaboratively. The gateway takes care of requests and responses between mobile clients and the rendering server for efficient communication. Through the gateway, it is possible to share working environments between users, allowing them to work together in CSCW mode. Finally, the parallel rendering server is responsible for performing heavy visualization tasks that are beyond the capability of current mobile devices. We explain our system architecture and discuss our experience obtained from its implementation.

The paper is organized as follows. Section 2 reviews some related studies on PDA-based visualization. In Section 3, we explain in detail how our mobile medical system has been designed and implemented on PDAs, and demonstrate that it can be used effectively for building a cooperative mobile visualization environment. In Section 4, we briefly discuss performance issues by presenting some experimental results, and conclude the paper in Section 5.

2 Related Studies

To handle the computational limitation of mobile devices and the low bandwidth of data transmission networks, often experienced in the development of mobile visualization systems, several approaches have been proposed. In [8, 9], two solutions for remote visualization systems, involving low-end computing devices, were studied, where centralized hardware resources of high performance were exploited to allow users to visualize large scientific datasets. A general client-server model was utilized to design a remote rendering system [2]. In that work, the techniques of adaptive rendering and data transmission were used to overcome the low performance of the computing

hardware and the transmission network. An image-based rendering method was applied to the display of large polygonal models on mobile devices at interactive rates, 5 to 6 frames per second [1]. Various design principles and methodologies were also studied in several works [3, 4, 5, 6], endeavoring to develop effective mobile visualization systems. An interesting approach was recently presented in [7], where high frame rates are achieved through the help of high-performance parallel rendering based on PC clusters and progressive data transmission based on MPEG stream. While all these mentioned systems were designed to optimize the visualization and remote display computation in the respective computing environment, the concept of collaborative work between remote users was not fully applied to building their remote visualization systems.

OpenGL Vizserver from Silicon Graphics, Inc. (SGI) is a commercial, client-server solution [10]. It is a technical and creative computing system designed to deliver visualization and collaboration functionality to clients, whether on a desktop workstation or a wireless handheld computer. OpenGL Vizserver allows users to remotely view and interact with large datasets from any other system at any location in an organization and to collaborate with multiple colleagues using the same application data. The rendered scenes are transported as streams of compressed pictures from the server to the different client. Because of its design, OpenGL Vizserver works only with the SGI Onyx family, limiting its applicability in the general computing environment.

Another approach for an integrated remote visualization and collaborative system was attempted in [11]. For the visualization task, it harnesses remote servers to produce the computation results as video streams, separately for each participating partner. Thus, the mobile devices can only play MPEG-4-compliant video streams, commonly provided to off-the-shelf PDAs and laptops. For collaborative support, they use a full-featured CSCW system. This approach differs from ours in that they focus on integrating facilities for exploring complex datasets by using professional visualization tools in their knowledge space.

As mentioned before, telemedicine is a promising field to which mobile technology can be applied effectively. Telemedicine scenarios include in-hospital care management, remote teleconsulting, collaborative diagnosis, and emergency situation handling. Personal mobile telemedicine systems using wireless communication links have been studied extensively [12, 13, 14, 15, 16, 17].

Other types of mobile-based visualization systems have also been introduced, including a context-adaptive scalable information visualization and management system, which enables mobile computing [18]. The system intends to support the operation and the workflow of wastewater systems. The IMoVis project demonstrated how mobile devices may be applied in security systems [19], where PDAs are used to visualize intrusion detection data. The mobile augmented reality system was also designed to provide users with geo-referenced information required to accomplish their environmental management tasks via PDAs [20]. An approach for presenting information on handheld devices that uses information visualization techniques for displaying and navigating structures of complex information spaces has been introduced [21]. A framework for mobile 3D visualization and interaction in an industrial environment was presented in [22]. This project deals with the seamless navigation and provision of multimodal, context-sensitive, speech-enabled, augmented reality interfaces for mobile maintenance.

Although many similar approaches have been presented in the literature, our system is unique in that it supports collaboration (the ability to communicate and work with remotely located users) as well as a level of detailed features for high-resolution volume data.

3 The Proposed System

3.1 Design Overview

As described in the previous section, several mobile visualization systems have been proposed and implemented successively. Some systems have shown an almost real-time performance in remotely visualizing large volume data, for example, refer to systems developed independently from ours [7]. Essentially our system pursues similar goals to the previous systems, but focuses more on the following characteristics:

- A simple token protocol is designed for session-based CSCW. Through this protocol, it is possible to open multiple independent sessions at a gateway. Users that participate in a common session can share the visualization tasks. The system architecture is designed so that the gateway component that works as a broker between the mobile client and the rendering server is utilized in a way that optimizes the communication performance in a given environment.

- The level-of-detail concept is implemented in the mobile system so that a user can effectively select and explore subvolumes of interest through arbitrarily oriented cutting planes. A local cache structure is designed to minimize the volume extraction and transmission costs so that the display operation can be performed efficiently using the not-so-fast OpenGL ES API on a PDA.

- During data communication through a gateway, we do not consider a lossy compression scheme because the fidelity of medical imaging data is often critical. It is assumed that image quality is a design factor that is more important than the processing time. Hence, such techniques as JPEG and MPEG are not considered in our system.

- Basically, the system is designed to be independent, as much as possible, of high-level toolkits, libraries, or protocols, thus offering a high level of portability and extensibility. The current implementation uses such low-level libraries as OpenGL ES and socket-based TCP/IP. While the parallel rendering server module uses the MPI library to implement a traditional CPU-based ray-casting algorithm, any other parallel rendering methods including GPU-based ray-casting or Chromium-based parallel rendering can be adopted.

The current system does not provide a real-time rendering performance as it is based on CPU-based ray-casting. However, when the parallel rendering module is replaced by a fast parallel render as in [7], our system will also offer high rendering performance.

Figure 1 illustrates an overview of our mobile medical display system that consists of three main components: mobile client, gateway, and parallel rendering server. The system, based on a client-server architecture, aims to offer efficient and collaborative access to a large set of raw medical volume data, such as found in CT, MRI, and PET, that is stored in remote archives. One of the most important design factors is to provide users with both interactivity and mobility in visualization. To achieve this goal, a parallel rendering server, abstract to users, is harnessed to perform the heavy 3D visualization tasks, allowing a fast response to the user's request. Thus, the mobile client is free from CPU-intensive computation, focusing the user on user interfacing and collaboration with other clients.

