A Distributed Virtual Reality-Based System for Neonatal Decision-Making Training

A. HOLOBAR, M. DIVJAK, I. PRELOG, D. KOROŠEC, D. ZAZULA

University of Maribor, Faculty of Electrical Engineering and Computer Science, Smetanova ulica 17, 2000 Maribor, Slovenia

Received 3 February 2006; accepted 15 November 2006

ABSTRACT: In many fields of education, there is a significant gap between theoretical study and practical apprenticeship, which training in virtual environments (VEs) can close. Recently, a standalone prototype for virtual reality (VR)-based training in neonatal decision-making, called Virtual Delivery Room (VIDERO) was developed. This prototype enables a simple and time-efficient simulation of the arbitrary postnatal critical situations. The purpose behind this paper is twofold: to outline the role of trainee's immersion into VIDERO environment and to addresses the need for assessment and upgrading of training scenarios. In particular, the distributed services enabling fine-tuning of both virtual patient's behavioral model and trainee's interventions through the practice and user interactions in shared VE are described. © 2007 Wiley Periodicals, Inc. Comput Appl Eng Educ 15: 329–339, 2007; Published online in Wiley InterScience (www.interscience.wiley.com); DOI 10.1002/cae.20120

Keywords: shared virtual environments; distributed systems; virtual reality in education

INTRODUCTION

Medicine is often inherently limited with respect to training possibilities. As more advanced and sophisticated procedures enter everyday medical practice more specialists are faced with a requirement for longer trainings. Experienced medical instructors are usually busy with their own work, so fewer opportunities of such training exist. On the other hand, the trainings and exercises in real situations bring increased risk to the patients, who are expecting an expert treatment instead of novices [1].

Recently, a prototype of a virtual reality (VR)-based system for neonatal resuscitation training called Virtual Delivery Room (VIDERO) [2] was developed. The prototype simulates a real delivery room with all the necessary medical devices (ECG and respiration rate monitor, radiant heater, stop watch, etc.) and instrumentation (laryngoscope, stethoscope, various catheters, etc.), and, in particular, a virtual baby with its most typical vital signs. Running on personal computer platforms, it offers the trainee an excellent opportunity to practice required resuscitation procedures,

Correspondence to A. Holobar (ales.holobar@uni-mb.si).

Contract grant sponsor: Slovenian Ministry of Education, Science and Sport; contract grant numbers: 3411-98-71-0001, S2-796-010/21301/2000, and Programme Funding P2-0041.

their proper order, type and timing. In contrast to other similar simulators, all simulated postnatal complications and trainee's interventions can be predefined in a simple, human readable text files. This enables the medical instructors to create an arbitrary training scenario, set up possible interventions and define the corresponding reactions of the virtual newborn.

The main focus of the work presented in Reference [2] is on simulation of the newborn's vital signs and on scalable definition of the training scenarios and interventions, while the issues of the trainee's immersion into the virtual environment (VE) and his interaction with virtual entities is mainly ignored. Indeed, there have been some studies reporting no special enhancement in students' performance when comparing immersive VEs to desktop simulations [1,3]. On the other hand, several authors suggested that immersive VR represents a promising concept with high potential of enhancing and modifying the learning experience. Ponder et al. [4], for example, pointed out that immersive VE raises trainees' interests and motivation, and increases knowledge transfer. Mantovani et al. [5] underlined that immersive VR provides a rich medium to 'learning-by-doing' through first-person experience and can effectively support skills acquisition. As Lombard and Ditton [6] argue, the immersive VR should provide perceptual illusion of non-mediation and let the trainees suspend disbeliefs, while according to Slater [7] immersive experience should lead to the 'sense of presence' defined as a sense of being there and remembering VE as having actually visited the place. On the other hand, Riva [8] and Mantovani [9] claim that experiencing presence in a clinical VE requires more than reproduction of the physical features, and stress the importance of the sense of community developed by interaction in shared multi-user VEs. An extensive study of the measures enabling evaluation of immersive VE was introduced by Kalawsky et al. [10].

In this paper we describe simple but powerful client−server architecture required to extend the VIDERO prototype to an immersive, distributed VE. The main focus is on distributed services providing: (a) trainee's immersion into VE, and (b) assessment and simultaneous upgrading of training scenarios and trainee's interventions through the natural mentor−trainee interaction in shared VE. To be more precise, this paper describes the mechanisms which enable the logical synchronization between the shared VE scenes, trainee's natural navigation through VE, multi-user audio communication and dynamic configuration of the system components.

