Chung Yuan Journal, Vol. 30, No. 3 / 2002, September / p. 411-420

Distributed Virtual Environment for Volume Based Surgical Simulation via the World Wide Web

MING-DAR TSAI AND CHUNG-SHYAN LIU

Department of Information and Computer Engineering
Chung Yuan Christian University
Chung Li, Taiwan, R.O.C.

MING-SHIUM HSIEH

Orthopaedics and Traumatology Department, Taipei Medical University Hospital,
Taipei Medical University, Taipei, Taiwan, R.O.C

(Received: May 6, 2002; Accepted: July 31, 2002)

ABSTRACT

Educating and training a surgeon requires much time because a surgeon has few opportunities to rehearsal surgical modalities on patients. This study proposes a low-cost and widely-available distributed VR architecture via the WWW to solve this problem. In this architecture, Web pages from the server drive VR I/O devices on clients to enable users to immerse in a virtual environment. This process is different from the conventional approach that the server is responsible for most works and devices. Methods of driving the devices on clients and being called by the Web page are introduced herein. This paper focuses on the application of volume based surgical simulations that is usually considered involving much computation and expensive workstations. A system that uses PC platforms and the WWW media was built to support the distributed VR surgical simulation. This paper introduces the algorithms for surgical simulation, isosurface reconstruction and rendition, and the methods by which clients can cooperate with the server. A simulation example of musculoskeletal surgery is present.

Key words: *Volume visualization, 3D image based surgical simulation, WWW, VR.*

I. Introduction

Today, rapidly growing numbers of servers broadcast their works as hypermedia documents to allow low cost public access through the World Wide Web (WWW). At the same time, Web browsers have been ported on almost computer platforms for reading the HTML files [1]. Instead of only text or static graphics, a number of servers have enriched their Web pages by using Java or VRML to add user interactions or describe complex 3D scenes and object behavior [2, 3]. Moreover, some servers provide access to devices on server sites to drive the access. For example, Goldberg et al. reported a robot manipulator being operated through the WWW [4]. Therefore, the medium of the WWW no doubt offers great,opportunities in many applications, such as surgical simulation, that require driving devices on server or client's sites.

A surgical simulation system takes physical data from individual patient to make a simulation that help plan and rehearse surgical procedures both for verifying and teaching procedures of an operation. Surgical simulations combining with virtual reality technology provides clinicians computer-based generation of 3D visual and even tactile environment to allow surgeons to immerse, navigate and interact with virtual patients. Such Virtual Reality (VR) simulation can duplicate the operation field and thereby enhance training and reduce the need for expensive animal training model [5]. However, in order to produce a sense of realism for medical purposes, developers bare faced with providing organ fidelity, interactivity, physical and physiological properties of organs and sensory input. The integrated package requires large amounts of computational power, which has traditionally come from mainframes (e.g., [6-10]). This is considered as a drawback to promote VR system to practical uses in surgical simulations. But the economic barrier to the uses of VR in surgical simulations will be much lessened because of the price dropping in CPU, RAM and 3D graphics accelerator. VR surgical simulations begin to work on products based on lower-cost desktop systems [11,12].

This paper presents the possibility of using the WWW to simulate volume based musculoskeletal surgery in a virtual environment (VE). A medical volume consists of computed tomography (CT) or magnetic resonance imaging (MRI) sections that are frequently employed to visualize interior anatomies and have already become a standard procedure for evaluating a complicated musculoskeletal surgery. Surgeons must plan and manage such complicated musculoskeletal surgery, and verify their plans. Therefore, volume manipulation systems for simulating musculoskeletal surgery have received considerable attentions [13, 14]. Among methods of manipulating a volume in surgical simulations, our reported method superiors other ones in operating a 3D image (virtual patient) through some VR I/O devices (tracker and shutter glass) as actual procedures on a real patient, and ensuring the accuracy of anatomic morphology in interactive responses [15-17].

This study extends our reported to use the HTML language to provide a "point-and-click" interface for simulating musculoskeletal surgery in a VE, meaning that users can use VR surgical instruments to operate on stereographic images. The scenario is that the server provides software packages and patient's data over the WWW, while users simulate surgery by using VR devices such as a shutter glass, a head mount display, a glove and a tracker equipped on a client. Thus the client can download the software packages through WWW to implement VR surgical simulation. The client can operate the patient's data but do not need hold or maintain the data. The server serves the computation and maintains the patient data. As the result, it becomes possible to communicate with multiple clients but discuss a single patient. This paper describes a novel approach that allows a WWW site to access devices on clients to simulate VR surgery. This research provides a shared virtual environment that multiple surgeons can incorporate decision making for an actual surgical intervention or a rehearsal through the WWW. The server provides a surgical simulator that manipulates medical volumes and reconstructs surfaces from these volumes. While, PC clients are responsible for the rendering computation to generate 3D shaded images and stereographic images of different perspectives, and should be equipped with a shutter glass to observe the stereographic images and a tracker to simulate the surgical procedures.

Two problems should be solved to achieve such scenario. First, isosurface reconstruction and transfer of a volume usually takes much time letting real time or even interactive responses impossible. Although transforming a volume as frequency-domain data then transferring or visualizing the data can save time [18, 19], visualization of a frequency-domain volume is not realistic to reveal clear anatomies for diagnosis. Surface rendering techniques, which use sample points to reconstruct triangulated isosurfaces, can achieve good visualization of the surfaces of anatomic structures. However, such isosurface reconstruction suffers the disadvantage of being time-consuming [20]. Because a volume manipulation usually only involved small part of a medical volume in simulations of a surgical modality (a set of certain surgical procedures for achieving some surgical purpose), a volume can be divided into several subvolumes. Thus only the isosurfaces of manipulated subvolume must be reconstructed to save the isosurface reconstruction time. Second, the WWW program provided by the server has to drive the devices on the client. Several techniques such as the ActiveX control techniques that follow the protocol of COM (Component Object Model) by Microsoft Inc. can satisfy this requirement [21]. Herein, we use JNI (Java Native Interface) methods to drive the devices because the Java language is rather open [22].