For smooth communication between a client and the parallel rendering server or other clients, we introduce a gateway component that works as a broker. The gateway is primarily responsible

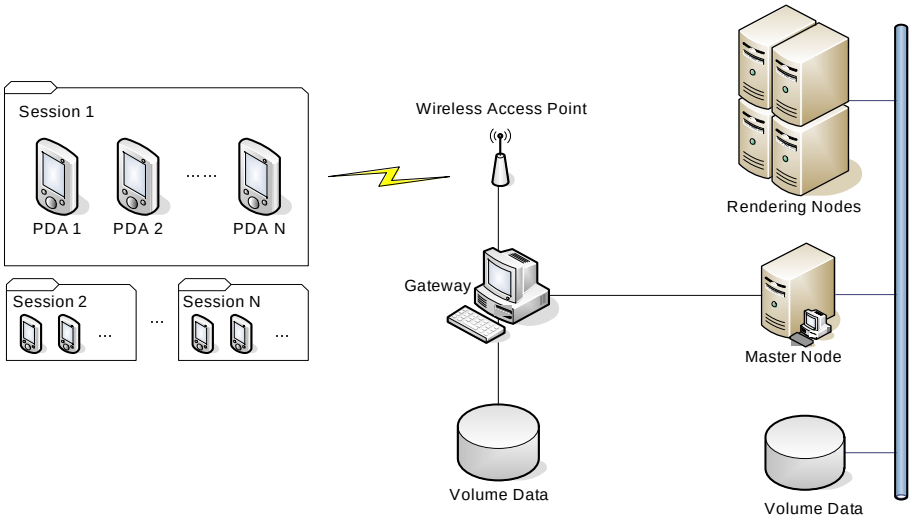


Figure 1: The system architecture: it consists of mobile client, gateway, and parallel rendering server.

for delivering requests collected from clients to the rendering server and multicasting the returned visualization results back to the clients. Furthermore, it controls the collaboration by maintaining contexts shared by clients. The communication between the components is performed using the socket-based TCP/IP protocol, which allows heterogeneous client devices to connect to the system stably and efficiently.

3.1.1 Mobile Client

The mobile client serves as an interface to the entire system that abstracts users from the implementation details. Mainly, it handles efficient interaction with users and displays images and movies sent from the rendering cluster. The client component, executed on a PDA, consists of the four submodules as illustrated in Figure 2: interface module, control module, network module, and rendering module. When a mobile client connects to the system for the first time, the control module collects relevant information through the network module, including the details of available volume datasets. Once the user selects a medical dataset to be processed, it prepares to start a visualization session. The interface module takes care of the user's actions by detecting input events and calling necessary callback functions. Through this interface, it is possible to perform many functions including: choosing a proper volume dataset, manipulating the selected data interactively, choosing the camera and light parameters, sending the rendering context, issuing a rendering request, and displaying rendered images. On the other hand, the rendering module interactively (at least 10 to 15 frames per second) displays subvolumes of interest using 2D texture-based rendering methods.

Once transmitted into the mobile client from the volume data repository as requested by the user, the raw CT or MRI data are stored in the form of 3D arrays for effective display that allows the user to scrutinize them with user-controllable cutting planes with arbitrary orientations. To obtain 3D-rendered images corresponding to a specific volume region, the user requests the parallel rendering server responsible for heavy rendering computations, for the visualization task. In return, the PDA receives the final rendering images or MPEG-encoded movies, and displays them on its screen.

In spite of the recent performance enhancement of mobile devices, it is still critical to make

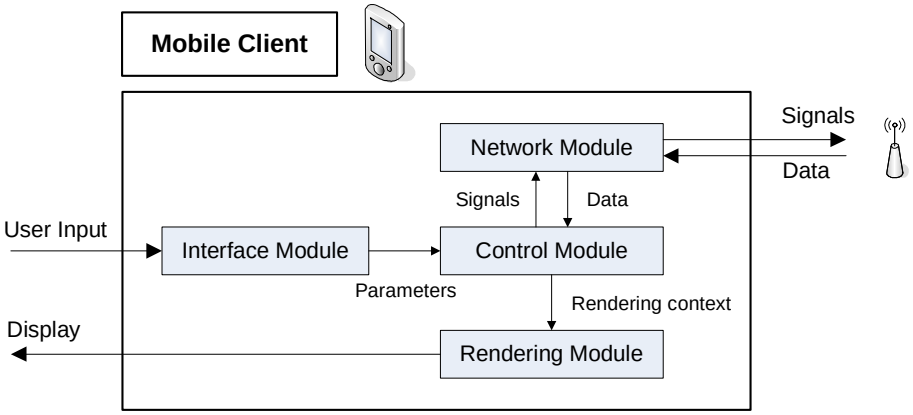
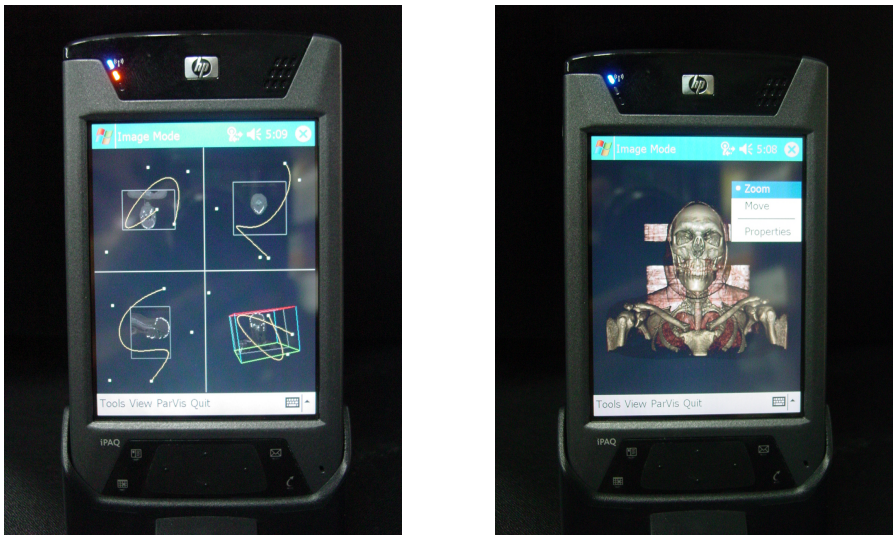


Figure 2: Mobile client modules



(a) (b)

Figure 3: Snapshot of mobile client's graphical user interface. Picture (a) shows an example where a nonuniform rational B-spline (NURBS) curve is being designed to set up a camera path that will be applied to the production of an animation movie. In picture (b), a 3D-rendered CT image is being displayed as received by the parallel rendering server. In case the resolution of the created image is higher than that of the PDA's screen, the user can pan or zoom with a pointing stick.

every effort to optimize the amount of computation carried out on the client side. In displaying a possibly large volume dataset on a screen of limited resolution, it is not wise to use the volume dataset of full resolution. In our framework, a multiresolution representation of each volume dataset is stored in the volume data repositories, managed by the parallel rendering server, and a set of subblocks with the proper level of detail, selected according to the viewing parameters and available client's memory, is transferred to the client. The data are reloaded into the client once every time the required level changes through a zooming operation. In this way, the usage of limited memory and wireless network bandwidth is optimized. See Figure 3 for examples of user interface where a filtered volume dataset is manipulated and a rendered image is displayed.

3.1.2 Gateway

The gateway is the central processing component that controls the entire system. The major role of this component is to drive the rendering cluster to perform parallel volume ray-casting with the rendering parameters received from mobile clients, and multicast the computed images or MPEG movies back to the clients. It also serves as a buffer that holds a set of raw subvolumes of selected level-of-detail values and delivers them to the clients as necessary. When a mobile client defines a camera path for the production of an animation movie (refer to Figure 3(a)), the gateway issues the corresponding rendering request at each camera point whose interval is controlled by the client. Then it receives multiple rendered images from the parallel rendering server, encodes them in an MPEG movie, and delivers it to the clients. The final task of the gateway is to coordinate the collaborative work between mobile clients, which will be explained in more detail in subsection 3.2.