Design and implementation of the shared VE have been under investigation for many years and several similar works have already been published. For example, in Reference [11] Green discussed problems of real-time processing and concurrency control in shared VEs. England and Gray [12] examined several aspects of shared VE design, including real-time object interaction, user communications and world coherence versus workload ratio. Fully immersive and multi-modal-distributed VR platform supporting the training of medical first responders was introduced by Stansfield et al. [13]. Throughout this paper, our work will be compared to those VEs which will clarify the potential benefits or drawbacks of the solutions applied.

The paper is organized as follows: in Section VIDERO Prototype the main features of VIDERO prototype are briefly outlined, along with two different viewpoints, the mentor's and the trainee's, of the scalable training session. Section Distributed Virtual Delivery Room discusses the main compromises between the scalability and representativeness of the VIDERO VE and upgrades the standalone VIDERO simulator into a distributed, multi-user-oriented application. Basic performance indices are discussed in Section Performance Evaluation, while Section Discussion and Conclusions concludes the paper.

VIDERO PROTOTYPE

VIDERO is implemented in VRML [14] and Java programming language. It combines a 3D model of a virtual newborn with text-based descriptions of its physiological and behavioral responses in order to enable a simple and time-efficient construction of arbitrary postnatal critical situations. The prototype exhibits scalable and open-structured architecture and demonstrates substantial robustness, responsiveness and efficiency of text-based training scenarios. Following the specifications of H-anim [15], the newborn's body consists of many separate segments (e.g. forearms, hands, feet, etc.) which are connected to each other by joints (such as the elbow, wrist and ankle). Each part of the body is controlled by one or more corresponding vital sign mechanisms. The current version simulates seven most crucial newborn's vital signs: breathing, heartbeat, colour of skin and lips, motion, crying, sobbing and facial mimics [2].

The Need for an Upgrade to a Distributed VIDERO Training System

According to Ponder et al. [4], VR-based training should provide the course instructors with an option to

supervise the entire training session. In order to introduce an on-line surveillance into the VR-based decision-making training, the concepts of a mentor (supervisor) and a trainee must be set first. For the trainee, the assessment of the baby's health condition from the observable parameters (e.g. the heartbeat, respiration cycle, etc.), and the ability to respond adequately are crucial. For realistic immersion into VE, a support for special VR equipment such as head-mounted display and motion-tracking device should be considered. Although this solution may prove highly beneficial in specialized training centres, such as the schools of medicine, which can afford considerable expenses more easily than individuals, it may limit the feasibility of training in home environments. Hence, the navigation and virtual object manipulation must remain possible also when no special navigation device is available.

The mentor, in contrast to the trainee, has to have a total control over newborn's internal parameters. Beside the option of an automatic execution of the scenarios, he must also be able to change the course of events and vary the virtual newborn's symptoms. In addition, the mentor should have a complete overview of trainee's activity. While observing the trainee's responses, he has an excellent opportunity to critically assess the efficiency of the training scenarios and to eventually upgrade them. In this way, all the training parameters can be easily adapted to the trainee's needs and the intention behind the training is optimized through a natural evolution process.

Computer-simulated resuscitation training also puts less pressure on the mentor, especially in the automatic mode where baby's health is controlled by scenarios. It is, therefore, easier for him to monitor and acknowledge the actions taken by the trainee. Nevertheless, according to Taffiner et al. [16] the VR-based teaching tools should facilitate evaluation and assessment of trainee's performance, as every session in VE can easily be recorded for latter examinations. For this reason, a special program module which renders recording and replaying of the whole training session feasible was also introduced to the distributed VIDERO (D-VIDERO) prototype.

DISTRIBUTED VIRTUAL DELIVERY ROOM

Following the requirements stated in the previous section, the VIDERO prototype was upgraded into a distributed, multi-user-oriented application. To preserve its platform independence, all added modules were built by using Java programming language. The system now consists of trainee's and mentor's applications running on a set-up of two or more separate, network-connected personal computers and has an option to interface with a Polhemus 3Space Fastrak motion-tracking device [17] and a head-mounted display. From a user's point of view, it consists of three different modules: one or more mentor applications, one or more trainee applications and a so-called Replayer module.

Each trainee application consists of two parts: (a) VRML/X3D browser's 3D view of dynamic VE and (b) menu-based user interface (Fig. 1). Mentor application (optionally) runs on a remote computer and also consists of: (a) 3D view of VE, and (b) user interface for controlling the parameters of the virtual newborn, starting the training scenarios and receiving notification on the student's actions (Fig. 1). The mentor has an option of locking his view to the trainee's point of view in order to observe the VR room through the trainee's eyes. This enables him to assess the trainee's ability of orientation in VR environment and, possibly, trainee's motion sickness [18]. Later, when the trainee has already adapted to the virtual world, the mentor can use this feature to gather suggestions about the trainee's mental concentration (confusion), stress handling and ability to confront immediately with time-critical complications.