The prototype system can be used to educate and train residents and students who can operate several different surgical modalities to show their effects for correcting the same musculoskeletal deformity and get more real ideas about the whole operation and each procedure of a surgical modality trough the VR interactions. Visiting doctors also use the system to rehearse surgical modalities before an operation, confirm and discuss surgical plans, or try new modalities. Because of using the WWW media that is familiar to most people, it is easy to use. This study uses PC platforms and VR devices on PCs. Such system is cheap to set up. Section 2 introduces the structure of the prototype system. The third until the six sections introduce several modules of the system. Section 7 introduces an implementation example. Conclusion remarks are made in Section 8.

II. System Architecture

Figure 1 shows the system architecture. The system includes four modules at each client. The interface module and the socket module are implemented by a Java applet. The hardware on the client can be accessed from the JNI by wrapping the native codes that drive the devices. The native codes in the rendering and tracker modules are written in C++.

The interface module interprets and processes every command by a user. After a client has downloaded Web pages of our system through the Internet, a Java applet of the interface module and the socket module begins. Then, the VR devices on the client are initialized and a socket of TCP protocol is created to connect with the server. Section 3 describes the interface module that provides a panel through which a user can input various parameters and commands.

The rendering module uses the OpenGL library to drive 3D graphics accelerators. Even an inexpensive graphics accelerator can now render millions of triangles in one second. The triangulated isosurfaces from the server are rendered at this module. The tracker module detects the data of the positions and angles of the tracker, and then sends the data to the server to simulate the surgery. The two modules use the JNI to wrap device drivers. Section 6 and 4 introduce the rendering and tracker modules, respectively.

The system includes four modules on the server. The triangulation module reconstructs triangulated isosurfaces from medical volumes using the marching cube algorithm [15]. In this module, a cuboid volume can be divided into several smaller cuboid subvolumes to achieve interactive rendering. (It is reported a clinician can wait patiently only for 2 seconds for a response [23].) This module independently reconstructs isosurfaces of each subvolume. Because most part of a medical volume is not changed during simulation, the server does not need to reconstruct the triangles of the whole volume. Efficiency can be improved if only reconstructing the triangles of the subvolumes changed during simulation. A clinician can divide the volume as some subvolumes based on clinical knowledge that enables only one subvolume is operated during the simulation of one surgical modality. For example, a skull for orthogonathic surgery simulations can be divided into the brain, mandible or maxilla because surgeons usually operate only on the maxilla or the mandible under one surgical modality of the orthogonathic surgery.

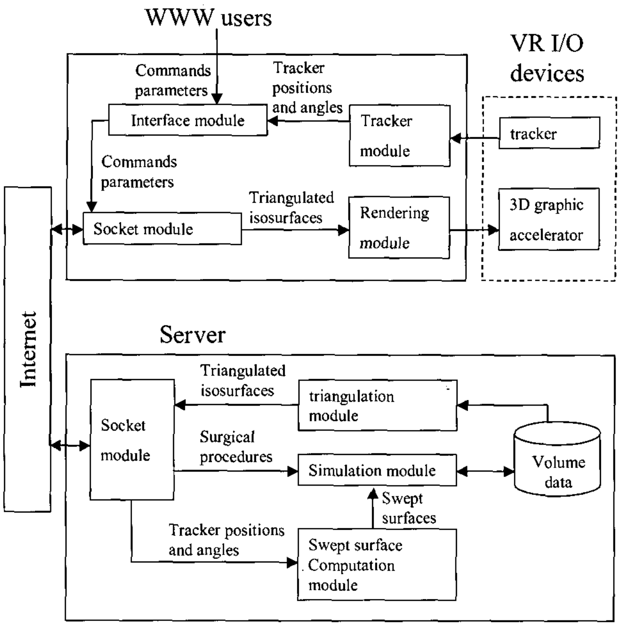


Fig. 1. System architecture

The server provides the functions of manipulating volume data to simulate surgeries of the musculoskeletal system. Surgeons in orthopedics, oral and maxillofacial departments use the surgeries to solve functional and aesthetic problems caused by skeletal deformities. The simulation system uses a voxel structure to represent the topology and geometry of a solid's surfaces. In this data structure, a voxel has 6 face-flags and distance-levels that can be used to improve rendering speed and quality and enables the closure check for the intersection of surfaces and a solid, and thus makes various manipulations on the solid feasible. The simulated results of every surgical procedure of the surgical modalities can be used to impress surgeons to show how the bone is opened, corrected and closed. Surgeons use these function to operate on computer-generated patients as the way of the actual surgical procedures: sectioning anatomic structures of bone and holding (recognizing) separate anatomic structures, removing: translating and rotating separate structures, fusing and healing up separate structures and associated soft tissues. Regarding techniques of simulating musculoskeletal surgeries, our previous reports discuss the techniques [15-17].

The swept surface computation module computes swept surfaces of the instrument based on the positions and angles of the tracker attached on the instrument. The swept surfaces are then used in the simulation module to implement the section simulation. Section 5 introduces the swept surface computation module. When a user is operating a volume, another user can request to register as an observer. The observer cannot operate on the same volume but can see the operation performed by the origin user. The socket module on the server records all observers in a socket list. The server will send the reconstructed isosurfaces to the clients of the operator and all observers. The modules on the server are written in C++.

III. Interface Module

Figure 2 shows a Java applet of an interface panel implemented by the interface module. By pointing and clicking on the panel, the user can send one of the following commands: choose a volume, initialize this volume, set parameters of the shading model, render the volume, and simulate surgical procedures (section, recognize, remove, translate, rotate, fuse and heal up).