As illustrated in Figure 4, the gateway is made up of the control module, network module, client management module, session management module, and data management module. The client management module handles each individual client, while the session management module manages active sessions to support the CSCW feature. The data management module plays a key role in retrieving volume data from the rendering server and filtering it so as to fit the volume size into the mobile client memory. When a client selects a region of interest, if the volume size is larger than the PDA's available memory, then the gateway determines the appropriate level of detail dynamically and reduces the volume data size accordingly.

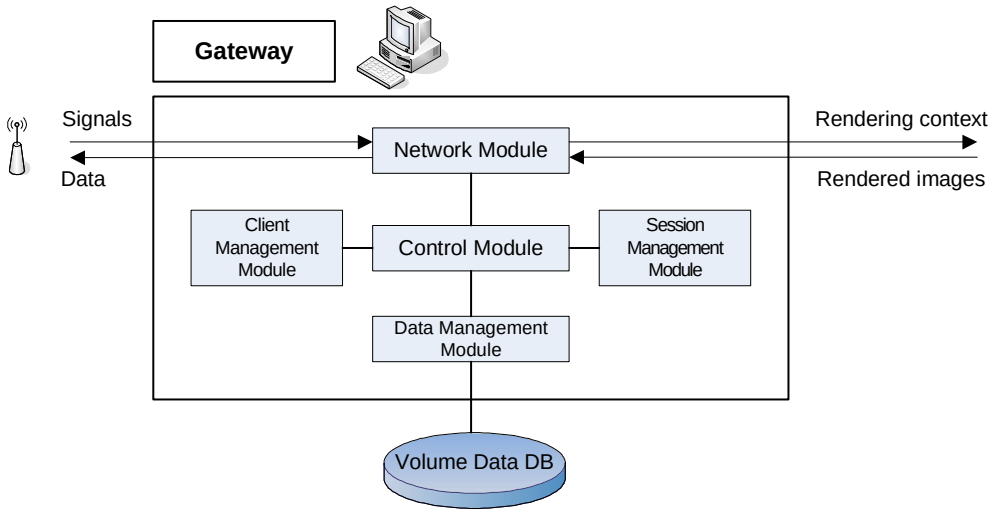
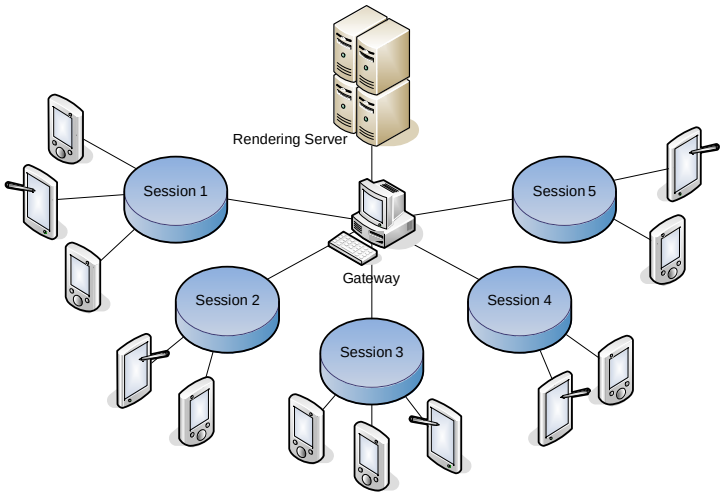


Figure 4: Gateway modules

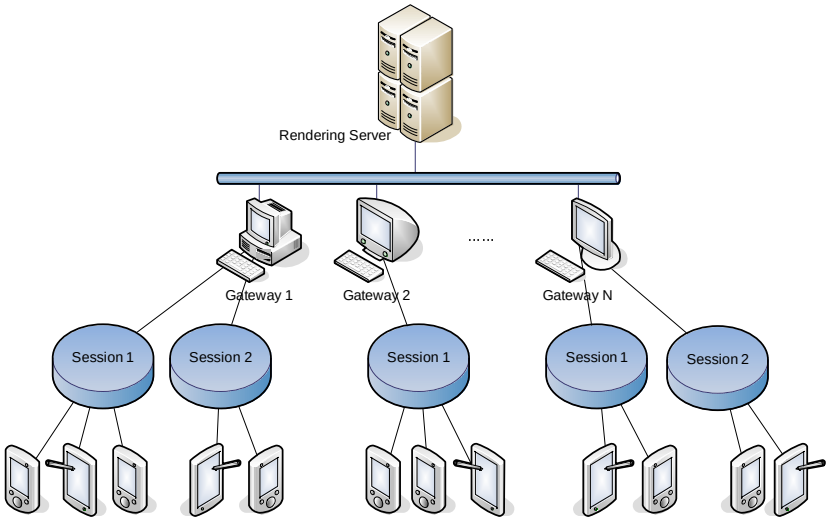
Importantly, when several mobile clients or sessions attempt to access a gateway at the same time, the communication needs may cause a severe bottleneck (see Figure 5(a)). To cope with this problem, we designed our system to allow multiple gateways to manage a group of sessions separately, as depicted in Figure 5(b).

3.1.3 Parallel Rendering Server

The parallel rendering server serves as computational clusters that handle the rendering requests delivered through the gateway as illustrated in Figure 6. As it is invisible to the clients, it may



(a)



(b)

Figure 5: Single gateway versus multiple gateways. When the number of participating clients increases, the chances are that the gateway becomes a bottleneck in the system. By employing multiple gateways, as in (b), instead of using a single gateway, as in (a), it is possible to make the load of the gateway component well balanced.

run on a supercomputer or fast PC clusters, interacting with the gateway. The server manages the volume data repositories that store the original volume datasets with their multiresolutional representations. For 3D volume rendering, our system harnesses a PC cluster that runs a parallel volume ray-casting algorithm designed for distributed memory architectures [23], while any other efficient parallel rendering technique may be used.

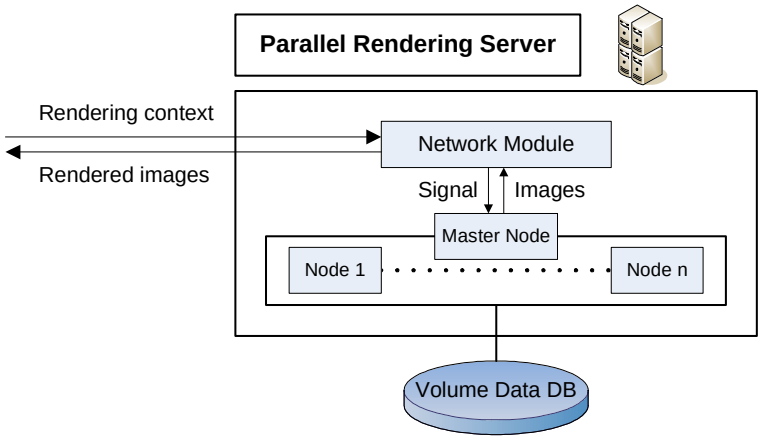


Figure 6: Parallel rendering server modules

3.2 CSCW

To support a complete cooperative environment, sometimes visualization systems have to exchange various imaging datasets and need efficient hardware accelerated volume rendering techniques for displaying the three-dimensional representation in real-time. However, it is difficult to implement such full-featured cooperative visualization techniques for huge and complex medical datasets on mobile devices because of their limited resources.