Replayer is a separate program which allows the stored training session to be repeated in real-time, exactly in the way as it was carried out. It loads all the data saved during a training session and reconstructs all the events in VE.

Two basic modes of operation are available from the mentor's point of view: (a) manual, where the mentor controls each vital sign of the virtual baby according to his own judgement, and (b) semi-automatic operation in which each student's intervention is related to the predefined training scenarios. In the latter case, the mentor should prepare these scenarios in advance, but he can always intervene in the session by manually adjusting certain parameters.

Distributed Services

The final configuration of the D-VIDERO environment is case dependent. The system allows several trainees and mentors to be simultaneously present in the same training session. The participants share all newborn's parameters and, optionally, the view of the virtual room. They also communicate with each other, the mentors are immediately notified about all the trainees' actions and the scenario changes are reflected on both sides simultaneously. All this is

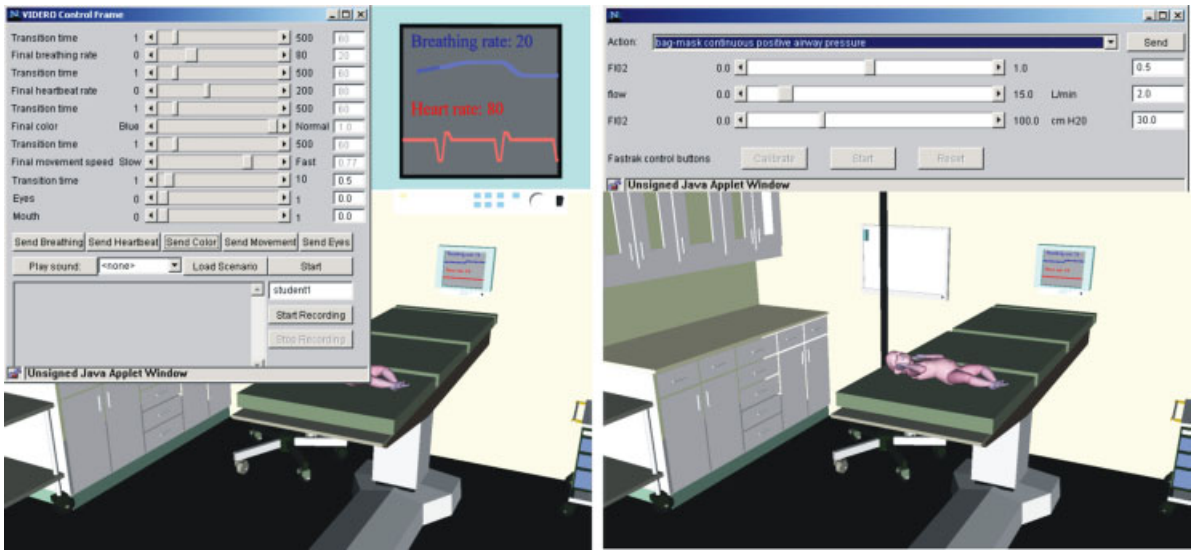

Figure 1 The D-VIDERO prototype with distributed VE (background image) and pop-up control windows (foreground image): mentor application (left) and trainee application (right). Both users can share the same view of the Virtual Delivery Room.

made possible by various distributed services that are incorporated in VIDERO.

Following the VIDERO prototype scalable structure [2], four different services are currently implemented using Java Remote Method Invocation (RMI) [19]. Although this may not be an optimal solution (with respect to both the performance and network load), RMI is simple, quick and reliable. By separating the program interface from the implementation logic, the client−server architecture becomes apt for modular construction, easy upgrade of services and simple adding or removal of different components (switching the services on and off). Moreover, only the RMI clients are included in the mentor's and trainee's application, while the essential part of services is encoded in RMI servers (see Subsections Consistency Control, Navigation in D-VIDERO, Multi-User Audio Communication and Recording and Reconstruction of the Training Session). The mentor's and student's applications can, therefore, easily be connected to, and disconnected from, the central training session. The main limitation induced by RMI is, of course, the responsiveness of the VE over slow network connections. For the time being, this is not a very serious limitation as the main intention of the distributed VIDERO prototype is to evaluate the feasibility of interactive and constant upgrading of the training scenarios by the medical personnel. As shown in Section Performance Evaluation, when running on high-speed networks, RMI provides a reliable and efficient solution.

Consistency Control

General ideas concerning the design and implementation of a VRML-based multi-user layer have been introduced in Reference [20]. However, sharing the VRML scene permits only a partial consistency control. Considering the three-layer implementation of the VIDERO prototype [2], the highest layer is implemented in Java and cannot be synchronized through the mechanisms outlined in Reference [20]. On the other hand, Stansfield et al. [13] considered synchronization through the UDP/TCP packet multicasting. Indeed, this minimizes the workload and network throughput but makes strict time synchronization tricky. With the development of faster networks the benefits of the RMI interface become evident, at least in the prototyping phase.