If a user chose a new volume, the server sends a pair of pre-rendered stereographic images of the volume to be displayed on an applet frame. The stereographic images have the origin and three primary axes of the volume. If another user also chooses this volume, he can only receive reconstructed triangles of the operated volume to see the operation result. The original user can initialize the volume by using the tracker to match the coordinate system (the origin and primary axes) of the volume on the stereographic images. The user can then divide the volume. The client then transfer the division information to the server that then reconstructs triangulated isosurfaces of bones for each subvolume and sends them back to the client for rendering. The server also reconstructs isosurfaces of the skin if requested by the user. The user can also change the parameters of the rendering models including materials (colors for types of isosurfaces), lights, perspectives, viewing conditions and the shading model (The OpenGL libraries use the Phong shading model that is simple but requires users to adjust its parameters to improve the shading.).

After the initialization of a volume, the user can choose one of the surgical commands (section, recognize, remove, translate and rotate, fuse and heal up) to operate on the volume by using the tracker. Because all simulations need the tracker positions as parameters, the client sends the positions and angles of the tracker and the command type to enable the server to simulate a corresponding surgical procedure. Then, a network event occurs when the server send back the triangles of the operated subvolume for refreshing the result of the surgical simulation. However, even when no commands requested from the user, a timer event occurs in which the client read the tracker's position and attitude and then draw the surgical instrument that the tracker represents on the applet frame. Figure 3 shows the timer event.

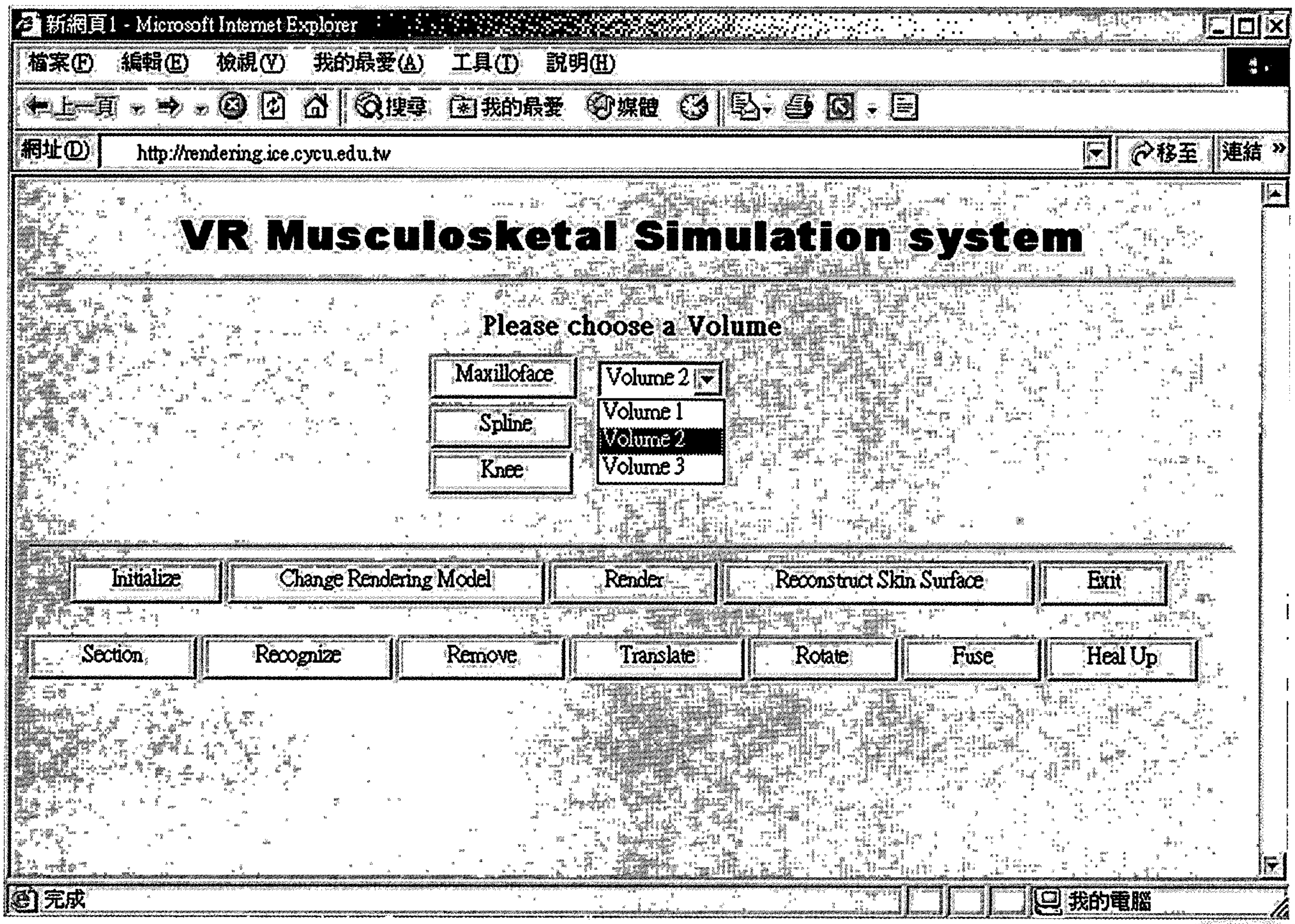


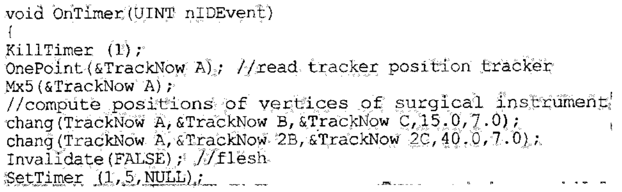
Fig. 2. User interface panel of the system

IV. Tracker Module

Figure 4 shows the relation between the tracker module and the Java applet of the interface module. Through the JNI interface, a tracker point data including three coordinates and three angles are delivered to the Java applet in an event object after a Java method requests. The Java applet may use the data to deal with one of the commands including initialization of a volume and simulations on the volume (section, recognize, remove, translate and rotate, fuse and heal up). The Web page can access any tracker if the native codes for the tracker are available and wrapped by the JNI already.