In our approach, the synchronization problem is solved through the gateway that always manages any updates for rendering contexts as well as volume data in a session. To describe the usage of the presented system, the following scenario summarizes the cooperation process: all cooperation partners who have access to a specific visualization can join a session through the gateway. When users contact the gateway, they can either select a session to join or create a new session. Depending on the working mode, the collaborators are able to see the same view as each other or an individual view of the visualization. Stored views, in which interesting regions are displayed, can be restored and broadcast immediately. Each collaborator who receives the broadcasting message can either update their own current view or maintain multiple views independently. This method allows users in the same session to share medical data, rendering contexts, and rendered images during the visualization task.

In introducing this ability for users to collaborate, it is important to avoid any disorder induced by random, simultaneous issues of the broadcasting messages by multiple users. Therefore, we have designed the session management module in such a way that only one user is given a token per session that authorizes its holder to issue a broadcasting message. The authority as a host may be turned over to other users in the same session by passing the token.

3.3 Scenarios of Use

To describe the mechanisms of a progressive display of remotely located large sets of medical data, we now introduce a possible interaction example between a client and a gateway, as illustrated in Figure 7. After the mobile client process is launched, a user connects to the gateway and sends the hardware configuration of the mobile device. In each request the mobile client sends a message or signal to the gateway. When the client requests an identifier, a unique identifier is returned, stored in the client device, and sent to the gateway in all subsequent requests. This operation enables the

gateway to keep track of all sessions. When the user requests a new part of the volume of raw data of interest using the bounding box of the graphical user interface, the mobile client computes the bounding box in volume coordinates. The client sends this information to the gateway, which can retrieve an appropriate subblock from the database. Next, the gateway determines a level-of-detail value, possibly based on the memory size of the mobile client device as well as the volume resolution. As the gateway keeps track of which subblocks have already been sent to the client, it can deduce the range of volume needed to update the display and send them to the client.

Figure 8 illustrates an example interaction sequence that may be observed when a user wants to render volume data of interest. The snapshots in Figure 9 show the volume rendering process step-by-step where 2D texture mapping and volume ray-casting methods are used in the mobile client and the parallel rendering server, respectively. Once a client connects to a gateway using their IP address, they select a dataset from the available raw data list (the pictures (a), (b), and (c)). Then the client determines a set of various rendering parameters through an interactive navigation of the chosen medical data (the pictures (d), (e) and (f)). After the setup of a transfer function for the appropriate classification of materials to be visualized, the client requests a parallel rendering server to perform the heavy visualization work. Finally, the rendered image is displayed on the screen (the pictures (g), (h) and (i)).

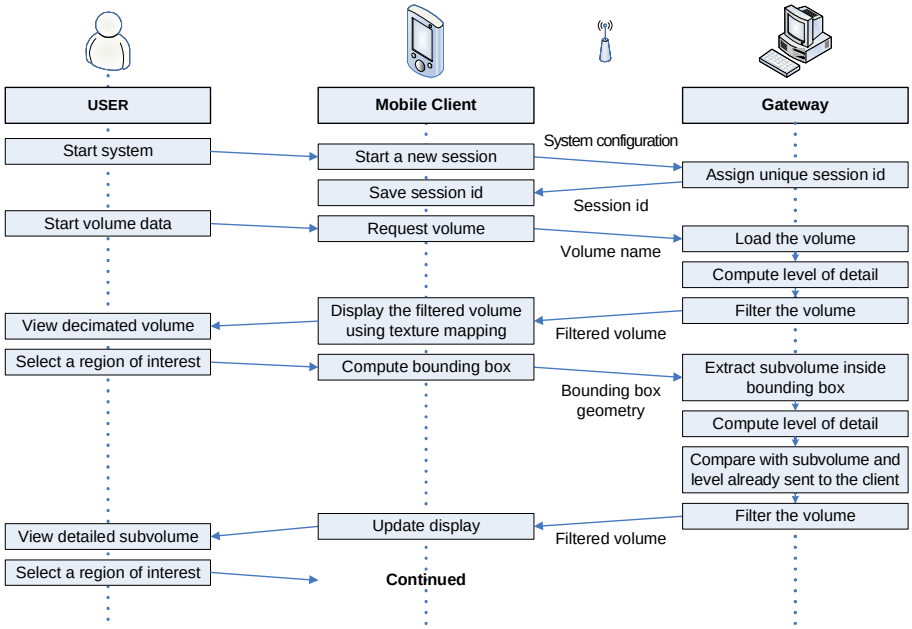


Figure 7: Example interaction diagram for client-gateway communication

Finally, Figure 10 illustrates an example sequence where two participating clients interact with each other for a CSCW purpose. In (a), the client on the left receives a signal from the current host client on the right, and is asked if the client will accept the new context, broadcast by the host. The client on the left may either ignore the message or choose to overwrite his context with the new one with or without saving the current context for the later restoration. On the other hand, (b) shows a situation where the host client is asked by the client on the left to turn over the host token. In (c), after the acquisition of the authorization, the client on the left, a new host, selects a menu item to broadcast his own 3D-rendered image. After the client on the right decides to accept the message, the received image is displayed on the screen as shown in (d).

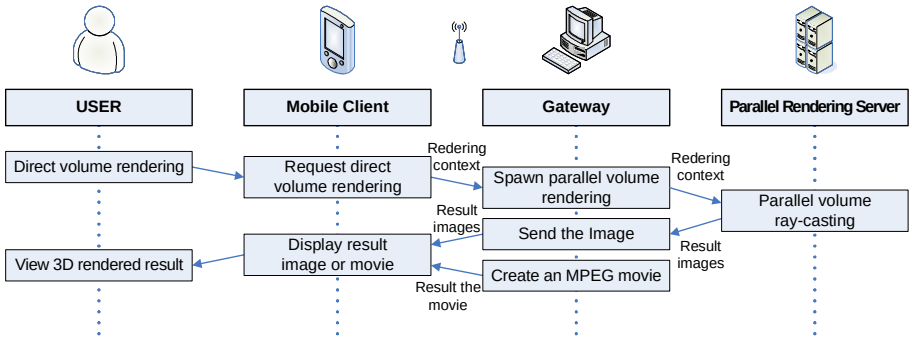


Figure 8: Example interaction diagram for client-gateway-server communication

4 Implementations and Optimization Efforts

To demonstrate the effectiveness of the presented prototype, we have implemented the mobile collaborative medical display system as described in the paper. For the mobile client, we used an HP iPaq hx4700 PDA equipped with 624 Mhz Intel PXA270 CPU, and 6.5k-color thin film transistor liquid crystal (TFT LCD) display supporting 480×640. The device, running Microsoft (MS) Windows Mobile 2003 for the Pocket PC operating system, was connected to the gateway via a wireless LAN or USB desktop synchronization cable. In developing the mobile user interface, we used MS Embedded Visual C++ 4.0 compiler, Vincent OpenGL ES [24], and MS Winsock.

The parallel rendering server was implemented and tested on an IBM PC cluster consisting of 15 nodes of 3.0 Ghz Intel Pentium4 processors and 2 Gb main memory. For communication between the nodes, MPICH2 message-passing interface was employed [25]. The mobile clients and the parallel rendering server were connected through a gateway for which an ordinary desktop PC was exploited. For the test volume data, the Visible Human male CT with $512 \times 512 \times 1440$ resolution and uptake capacity of 720 Mb was used. Figure 11 shows some examples of remotely visualized images that were created by the parallel rendering server and then displayed on the mobile client. See supplementary animation Movie 1(a) and (b) [1] for a demonstration of where a parallel rendering server with eight nodes was exploited.