In the case of D-VIDERO prototype, the consistency of VE is enforced by a RMI server called Control Server. The latter mediates all the changes of the virtual newborn's parameters from the mentor to all the trainee applications, and the applied interventions in the opposite direction (Fig. 2). The Control Server also initiates parts of the predefined training scenarios in response to the mentor's requests.

Whenever the change in the newborn's condition is initiated, the mentor application sends events describing requests for changes to Control Server, which broadcasts them through RMI clients (Control Client) to all the trainee and mentor applications.

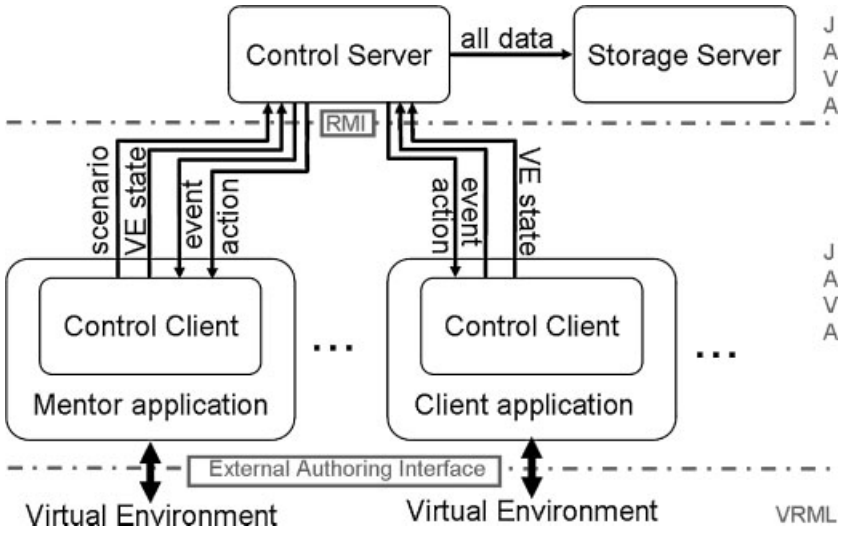

Figure 2 Schema of the D-VIDERO control service: rounded rectangles denote different program modules, thick arrows connecting the rectangles indicate data flows through the External Authoring interface, while the narrow arrows indicate the RMI connections. The superimposed labels denote: event—events carrying the changes of newborn's parameters, VE state—current values of newborn's parameters, action—student's intervention along with its parameters, scenario—the name of the scenario file to be loaded, all data—all data received by Control Server.

Similarly, when the trainee's action is applied, the corresponding Control Client instructs Control Server to broadcast all the accessible data, including the exact time, action parameters and the name of the corresponding scenario files, back to all mentor applications where all data is displayed on the screen (Fig. 1). At the same time, the so-far executed scenario is interrupted and a new scenario file containing a proper description of the newborn's response is loaded by Control Server. Following the concept of text-based definitions, the scenario's conditional branching can be defined in a simple text-based format [2].

Every Control Client has a built-in access to all the crucial VRML mechanisms (current values of all the newborn's parameters). Upon request, it forwards the gathered information to Control Server where it is accessible to other services and possible external applications, e.g. voice synthesis module emulating the assistants (nurses) and their verbal descriptions of vital signs. For this reasons, Control Server has a separate thread waiting for different requests from all possible clients. Each received request is accordingly forwarded to all corresponding receivers.

Navigation in D-VIDERO

The navigation in VE, as well as the synchronization of shared viewpoints, is handled by the so-called Navigation Server. In order to avoid ambiguities in navigation, the shared navigation in 3D virtual world is always controlled by a single predefined trainee user (called a leading trainee in the sequel). All other trainees and mentors share only 1 out of 10 viewpoints that are built in the D-VIDERO environment. Other nine viewpoints are bound to important virtual objects (monitor, infant, radiant heater, medical instrumentation, etc.) and can be employed by other users as starting points for their independent VE explorations.

The current version of Navigation Server has a built-in program interface to communicate with external navigation devices. Sensors with six degrees of freedom (DOF) are preferred, however, less sophisticated sensors can also be used. The Euler's angles and direction cosine angle orientations are supported along with the VRML built-in quaternion orientation representation [14]. The only restriction inherited from the core VIDERO system concerns the implementation platform of the motion-tracking device drivers, which have to be able to communicate with the Java applications. A supporting Java FastrakLib library for the Polhemus Fastrak motion-tracking device [17] has been developed and built in the system.