The InsideTRAK tracker made by Polhemus Inc. is currently available. The tracker is actually a transmitter that generates a magnetic field to detect the coordinates and an angular attitude of a receiver attached to a surgical instrument. Through a specific I/O port, the tracker module uses the native functions of "start", "stop" and "getpoint" to initialize, close and request the tracker to get one point data. Then, the tracker sends raw data of the positions and angular attitude of the receiver through another specific I/O port. The module interprets the raw data as a tracker point including three coordinates and three angles. Although this tracker can provide data of continuous points of the receiver, we let the module accept the data as discrete points at 10ms intervals. Under the same specific I/O ports, the tracker can work for any PC platform.

Figure 5 shows the pseudo code of the "getpoint" function. The connection with current data flow is first terminated to begin a new one to correctly obtain one point data. The header data is neglected to interpret the successive three position and three rotation data.



```
void OnTimer(UINT nIDEvent)
{
KillTimer (1);
OnePoint(&TrackNow A); //read tracker position tracker
Mx5(&TrackNow A);
//compute positions of vertices of surgical instrument
chang(TrackNow A,&TrackNow B,&TrackNow C,15.0,7.0);
chang(TrackNow A,&TrackNow 2B,&TrackNow 2C,40.0,7.0);
Invalidate(FALSE); //flesh
SetTimer (1,5,NULL);
```

Fig. 3. Timer event

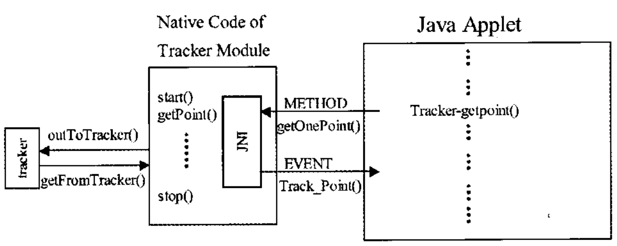
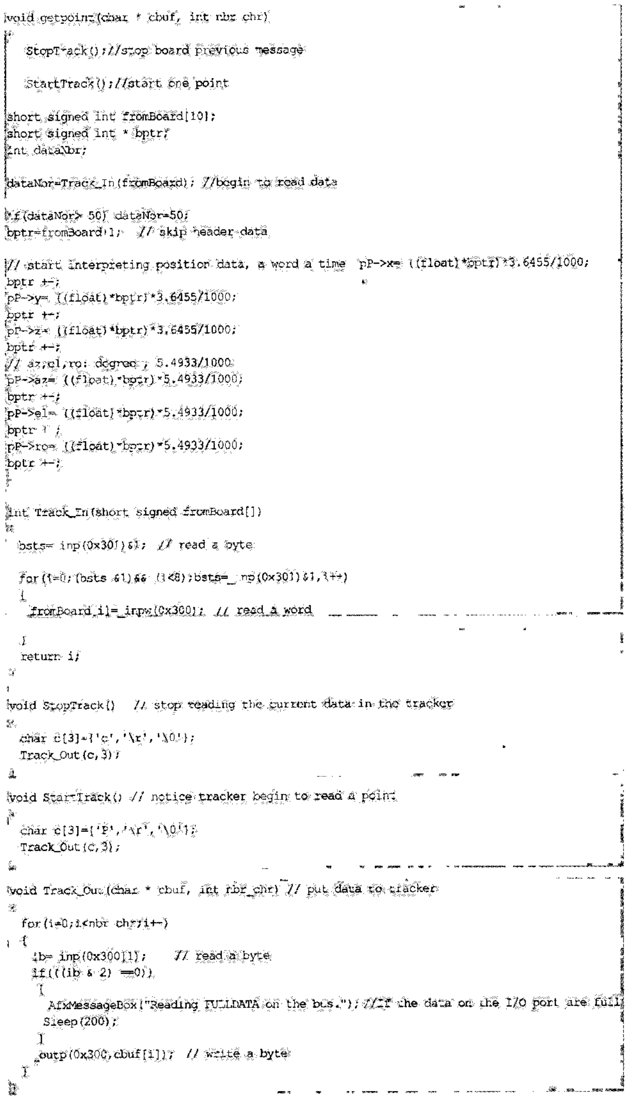


Fig. 4. Data flow of tracker for acquiring tracker points

V. Swept Surface Computation Module

A tracker's position can be used to identify an anatomic structure or to determine its position in the surgical simulations of recognizing separate bones, repositioning a bone, and fusing or healing up separate bones or soft tissues. For example, in the translation simulation, one tracker's position identifies a structure at some position and a successive position may indicate a new position that the structure should be repositioned.

However, in the section simulation, the tracker's positions refer to simulate a surgical instrument and its swept surfaces. As illustrated in Figure 6, the tracker is attached on one end (*a*) of some instrument. The surface normal of the instrument can be obtained by the angular attitude of the tracker. The other end (*b* and *b*')



```
void getpoint(char * cbuf, int nbr chr)
{
   StopTrack();//stop board previous message

   StartTrack();//start one point

short signed int fromBoard[10];
short signed int * bptr;
int dataNbr;

dataNor=Track_In(fromBoard); //begin to read data

if(dataNor> 50) dataNor=50;
bptr=fromBoard+1;  // skip header data

// start interpreting position data, a word a time  pP->x= ((float)*bptr)*3.6455/1000;
bptr ++;
pP->y= ((float)*bptr)*3.6455/1000;
bptr ++;
pP->z= ((float)*bptr)*3.6455/1000;
bptr ++;
// az,el,ro: degree , 5.4933/1000
pP->az= ((float)*bptr)*5.4933/1000;
bptr ++;
pP->el= ((float)*bptr)*5.4933/1000;
bptr ++;
pP->ro= ((float)*bptr)*5.4933/1000;
bptr ++;
}

int Track_In(short signed fromBoard[])
{
  bsts= inp(0x301)&1;  // read a byte

  for(i=0;(bsts &1)&& (i<8);bsts=_inp(0x301)&1,i++)
  {
   fromBoard[i]=_inpw(0x300);  // read a word

  }
  return i;
}

void StopTrack()   // stop reading the current data in the tracker
{
  char c[3]={'c','\r','\0'};
  Track_Out(c,3);
}

void StartTrack() // notice tracker begin to read a point
{
  char c[3]={'P','\r','\0'};
  Track_Out(c,3);
}

void Track_Out(char * cbuf, int nbr_chr) // put data to tracker
{
  for(i=0;i<nbr chr;i++)
  {
    ib= inp(0x300[1]);     // read a byte
    if(((ib & 2) ==0))
     {
      AfxMessageBox("Reading FULLDATA on the bus."); //If the data on the I/O port are full
      Sleep(200);
     }
    outp(0x300,cbuf[i]);  // write a byte
  }
}
```