When developing a software system on hardware with limited capacity, it is critical to make every effort for code optimization. In coding our system, we tried several optimization techniques for Intel XScale processor-based PDAs, and observed significant performance gains for some. As many mobile systems, based on the XScale processor, still are not accelerated by a floating-point unit, floating-point operations must be avoided as much as possible. An alternative is to employ a fixed-point number that approximates the floating-point number. Despite the limited range, the fixed-point data are often effective enough to code graphic applications on a PDA if the range is selected appropriately.

To enhance the processing speed on the mobile client side, we employed a 32-bit, fixed-point number where 16 bits are allocated to the fraction part. For multiplication and division operations, temporal 64-bit integer variables were used to maintain maximum precision. In displaying the raw volume data interactively, the frequently called trigonometric functions took a substantial portion of the floating-point operations. To handle this, we used a lookup table whose elements were generated for 0 to 90 degrees at one-degree intervals. We also applied several assembler-level code optimization techniques manually for compiler-optimized assembler codes, especially for repeat-

[1] The animation movies are also available at ftp://grmanet.sogang.ac.kr/pub/ihm/out/{mov1a.avi, mov1b.avi}

edly executed parts. Figure 12(a) indicates that these optimization efforts were rewarding. As implied in the diagrams, the application of both fixed-point numbers and optimization techniques remarkably improved the frame rate, while the code optimization techniques for floating-point numbers produced only a slight performance improvement.

The performance on the mobile client side is quite sensitive to the demands of various computational resources, and hence the code must be tuned with great care. In an attempt to reveal the dependency, we collected some performance data for several resources. As Figure 12(b) shows, the size of volume data loaded into the PDA's memory affects the speed of the interactive display of raw volume data. It is obvious that the smaller volume size results in a higher frame rate, whereas the performance does not vary much so long as the demanded memory stays in the 2–16 Mb range (note that the tested PDA holds 64 Mb of memory).

Currently, the performance is also dependent on the number of pixels that must be projected to compute an image because the PDA is not accelerated by a graphics unit. In the mobile client, the raw volume data are displayed using an interactively controlled cutting plane where the 2D texture-mapping feature is exploited for computing the resulting image. We have varied the distance of the cutting plane from the camera to vary the magnitude of pixels that need to be mapped onto the screen. As expected and demonstrated in Figure 12(c), the frame rate increases as the cutting plane goes farther, that is, the number of pixels to be colored decreases. However, this performance variation is expected to be avoided as a 3D graphics accelerator chip will be used in the near future.

Figure 12 (d) reports how performance is affected by the screen's resolution. To evaluate this performance, we used another PDA with a smaller resolution of 240×320 **pixels**, and measured the frame rate under the same rendering context.

Finally, Figure 13 shows some statistics that reveal how different block subvolume sizes affect the overall performance. First, the detailed analysis on the computational costs, needed to display level-of-detail images on the PDA screen, is given in (a) in the figure. In our system, when a mobile client selects a region of interest, the gateway module extracts the necessary subvolumes from disks, performs the resampling process so that the resulting image fits into the PDA's memory (32 Mb in this experiment), and sends the (processed) image to the client through the network. When the selected screen space corresponded to a subvolume of $512 \times 512 \times 512$ voxels, a resampling process was needed to reduce its size to less than 32 Mb. In the other three cases, no resampling was necessary. As seen in the graph, the disk access time was a minor factor in the overall performance. On the other hand, as expected, the sizes of subvolume to be transferred from the gateway to the mobile client greatly affected the processing time. While the bandwidth of the tested wireless communication network was low, the data transfer problem is expected to be resolved, as a much faster network will be available in the very near future.

The figure in (b), on the other hand, shows how the data size affects the parallel volume rendering in our current system. Again, four cases of volume size from $64 \times 64 \times 64$ voxels to $512 \times 512 \times 512$ voxels were tested against two screen image resolutions of 240×320 pixels and 480×640 pixels, respectively. In this experiment, a rendering server made of nine PCs was utilized to run our parallel volume ray caster. Except for the case of $512 \times 512 \times 512$ volume data, the entire response times were 2.0 to 3.3 seconds for the smaller screen resolution, and 4.7 to 8.8 seconds for the larger screen resolution. The parallel rendering time is dependent on the overall power of the participating PCs. Furthermore, while our current renderer is CPU-based, the recent GPU-based volume renderer provides an almost real-time frame rate for the moderate sizes of volume data. Hence, the rendering time will be expected to be a minor factor. On the other hand, as the test result indicates, the data transfer times for sending rendered images to the mobile server take a substantial portion of the entire processing time.

Despite the relatively poorer system specification of the tested PDA model (HP iPaq h5550 with 400 Mhz Intel PXA255 CPU, and 6.5k-color TFT LCD display), the performance metric was improved. In conclusion, the features currently available to mobile clients for scientific visualization are limited due to the poor performance of mobile devices. In particular, the bandwidth of wireless networks remains a major bottleneck in building an effective mobile display system, although it is expected to increase significantly in the very near future. For environments where wireless communication is still slow, it will be important to minimize the delay caused by the tardy communication.

Note that one of the many reasons (except the controller task as a broker in the CSCW process) why we considered employing the gateway scheme in our architecture was to tune the communication task so that the performance is maximized in a given environment. For example, it might have been possible to encapsulate the work done by the gateway in the rendering server. In that case, however, the remote server must take up all the responsibility for maintaining the system. In particular, all communication messages need to be delivered to the possibly very remote server system, which entails a severe inefficiency.

On the other hand, the use of a gateway may improve the system performance dramatically. For example, a clinic may install a local gateway that supports a relatively high speed of wireless communication to and from the mobile clients, i.e., patients in the local area. In this environment, the collaboration work between doctors and patients can be performed efficiently through local communication without having to send the message to the rendering server, which is geographically far away. Only the necessary messages, for instance, for sending the rendering requests or receiving rendering results are sent through a communication line between the local gateway and the possibly very distant rendering server.

5 Concluding Remarks

In this paper, we proposed a client-server-based integrated architecture for mobile collaborative medical data visualization in which PDAs are used as the front end. The system, composed of mobile client, gateway, and parallel rendering server, aims to offer interactivity and mobility in visualizing large medical datasets where remote users are allowed to collaborate in shared contexts. One of the main emphases of this work was to add the feature of collaboration and coordination to the mobile distributed medical system, allowing cooperative work among remote users. Through the implementation of the presented system architecture and the application of various optimization and tuning techniques, we have shown that low-end PDAs can be exploited effectively for such a mobile CSCW system.

As analyzed in the paper, the system performance relies on several design factors. Above all, the features currently available to the mobile client for scientific visualization are limited because of the poor performance of mobile devices. In particular, the low bandwidth of wireless communication networks is a heavy bottleneck in building a real-time or interactive mobile collaborative visualization system. As explained in the previous section, it is critical to design architecture that fully exploits the advantage of the gateway component to overcome the wireless communication problem. In our implementation, the gateway could handle no more than four or five mobile clients because of the poor performance of the tested wireless network. However, the capacity may vary as the system performance increases. Hence, it will be quite important to carry out the preperformance analysis when the system configuration is determined in the design step.