Basic schema of the navigation system is depicted in Figure 3. Each participant's application has two built-in navigation modules: Navigation Client is an RMI client which constantly updates the position data from the Navigation Server, while the

Viewpoint Handler module moves the VRML viewpoint [14] through VE. The trainee's application has one additional class called Viewpoint Observer. This class tracks the leading trainee's position and orientation as belonging to the avatar in VE and updates the trainee's position data within the Navigation Server when no motion-tracking device is connected.

The Fastrak motion-tracking device and the FastrakLib library support up to four sensors. Whereas the first one is used for the navigation through VE (the sensor is typically attached to the leading trainee's head-mounted display), the second and the third sensor have been employed to support tracking of the leading trainee's dominant hand. The second sensor is attached to the backside of his palm, while the third is fixed to the nail of his index finger. By monitoring the Euclidean distance between the two hand sensors, the clenching and releasing of the trainee's fist can be recognized. This enables simple manipulation of the virtual objects. However, to preserve the feature of defining arbitrary intervention scenarios, all the trainee's actions still have to be chosen from the pop-up menu, which appears at the upper half of the displayed screen (Fig. 1) whenever the trainee raises his dominant hand above his head (above the sensor attached to the head-mounted display). A special module called Mouse Emulator (Fig. 3) connects the mouse pointer to the data from the palm (second) sensor, whereas the sudden rapid move of the index finger (third sensor) is recognized as a mouse click. This enables the trainee to select the proper intervention while freely navigating through VE. The fourth sensor is attached to the palm of the leading trainee's non-dominant hand and facilitates the hands coordination in VE.

Multi-User Audio Communication

Audio Server provides real-time audio conferencing between all the training participants. Each user can freely start or stop the recording of his voice and adjust the volume of the playback. To preserve platform independence, the communication system was implemented using Java Media Framework (JMF) [21]. The basic structure of the communication system is shown in Figure 4.

Both the trainee and the mentor applications use the same communication client connected to the Audio Server. The Audio Client module records the speaker's voice using a microphone. The resulting audio stream is encoded into the preselected format (MPEG, G723, GSM, DVI, μLAW) [21] and, via the Real-Time Transport Protocol (RTP), sent to Audio Server where it is cloned. One copy of the stream is sent to the Storage Server, the other is multi-casted to all the Audio Client modules, where it is decoded into raw audio signals. The Spatial Effects module enhances the speaker's voice with the spatial sound effects in accordance with the distance and orientation

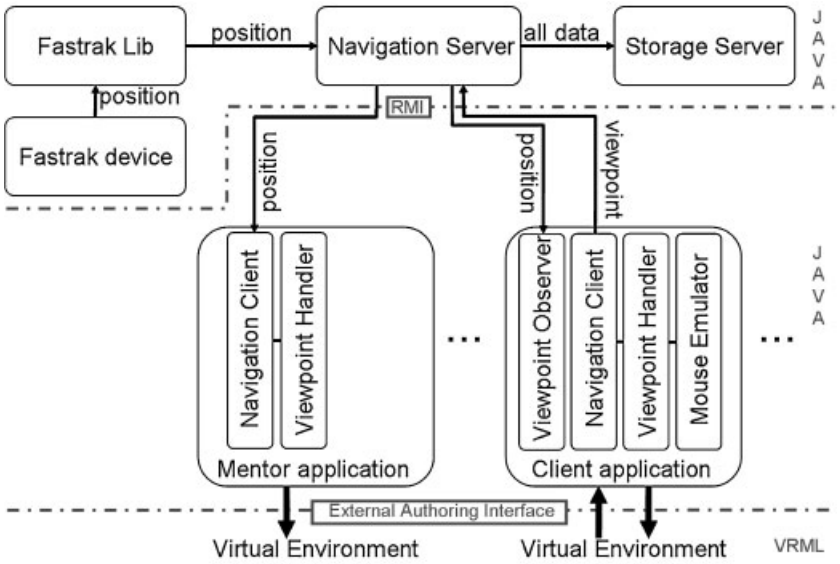

Figure 3 Navigation service in the D-VIDERO prototype: program modules are coded in the same way as in Figure 2. Narrow arrows with the superimposed labels illustrate the RMI calls and data flows: position—tracked position and orientation of the leading trainee's head and hands in the real world, viewpoint—trainee's current position and orientation in the virtual world, all data—all data received by Navigation Server.