Fig. 5. Pseudo code for acquiring the data of a turacker point

of the instrument is computed by the position of the tracker, α (the angular attitude of the tracker) and S (the length of the instrument). Figure 7 shows the pseudo code of changing a position data of a tracker to an instrument data. We use triangles to approximate the swept surface of the instrument. As the example shown in Figure 6, two approximate triangles can be obtained from four end points (*a*, *b*, *c* and *d* in Figure 6) of the two positions of the instrument.

The swept surface is then used to intersect with the patient's volume constituted from the patient's CT or MRI slices. In our simulation system, the number of intersected voxels that the system can calculate is also used to limit the feeding rate of the cutting edge to ensure real-time responses. The computation of the intersections of the swept surfaces with bones is discussed in [16].

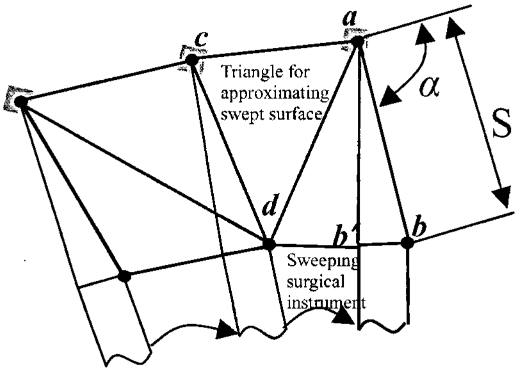
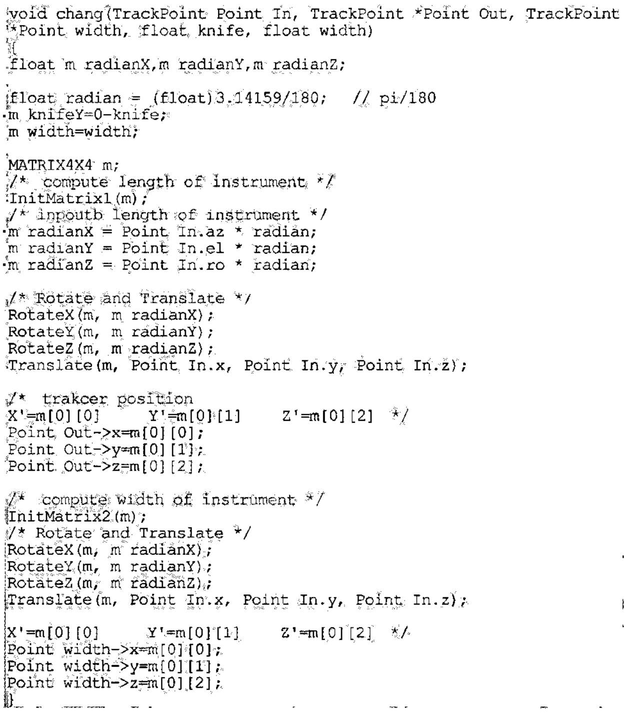

a
c
Triangle for approximating swept surface
α
S
d
b'
b
Sweeping surgical instrument

Fig. 6. Computation of a tool instrument and its swept surface



```
void chang(TrackPoint Point In, TrackPoint *Point Out, TrackPoint
*Point width, float knife, float width)
{
float m radianX,m radianY,m radianZ;

float radian = (float)3.14159/180;   // pi/180
m knifeY=0-knife;
m width=width;

MATRIX4X4 m;
/*  compute length of instrument */
InitMatrix1(m);
/* inpoutb length of instrument */
m radianX = Point In.az * radian;
m radianY = Point In.el * radian;
m radianZ = Point In.ro * radian;

/* Rotate and Translate */
RotateX(m, m radianX);
RotateY(m, m radianY);
RotateZ(m, m radianZ);
Translate(m, Point In.x, Point In.y, Point In.z);

/*  trakcer position
X'=m[0][0]      Y'=m[0][1]      Z'=m[0][2]  */
Point Out->x=m[0][0];
Point Out->y=m[0][1];
Point Out->z=m[0][2];

/*  compute width of instrument */
InitMatrix2(m);
/* Rotate and Translate */
RotateX(m, m radianX);
RotateY(m, m radianY);
RotateZ(m, m radianZ);
Translate(m, Point In.x, Point In.y, Point In.z);

X'=m[0][0]      Y'=m[0][1]      Z'=m[0][2]  */
Point width->x=m[0][0];
Point width->y=m[0][1];
Point width->z=m[0][2];
}
```

Fig. 7. Pseudo code for computing instrument from one tracker point

VI. Rendering Module

The rendering module renders the reconstructed triangles of the sub volumes transferred from the server. The user's commands including rendering triangles of specific subvolumes, changing parameters of the shading model and viewing conditions are written in a Java method that calls the OpenGL functions in the rendering module through the JNI. Then, the OpenGL functions drive the 3D graphics accelerator to render the triangles as illustrated in Figure 8. OpenGL is standard and available for any computer platform; therefore the Java method can work on any computer. The program architecture and libraries of OpenGL are described in [24]. Herein, Figure 9 shows the pseudo code that divides a screen as two viewports for displaying the two shaded images. The two images, one for each eye, generate the stereographic.

VII. Implementation Example

The prototype system uses a Pentium-IV 1.4 G server with 1.5G RAM. One PC client is a Pentium-IV 1G with 512M RAM and a 21" monitor. The client is also equipped with a shutter glass (CrystalEyes PC by StereoGraphics) and a tracker (InsideTRAK by Polhemus) that cost about \$1,500.