While the low wireless communication bandwidth is a current bottleneck, it is getting wider;

hence, quite responsive mobile collaborative medical systems will be able to appear in the very near future. We hope that the proposed mobile cooperative scheme will be applied in a wide range of applications. In telemedicine, for example, the expanded system may enable doctors to access 2D images of patients' raw data remotely, to visualize 3D images interactively, and collaborate with others in making a diagnosis, even when they are thousands of miles away from the patient.

References

[1] C.-F. Chang and S.-H. Ger. Enhancing 3D graphics on mobile devices by image-based rendering. In *Proceedings of The Third IEEE Pacific Rim Conference on Multimedia*, pages 1105–1111, December 2002.

[2] I. Martin. Adaptive rendering of 3D models over networks using multiple modalities. Technical Report RC 21722 (Log 97821), IBM T. J. Watson Research Center, April 2000.

[3] S. Stegmaier, J. Diepstraten, M. Weiler, and T. Ertl. Widening the remote visualization bottleneck. In *Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis*, pages 174–179, September 2003.

[4] K. Engel, O. Sommer, and T. Ertl. A framework for interactive hardware accelerated remote 3D-visualziation. In *Proceedings of Joint EUROGRAPHICS-IEEE TCVG Symposium on Visualization*, pages 167–177, May 2000.

[5] D. Beerman. Event distribution and image delivery for cluster-based remote visualization. In *Proceedings of IEEE Visualization - Workshop on Parallel Visualization and Graphics*, October 2003.

[6] F. Lamberti and A. Sanna. A solution for displaying medical data models on mobile devices. *WSEAS Transactions on Information Science and Applications*, 2(2):258–264, February 2005.

[7] F. Lamberti and A. Sanna. A streaming-based solution for remote visualization of 3D graphics on mobile devices. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):247–260, March/April 2007.

[8] S. Stegmaier, M. Magallón, and T. Ertl. A generic solution for hardware-accelerated remote visualization. In *Proceedings of Joint EUROGRAPHICS-IEEE TCVG Symposium on Visualization 2002*, pages 87–94, May 2002.

[9] F. Lamberti, C. Zunino, A. Sanna, A. Fiume, and M. Maniezzo. An accerlerated remote graphics architecture for PDAs. In *Proceedings of Web3D 2003 Symposium*, March 2003.

[10] Silicon Graphics Inc. SGI OpenGL Vizserver 3.5: Visualization and collaboration. White Paper, 2005.

[11] F. Götz, B. Eβmann, and T. Hampel. Using a shared whiteboard for cooperative visualizations. In *Proceedings of HCI International 2005*, July 2005.

[12] V. Mea. Agents acting and moving in healthcare scenario – a paradigm for telemedical collaboration. *IEEE Transactions on Information Technology in Biomedicine*, 5(1):10–13, March 2001.

[13] B. Woodward, R. Istepanian, and C. Richards. Design of a telemedicine system using a mobile telephone. *IEEE Transactions on Information Technology in Biomedicine*, 5(1):13–15, March 2001.

[14] Y.-H. Lin, I.-C. Jan, P. Ko, T.-Y. Chen, J.-M. Wong, and G.-J. Jan. A wireless PDA-based physiological monitoring system for patient transport. *IEEE Transactions on Information Technology in Biomedicine*, 8(4):439–447, December 2004.

[15] M. Bojović and D. Bojić. mobilePDR: a mobile medical information system featuring update via internet. *IEEE Transactions on Information Technology in Biomedicine*, 9(1):1–3, March 2005.

[16] J. Gállego, Á. Hernández-Solana, M. Canales, J. Lafuente, A. Valdovinos, and J. Fernández-Navajas. Performance analysis of multiplexed medical data transmission for mobile emergency care over the UMTS channel. *IEEE Transactions on Information Technology in Biomedicine*, 9(1):13–22, March 2005.

[17] J. Rodríguez, A. Goñi, and A. Illarramendi. Real-time classification of ECGs on PDA. *IEEE Transactions on Information Technology in Biomedicine*, 9(1):23–34, March 2005.

[18] J. Ehret, A. Ebert, L. Schuchardt, H. Steinmetz, and H. Hagen. Context-adaptive mobile visualization and information management. In *Proceedings of IEEE Visualization 2004*, 2004.

[19] A. Sanna and C. Fornaro. IMoVis: A system for mobile visualization of intrusion detection data. *International Journal of Information & Security*, 12(2):235–249, 2003.

[20] J. Danado, E. Dias, T. Romão, N. Correia, A. Trabuco, C. Santos, J. Serpa, M. Costa, and A. Câmara. Mobile environmental visualization. *The Cartographic Journal*, 42(1):61–68, June 2005.

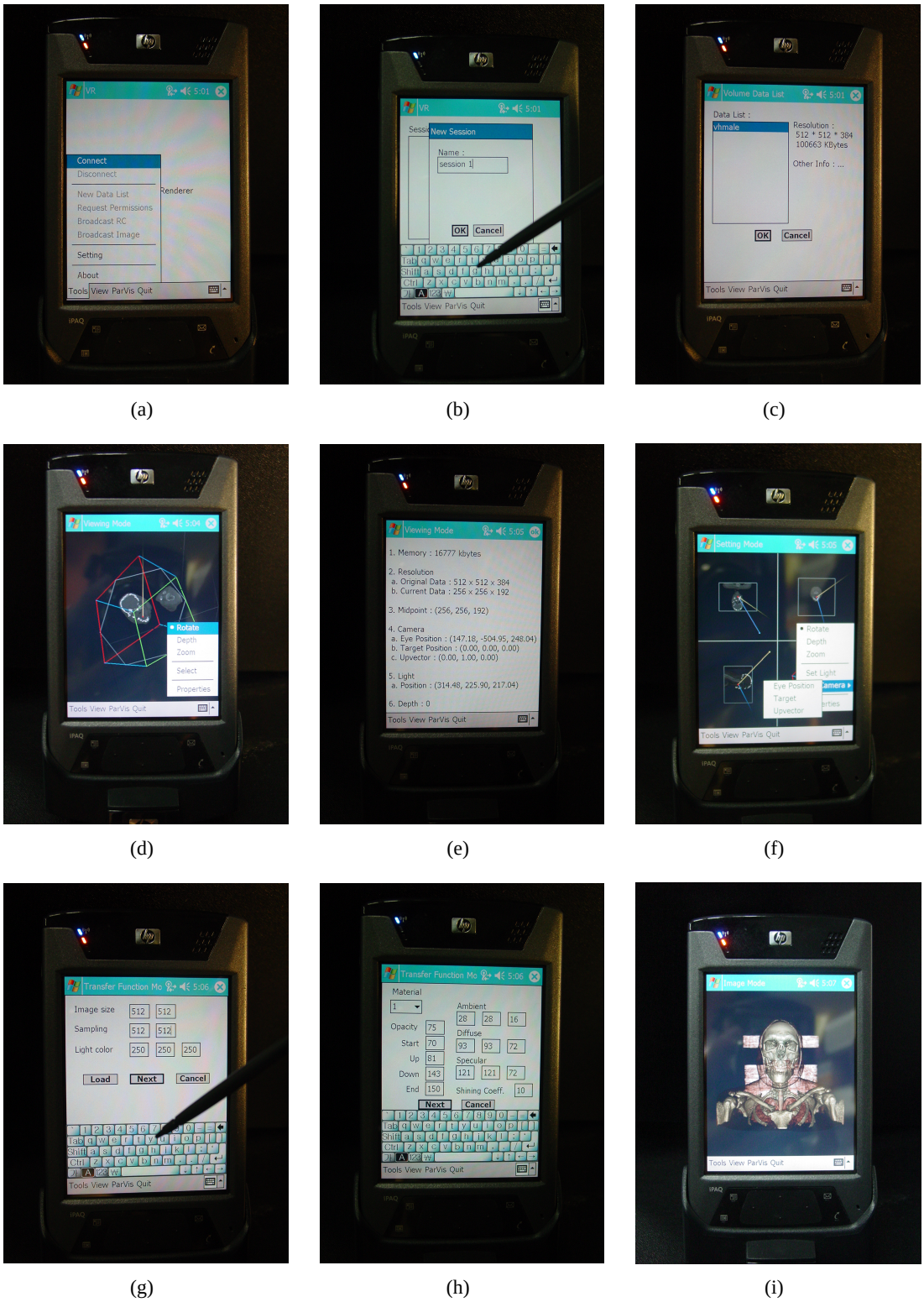
[21] B. Karstens, M. Kreuseler, and H. Schumann. Visualization of complex structures on mobile handhelds. In *Proceedings of International Workshop on Mobile Computing 2003*, June 2003.