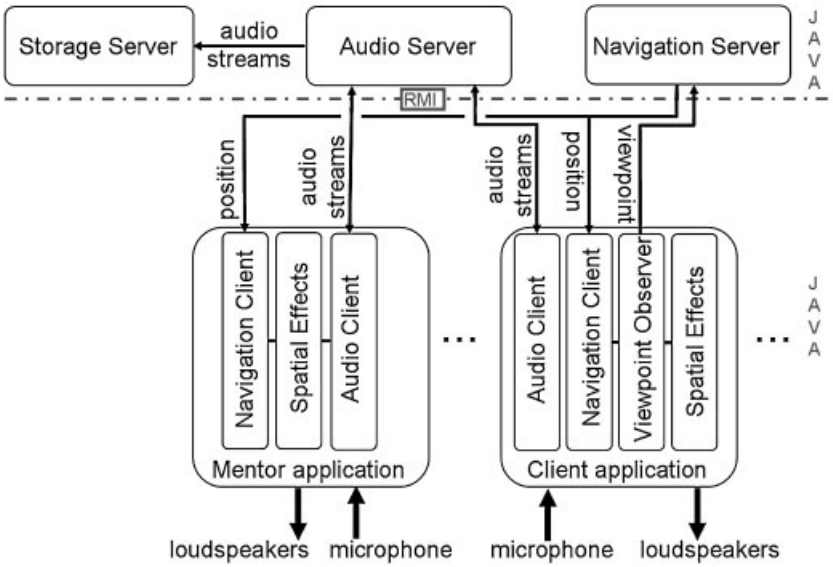

Figure 4 Real-time audio communication system in D-VIDERO: program modules are coded in the same way as in Figure 2. Narrow arrows with the superimposed labels illustrate the courses of RMI data flows: position—tracked position and orientation of the leading trainee in the real world, viewpoint—leading trainee's current position and orientation in the virtual world, audio streams—streaming audio data.

of sound sources and listeners. In order to make real-time operation of the D-VIDERO system possible, only two simple effects are applied to raw samples: linear interpolation of the source's amplitude and adjustment of the interaural signal difference [22] between two stereo channels. This way, a user can determine the speaker's position merely by listening to his voice. The trainees' positions are supplied by the Navigation Server through the Navigation Client modules. The mentors' positions are currently fixed in the middle of the room, near the virtual baby. Finally, the augmented audio track is played through loudspeakers or headphones.

Recording and Reconstruction of the Training Session

Storage Server, depicted in Figures 2−4, can be activated by any mentor (Fig. 1). It communicates with Control, Navigation and Audio Servers and records all their data during a training session. This includes events for changing the newborn's model parameters, trainees' actions and their parameters, navigation (viewpoint changes) and also audio communication among the training participants. The whole session record can then be reviewed using the D-VIDERO Replayer program. The data received from Control Server is saved in plain text file with human-legible format which can, therefore, also be manually revised. The recordings of the participants' conversation are encoded into the MP3 format. Data flow among Storage Server and all other servers is schematically depicted in Figures 2−4.

PERFORMANCE EVALUATION

This section outlines the results obtained in several tests measuring the distributed VIDERO system real-time operation and synchronization between the mentor's and the trainee's applications. All the experiments were conducted on two Pentium IV Windows XP systems, with 2 GHz CPU, 512 MB RAM, Radeon 9000 graphics card and 100 Mbps Ethernet connection. For illustrative purposes only, the most common training configuration with one trainee and one mentor was tested. Cosmo Player VRML browser installed as a plug-in of Netscape Navigator was used to render the VE graphics, while Sun's JVM was used as Java interpreter. All performance indices were monitored for 20 min and sampled with the frequency of 10 Hz, while 10 test runs were recorded for each performance index. Special interest was paid to possible delays in latency times caused by the RMI services, as this could crucially influence the quality of the training in the distributed VIDERO environment.

Control Server

First, we investigated the virtual newborn's health state synchronization delay between the mentor's and

Table 1 Interclient Event Dispatch Time on Control Server and Total Event Dispatch Time (Mean ± Standard Deviation)

Interclient event dispatch time (ms)	4.3 ± 1.0
Total event dispatch time (ms)	12.7 ± 0.9

the trainee's application. We measured the time required by Control Server to dispatch the events to all the clients (interclient event dispatch time), and the total time to dispatch a scenario event, i.e. the time elapsed from the time instant when Control Server initiates the scenario event to the time instant the last Control Client receives it (total event dispatch time). As evident form Table 1, the differences in event dispatch time are in the range of few milliseconds and have a negligible impact on synchronization of the newborn's condition. Table 2 shows Control Server's CPU load is also negligible. This was expected: control events are sent with relatively low frequency, hardly exceeding the frequency of 1 Hz.

Navigation Server

The performance criterion with Navigation Server was the same as with Control Server. First, we measured the time (t_1) required by Navigation Server to read the Fastrak motion-tracking device data from the RS-232 port and to convert the direction cosine angles to the VRML orientation quaternion. Afterwards, the delay (t_2) caused by the transfer of data from the Navigation Server to the Navigation Clients, and the time (t_3) required to update the VRML viewpoint in VE were determined. The results are gathered in Table 3.