Figure 10 shows how a resident uses a surgical instrument to which a tracker is attached. He is initializing the volume coordinate system by matching the origin and the axes of the volume (a 256×256×45 CT volume). Figure 11 shows the images that form the pair

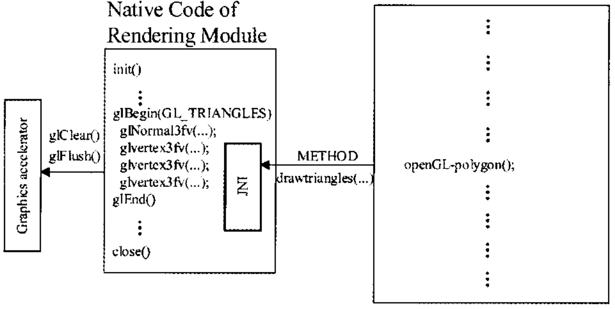
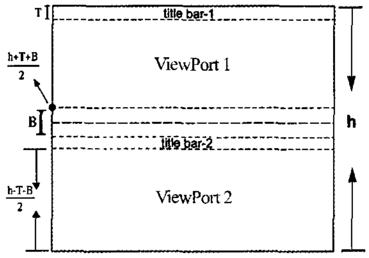


Fig. 8. Data flow of reconstructed triangles for rendering stereographic images

of stereographic images in Figure 10. The two pre-rendered images with the origin and primary axes of the volume were rendered under the same parameters of the shading model except different the eye positions; one for left eye (upper image), the other one for right eye (lower image). These two images can be zoomed into the full screen simultaneously when the user is wearing the shutter glass to observe the stereographic. During the transfer of the stereographic images, the server reconstructs the triangulated isosurfaces of bones and sends them to the client. It takes near 10 seconds to reconstruct 338,156 triangles at the server site. A triangle needs theoretically 6 floats: three pairs of vertices and their surface normals. However, several triangles may be grouped into a triangle strip in the marching cube isosurface reconstruction algorithm wherein one vertex may be shared by two or three triangle. Therefore, the 338,156 triangles take about 5Mbytes and can be transferred to the client in 1 second in a LAN. At the client site, it takes near 1 second to generate a pair of stereographic images for these triangles meaning that a satisfactory response if the user change a perspective to observe the skull (requiring re-rendition of the stereographic images). Therefore, the clinician can easily observe anatomies of the skull by changing perspectives (eye positions and attitudes).



```
void GLResize(GLsizei w, GLsizei h)
{
  glViewport( 0, (h+T+B) / 2, w, (h-T-B)/2 ) ;
  glViewport( 0, 0, w, (h-T-B)/2 ) ;
}
glViewport(GLint x, GLint y, GLsizei width, GLsizei height)
x -> GLint: The number of pixels from the left-hand side of the
        window to start the viewport.
y -> GLint: The number of pixels from the bottom of the window to
        start the viewport.
width -> GLsizei : The width in pixels of the viewport.
height -> GLsizei : The height in pixels of the viewport.
```

Fig. 9. Window division for stereographic images

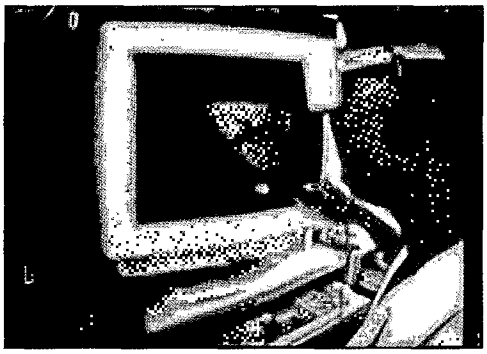


Fig. 10. Implementation of surgery simulation under virtual environment with a shutter glass and a tracker

The user then implements a volume-division command for two reasons: improving the processing time for isosurface reconstruction, rendition of stereographic images and surgical simulation, and easily focusing on the subvolume he want to operate. Figure 12 shows a dialog window (applet) and a divided subvolume (a mandible) that was chosen for further operations. Figure 13 shows that the user has chosen a surgical simulation of a section. Wherein, a tracker simulates a surgical instrument that cuts the mandible. After the section commanded from the client, a series of successive tracker positions are delivered to the server's site to generate swept surfaces that intersect the subvolume of the mandible to simulate the section procedure. The user then requests the remove simulation because the server gives the message that the sectioned structure can be separate (Figure 14). After the simulations of remove, translate, fuse and heal up, the server reconstructs the triangulated isosurfaces of the mandible subvolume because the voxel contents have changed. Because the reconstructed triangles of the sub volume are about 1/10 the

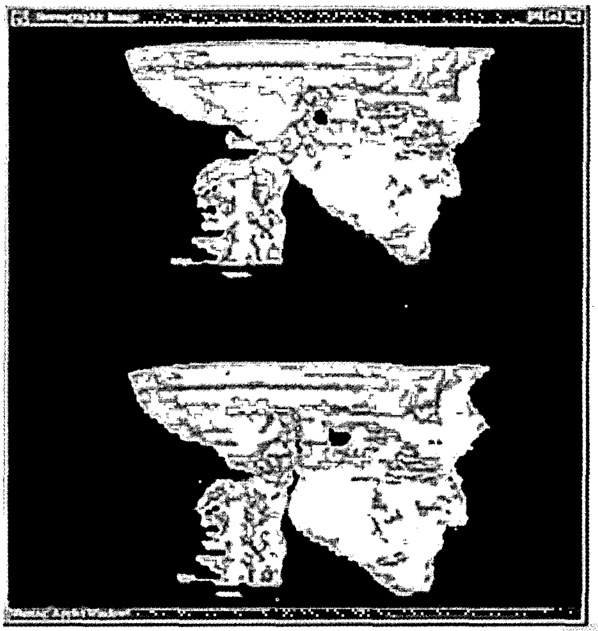
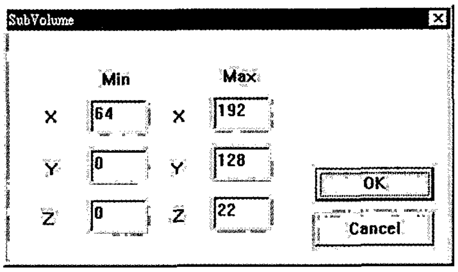
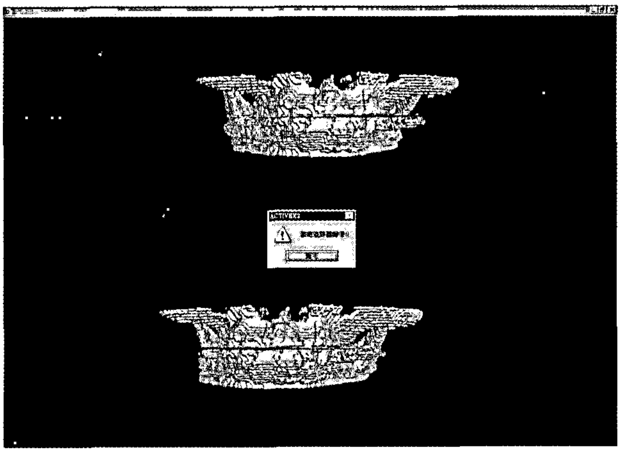