[22] S. Goose, S. Guven, X. Zhang, S. Sudarsky, and N. Navab. Mobile 3D visualization and interaction in an industrial environment. In *Proceedings of HCI International 2003*, June 2003.

[23] C. Bajaj, I. Ihm, G. Koo, and S. Park. Parallel ray casting of visible human on distributed memory architectures. In *Proceedings of Joint EUROGRAPHICS-IEEE TCCG Symposium on Visualization*, pages 269–276, May 1999.

[24] Vincent. A 3-D rendering library for mobile devices. http://ogl-es.sourceforge.net/index.htm, 2006.

[25] Argonne National Laboratory. MPICH2. http://www-unix.mcs.anl.gov/mpi/mpich, 2006.

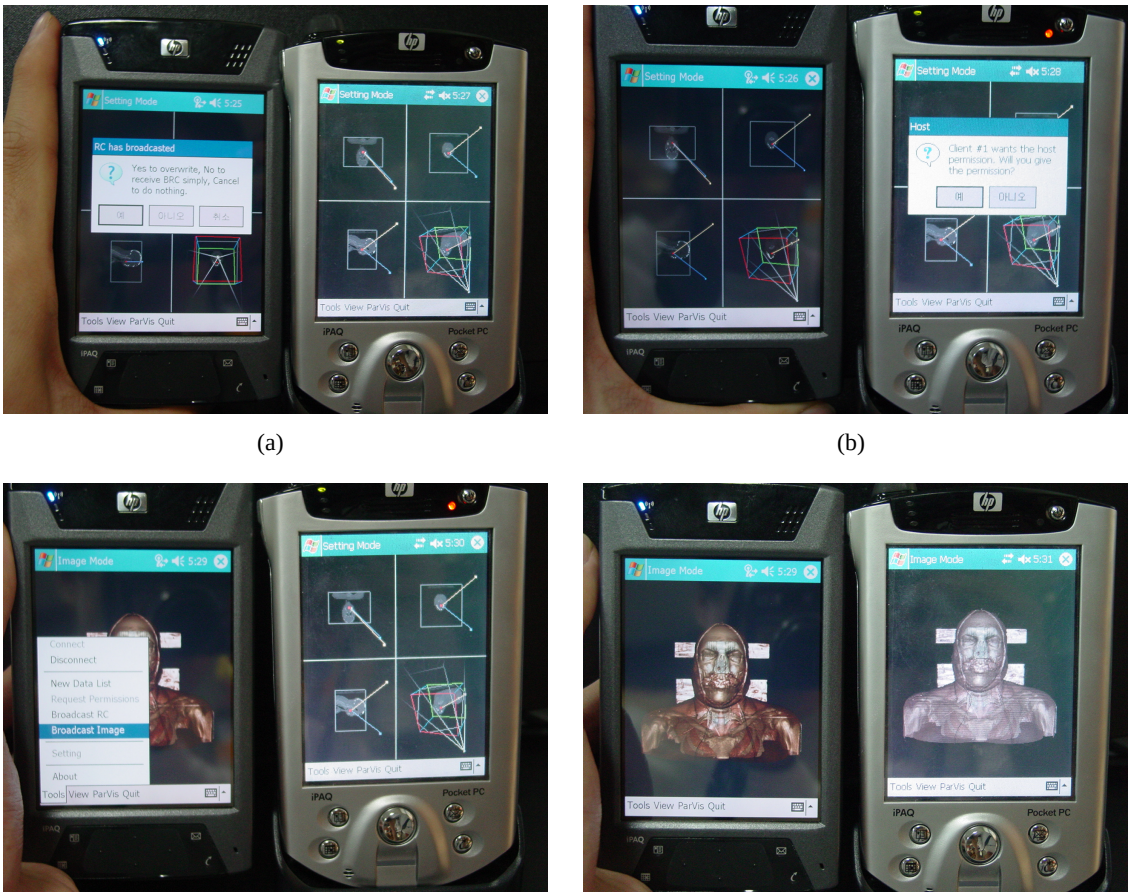


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 9: Example sequence of volume rendering. The detailed explanation is described in the text.



(a) (b)

(c) (d)

Figure 10: Example sequence of CSCW to share rendering contexts and images. See the text for a detailed explanation.

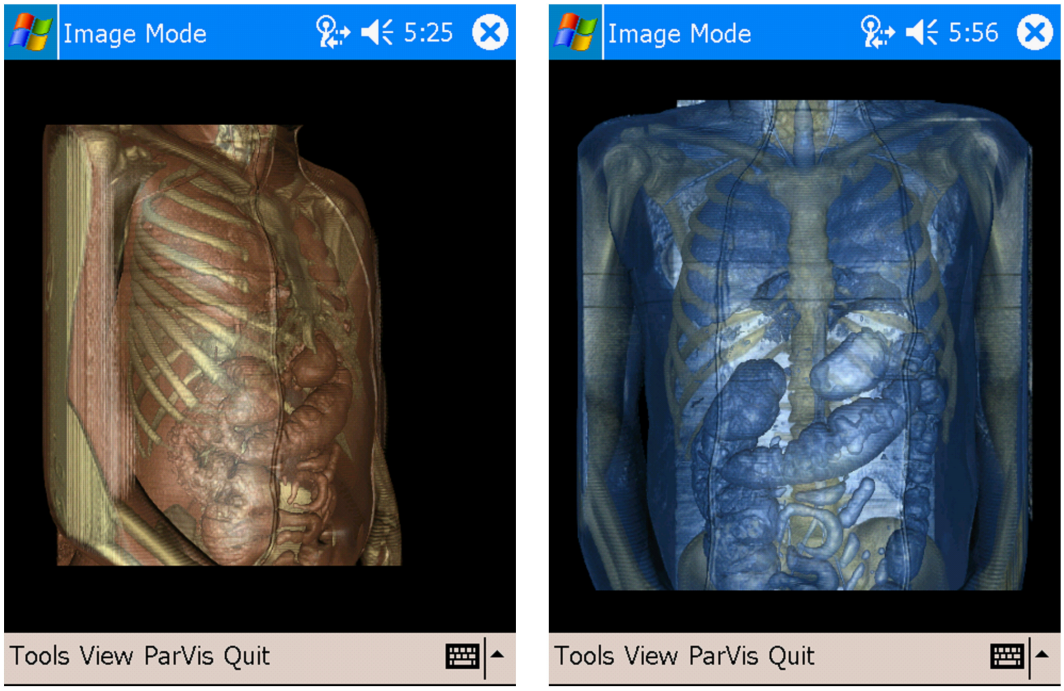
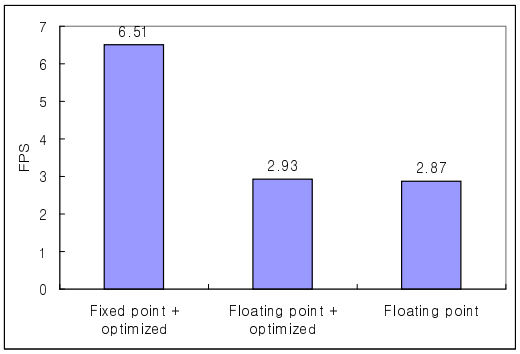
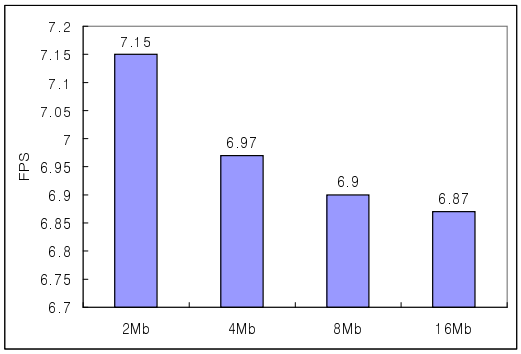