Following the requests from Navigation Client, Navigation Server forwards the position data to all clients. All the mentor's and the trainee's applications run independently, hence, the position synchronization depends mainly on the frequency of the position update calls triggered by the corresponding Navigation Clients. Table 4 shows the server's and client's CPU load, network throughput and the actual client's position refresh rate in dependence on the Navigation Client position update rate. The primary concern for

Table 2 The CPU Load (Mean ± Standard Deviation) Dependent of the Number of Scenario Events per Second

Frequency of dispatched events (Hz)	0.2	1	5
CPU load (%)	0.1 ± 0.2	0.2 ± 0.4	0.8 ± 0.7

Table 3 Times Required to Transfer the Position Information from the Real World into VE (Mean ± Standard Deviation)

t_1—Time to process the data from the Fastrak motion-tracking device (ms)	0.2 ± 0.1
t_2—Position dispatch time (ms)	13.2 ± 1.0
t_3—Time to update the VRML viewpoint (ms)	2.5 ± 0.7

The declared latency of Fastrak tracking device is 1 ms.

the real-time operation is the amount of processing which must be performed by the Navigation Client (mainly caused by the VE rendering). To illustrate a strong dependence between the CPU load and the avatar's position update rate, the data from Table 4 is partially depicted in Figure 5. Obviously, a compromise between the depicted factors must be made. As the position update frequency can easily be changed by changing the values in the D-VIDERO's initialization files, the user is given a chance to decide the exact performance versus CPU load ratio.

Audio and Storage Server

To evaluate the performance of Audio and Storage Servers, we consider only the CPU load and network throughput. The synchronization delay between two Audio Clients depends mainly on the network characteristics. According to our tests, it hardly exceeds 50 ms. More JMF performance indices can be found in Reference [21]. The Audio Client CPU load depends on the stream encoding format. The results for three commonly used formats are given in Table 5. The network throughput is kept low by applying spatial sound effects to audio streams at the client side. Every participant sends a single, non-spatialized stream to Audio Server who then multicasts it to all users, reducing the traffic even further. However, some extra network throughput is required (Table 5) to update the trainee's position information.

The amount of data received by Storage Server incorporates, in general, the data sent by Control and Navigation Servers and by both Audio Clients (Audio Server merely clones the audio streams from clients). Therefore, the obtained network throughput depends highly on the position update rate (Table 4) and audio encoding format (Table 5). Table 6 lists the Storage Server CPU load and IO activity in the case of the MPEG audio format and the navigation position update rate of 23 Hz. Note the low CPU load is due to the fact that the audio stream is already encoded into the RTP/MPEG format. Hence, only decoding from the RTP format is performed.

Table 4 The CPU Load and Network Throughput (Mean ± Standard Deviation) Dependent of the Navigation Client Position Update Rate

Navigation Client position update rate (Hz)	Display refresh rate (Hz)	Network throughput (kB/s)	Server's CPU load (%)	Client's CPU load (%)
20	14.5 ± 0.2	5.2 ± 0.1	2.7 ± 1.3	26.9 ± 2.8
40	22.9 ± 0.2	8.1 ± 0.1	3.6 ± 1.6	40.1 ± 4.1
60	26.7 ± 0.3	9.5 ± 0.1	4.4 ± 1.9	47.3 ± 3.9
80	30.9 ± 0.5	11.0 ± 0.1	5.2 ± 1.9	51.0 ± 4.4
100	37.9 ± 0.8	13.5 ± 0.1	5.2 ± 2.0	56.7 ± 3.8
150	52.8 ± 2.1	18.6 ± 0.3	5.6 ± 2.1	77.2 ± 3.4

DISCUSSION AND CONCLUSIONS

Although using the RMI services, the D-VIDERO prototype proved to be a very robust and accurate simulation system. As demonstrated in this paper, it is fully capable of following the predefined scenarios exactly, and to credibly represent the newborn's vital signs over the entire shared VE. There were two major implementation concerns: (a) valid synchronization of events in shared VE, and (b) low and controllable CPU load. The former enables the exact supervision of trainees' actions, whereas the latter guarantees the system's responsiveness and real-time operation. As demonstrated in the previous section, both issues are completely met when running VIDERO in high-speed network environments. The average delay between the events of the same type in different VEs is less than 0.1 s, while the servers add very little to the CPU load. Main CPU load is generated by the Navigation Clients due to continuous rendering of VE. In our tests, the position of the motion-tracking sensor was changed constantly, which resulted in frequent changes of the VRML viewpoint position. It is not likely that a trainee will always move with the same level of intensity, therefore, the results given in Table 4 should be read as the upper limits.