Fig. 11. Stereographic images


SubVolume

	Min		Max
X	64	X	192
Y	0	Y	128
Z	0	Z	22

(a) Dialog for dividing a volume



(b) a subvolume chosen for operation

Fig. 12. Volume division and stereographic images of a divided mandible

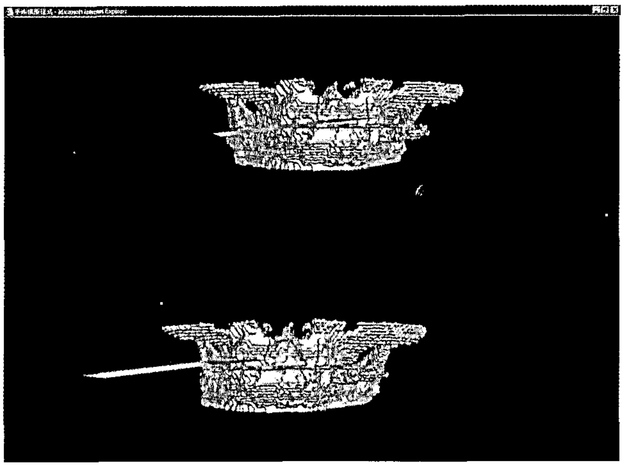


Fig. 13. VR surgical instrument simulated by tracker cutting the mandible stereographic images

triangles of the whole volume, the time for the isosurface reconstruction, transfer of the triangles and rendition for the subvolume all becomes 1/10 of the whole volume.

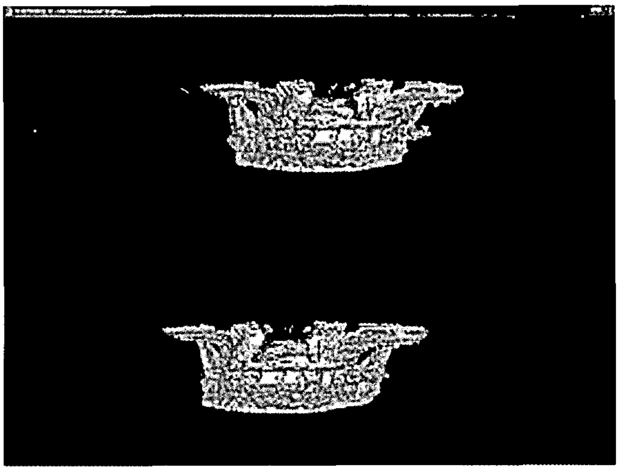


Fig. 14. Part of the mandible sectioned to form a separate structure and removed

VIII. Discussion and Future Works

We proposed a distributed VR surgical simulation system via the WWW. Unlike usual WWW applications, our proposed system can utilize the VR I/O devices on clients to enable WWW users to simulate surgery in virtual environment. Although the transmission time for reconstructed triangles may not offer time-critical responses (the user may have to wait for the result of surgical simulation), the user can still obtain fast interactions in observing stereographic images for different perspectives with a shuttle glass and ordering surgical simulations with a tracker. Although the prototype system is now experimented in a small group of Taipei Medical University Hospital, this research is feasible study of new applications of the Internet. Remote medical students and residents may access the VR resources to experience surgical modalities on textbooks or reports.

Future works include the following three aspects. First, surface compression techniques can add to our system to reduce the amount of the reconstructed triangles. Thus, the time for the transfer and rendition of the triangles can be reduced. However, the demerit by this compression, such as whether the isosurface reconstruction may increase or the rendering quality becomes too bad to reveal critical anatomic information, must be also considered. Second, we should add a haptic function to the system by using force feedback devices on clients [25]. Users can then obtain more realistic feeling when sectioning or repositioning bone structures. Third, isosurface reconstruction usually takes a large amount of computation. Therefore, the server performance must be improved by distributed computing if we want to enable public users (maybe many peoples simultaneously) access our web site. We can also employ the hypermedia inheritance of the WWW to build a computer-aided instruction system that assist surgeons to diagnosis, treat, analysis, planning and rehearse surgical cases [26].

Acknowledgment

This research work was sponsored by National Science Council, a grant number NSC-87-2213-E033-005. This support is gratefully acknowledged. We are also grateful to the visiting doctors and residents of orthopedics, oral and maxillofacial Departments of Taipei Medical University Hospital for testing the system and their helpful suggestions.