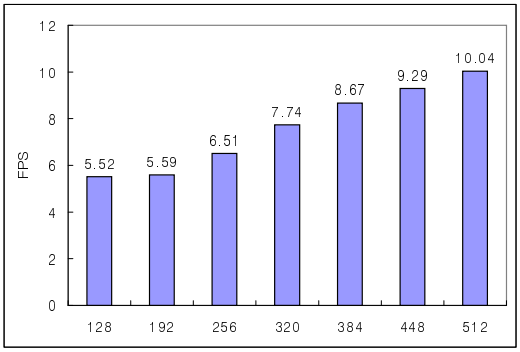
Figure 11: Rendering images of Visible Human male CT data on a mobile client. With the assistance of the presented medical display system, it is possible for a mobile user to interactively examine large medical datasets, possibly working with remote collaborators. The CPU-intensive processing, such as 3D visualization of large sets of raw data, is hidden from clients as such heavy computation is carried out by parallel rendering servers. This transparency makes the clients feel as though they are using a high-end computing system. Note that the time required to display a 3D-visualized image after a rendering request depends on a variety of factors including raw data sizes, classifications, rendering parameters, network traffic, and the computation power of rendering servers.



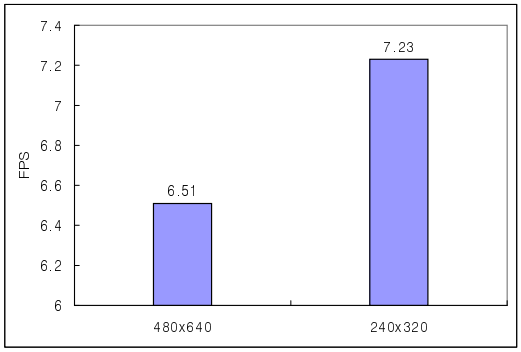
(a) Three optimization efforts



(b) Volume data sizes

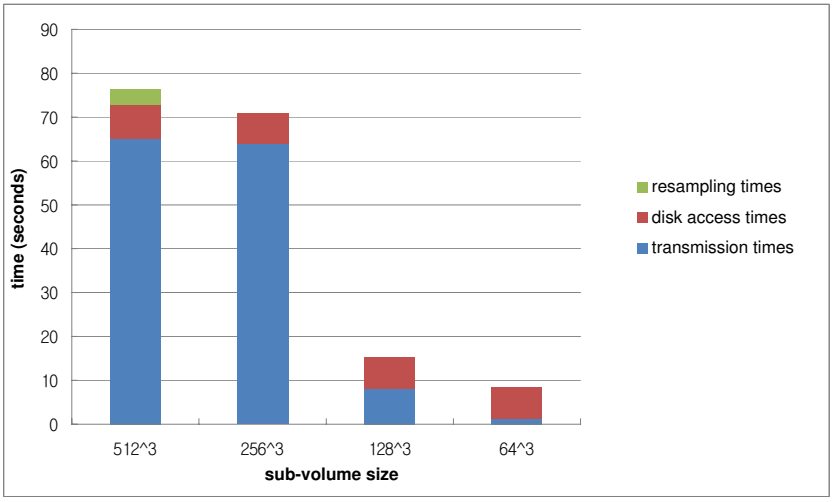


(c) Cutting plane distances from camera

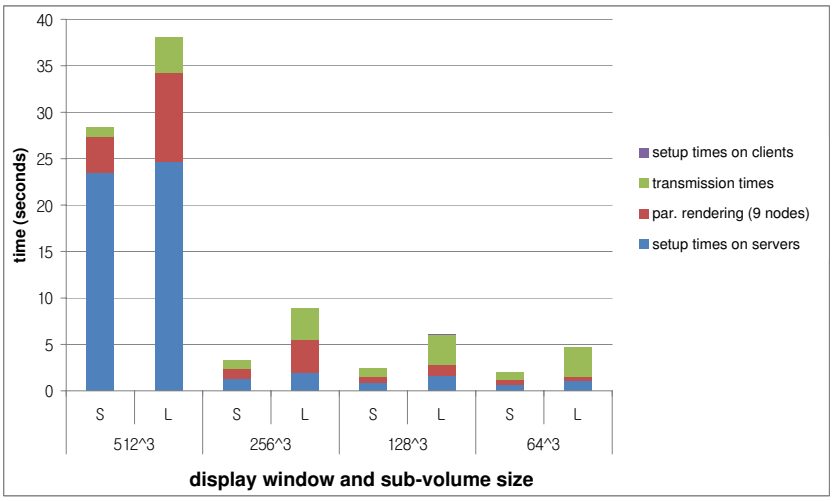


(d) Display window sizes

Figure 12: Performance comparisons for a variety of optimization efforts on Intel PXA CPUs. (a) The frame rate per second (FPS) is compared for the three combinations of applying floating-point/fixed-point numbers and general assembler-level code optimization techniques. (b) As the magnitude of volume data loaded into a PDA with 64 Mb of memory increases, the frame rate deteriorates. However, the difference is not substantial so long as the demanded memory size stays in the 2–16 Mb range. (c) As the cutting plane goes farther, indicated by the distances in the horizontal axis, the screen area to be colored gets smaller, enhancing performance. (d) Together with the result in (c), this graph indicates that the pixel fill rate of the current PDAs is still a problem, although the fill process is expected to be assisted by an acceleration chip in the near future.



(a) Level-of-detail visualization: volume resampling and transmission for different sizes of subvolumes



(b) Remote parallel ray-casting for different sizes of display windows (S: 240×320 pixels, L: 480×640 pixels) and subvolumes

Figure 13: Another performance analysis showing the effect of volume and screen sizes