Reference

1. Berners-Lee, T., Cailliau, R., Groff, J.F. and Pollerman, B., "World-wide web: The information universe", *Electronic Networking: Research, Application and Policy*, Westport CT, Spring, Vol. 1, No. 2 (1992).
2. Chan, P., *The Java Class Libraries : an Annotated Reference*, Addision-Wesley (1996).
3. Hardman, J. and Wernecke, J., *The VRML 2.0 Handbook, Building Moving Worlds on the Web, Addision-Wesley* (1996).
4. Goldb rg, K., Mascha, M., Gentner, G., Rothenberg, N., Sutter, C. and Wiegley., J., "Desktop Teleoperation via the World Wide Web", *IEEE Proceedings of ICAR'95*, 654 (1995).
5. Ota, D., Loftin, B., Saito, Tim., Lea, R. and Keller, J., "Virtual Reality in Surgical Education", *Computers in Biology and Medicine*, Vol. 25, No. 2, 127 (1995).
6. Bainville, E., Chaffanjon, P. and Cinquin, P., "Computer Generated Visual Assistance During Retroperitonenscopy", *Computers in Biology and Medicine*, Vol. 25, No. 2, 165 (1995).
7. Hunter, I.W., Jones, L.A., Sagar, M.A., Lafontaine, S. R. and Hunter. P.J., "Ophthalmic Microsurgeical Robot and Associated Virtual Environment", *Computers in Biology and Medicine*, Vol. 25, No. 2, 173 (1995).
8. Ziegler, R., Fischer, G., Muller, W. and Gobel, M., "Virtual Reality Arthroscopy Training Simulator", *Computers in Biology and Medicine*, Vol. 25, No. 2, 193 (1995).
9. Kuhlen, T. and Dohle, C., "Virtual Reality for Physically Disabled People", *Computers in Biology and Medicine*, Vol. 25, No. 2, 205 (1995).
10. Hong, L., Muraki, S., Kaufman, A., Bartz, D. and He, T., "Virtual Voyage: Interactive Navigation in the Human Colon", *ACM Computer Graphics*, Vol. 31, No. 5, 27 (1997).
11. "VR in Medicine-Part 1: Current Developments", VR news, Vol. 5, No. 7, 24 (1996).
12. Baatar, Sh., Tani, Sh., Suzuki, F., Kimura, M. and Kanno, T., "Thre-dimensional Visualizational Visualization of EGG topographic Mapping for Power PC-based Personal Computers", *Electroencephalography and Clinical Neurophysiology*, Vol. 99, No. 3, 20 (1996).
13. Satava, R.M., "Virtual Reality and Telepresence for Military Medicine", *Computers in Biology and Medicine*, Vol. 25, No. 2, 229 (1995).
14. Arai, F., Tanumoto, M., Fukuda, T., Shimojima, K., Matuura, H. and Negoro, M., "Distributed Virtual Environment for Intravascular Tele-Surgery Using Multimedia Telecommunication", *IEEE Proceedings of VRAIS'96*, 79 (1996).
15. Tsai, M.D., Hsieh, M.S. and Jou, S.B. "Virtual Reality Orthopedic Surgery Simulator", *Computers in Biology and Medicine*, Vol. 31, No. 5, 333 (2001).
16. Tsai, M.D., Jou, S.B. and Hsieh, M.S. "Accurate Surface Voxelization for Manipulating Volumetric Surfaces and Solids with Application in Simulating Musculoskeletal Surgery", *IEEE CS Pacific Graphics*, Tokyo, Japan, 234 (2001).
17. Hsieh, M.S., Tsai, M.D. and Chung, W.C. "Virtual Reality Simulator for Osteotomy and Fusion Involving the Musculoskeletal System", *Computerized Medical Imaging & Graphics*, Vol.26, No.2, 91 (2002).
18. Totsuka, T. and Levoy M., "Frequency Domain Volume Rendering", *ACM Computer Graphics*, Vol. 27, No. 5, 271 (1993).
19. Gross, M. H., Lippert, L, Dittrich, R. and Haring, S., "Two methods for wavelet-based volume rendering", *Computer & Graphics*, Vol. 21, No. 2, 237 (1997).
20. Lorensen, W. E. and Cline, H. E., "Marching Cubes: A High Resolution 3D Surface Construction Algorithm", *ACM Computer Graphics*, Vol. 21, No. 4, 163 (1987).
21. Denning, A., *ActiveX Controls Inside Out*, Microsoft Press, (1997).
22. Marketos, J., *The Java Developer's Toolkit*, Johnniley, (1997).
23. Chen, L.S. , Chen, J.P. , Chen, S.C. and Liu, P.W., "A Distributed and Interactive Three-Dimensional Medical Image System", *Computerized Medical Imaging & Graphics*, Vol.18, No.5, 325 (1994).
24. Wright, R.S. and Sweet, M. "OpenGL superbible", Waite Group Inc., (1996).
25. Massie, T.M. and Salisbury, J.K., "The PHANToM Haptic Interface: A Device for Probing Virtual Objects", *ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems in Dynamic Systems and Control* Chicago, 295 (1994).
26. Chen, L.S., Liu, P.W. , Chang, K.Y. , Chen, J.P. , Chen, S.C. , Hong, H.C. and Liu,J., "Using Hypermedia in Computer-aided Instruction", *IEEE CG&A*,Vol.16, No. 3, 52 (1996).

全球資訊網上分散虛擬實境環境下操作容積的手術模擬

蔡明達　留忠賢
中原大學資訊工程學系
中壢市普仁22號

謝銘勳
台北醫學大學附設醫院骨科
台北市信義區吳興街250號

摘　要

外科醫師的教育及訓練耗時，因少有機會去對病患做練習。因此我們提出一低價、全球資訊網上可廣被使用的分散虛擬實境架構。在此架構，從伺服器下載的網頁驅動顧客端的虛擬實境用輸出輸入裝置，讓使用者能浸於虛擬環境中。這有別於傳統的伺服器需負責大部分工作及裝置的方法。爲實現這構想，我們用全球資訊網這媒介驅動顧客端的裝置。我們著眼於以容積爲基礎的手術模擬應用，這通常被認爲需大量計算及使用高價電腦。但我們系統使用個人電腦及全球資訊網去達到分散虛擬環境的目的。本論文介紹手術模擬、同質面重建及立體顯像的演算法，和伺服器與顧客端合作的方法，也展示一個骨骼肌肉的手術模擬例。

關鍵詞：容積視覺化，三維影像爲基礎的手術模擬，全球資訊網，虛擬實境。