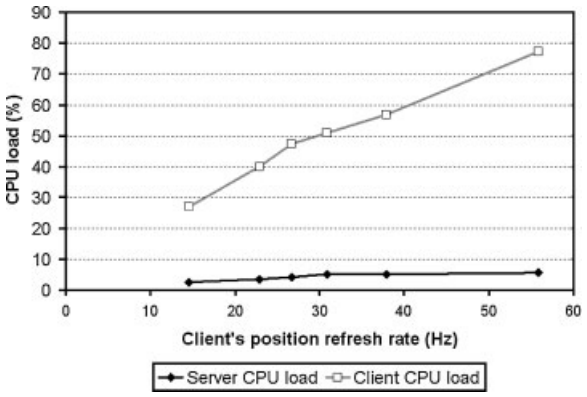

Figure 5 The Navigation Client's and Navigation Server's CPU load vs. the client's position refresh rate. While the server's CPU load maintains relatively constant, the client's CPU load increases linearly with the position refresh rate. The optimal performance vs. CPU load ratio can easily be set by modifying the refresh rate value in initialization files.

Altogether, the D-VIDERO prototype consists of at least five major components connected by a network: four RMI servers and at least one trainee application. An arbitrary number of mentor's and/or trainee's applications can be added, although the configuration with one trainee's and one mentor's application is most common. Regardless the number of mentor's and trainee's applications, Navigation Server should always run along with the leading trainee's application, whereas Control, Audio and Storage Servers can either run on the mentor's side, or on separate computers. The required network throughput in such arrangement depends, of course, on the audio stream format and the position update frequency, but varies only between 15.8 and 55.0 kB/s.

One of the main open issues, which have not been covered in this paper, remains the integration of the virtual objects' manipulation with the demand for scalable trainee's interventions. Beside restrictions induced by the state-of-the-art haptic interfaces, this issue is also limited by incomplete descriptions of the functionality of virtual objects. In order to enable a free manipulation with any medical instrumentation, all its functionality has to be predicted and predefined by the VR system's developers. No new instrumentation can be added by the training professionals, neither they can change the functionality of already defined objects. This clearly disagrees with the concept of an open and fully scalable training simulator. For the time being, a simple list of the virtual objects (i.e. ECG and respiration rate monitor, radiant heater, stop watch, stethoscope, laryngoscope, oxygen mask, etc.) that can be used by trainee in his

Table 5 CPU Load and Network Throughput (Mean ± Standard Deviation) in Dependence of the Audio Encoding Format

Audio encoding format	MPEG	MuLaw	GSM
Sampling frequency (Hz)	44,100	8,000	8,000
Bits per sample	16	8	8
Audio Server CPU load (%)	1.2 ± 0.3	1.2 ± 0.2	0.6 ± 1.0
Audio Client CPU load (%)	9.5 ± 1.8	2.2 ± 1.6	3.1 ± 1.8
Audio network throughput (kB/sec)	8.2 ± 0.4	6.7 ± 0.3	2.0 ± 0.1
Position network throughput (kB/sec)	1.6 ± 0.1	1.6 ± 0.1	1.6 ± 0.1
Position update rate (Hz)	4.5 ± 0.3	4.5 ± 0.3	4.5 ± 0.3

interactions with virtual newborn is hardcoded into the D-VIDERO prototype. However, the object manipulation is restricted only to the audio−visual feedback. Although force feedback is not of paramount importance in decision-making training, it may play a crucial role in the establishment of ''sense of presence'' in VE. Finally, to enable the scalable definition of the trainee's interventions by the training providers, the trainee still has to select the proper intervention from a rather simple and uncomfortable pup-up menu.

Future work will be devoted to the assessment of the existing D-VIDERO prototype and its feasibility to simulate an arbitrary postnatal complication. Additional effort is currently put into the development of user-friendly and easy-to-use graphical tools for creation of the text-based scenarios descriptions and trainee's interventions. Finally, the integration of scalable intervention definitions into VE and the principles of general object manipulation are being investigated.

ACKNOWLEDGMENTS

The authors are grateful to Professor Zvonko Fazarinc for his precious advice and vision of the future medical training systems, and to Professor Louis P. Halamek from Stanford University for his extensive medical and didactical support. This work was supported by the Slovenian Ministry of Education, Science and Sport (contracts nos. 3411-98-71-0001, S2-796-010/21301/2000, and Programme Funding P2-0041).

Table 6 Storage Server CPU Load, the Number of IO Write Operations and the Corresponding Data Flow (Mean ± Standard Deviation) in the Case of the MPEG Audio Format, with the Navigation Position Update Rate Set to 23 Hz

CPU load (%)	0.7 ± 0.6
IO Write frequency (operations/s)	2.0 ± 0.4
IO Write transfer rate (kB/s)	7.4 ± 1.6

REFERENCES

[1] L. P. Halamek, D. M. Kaegi, D. M. Gaba, Y. A. Sowb, B. C. Smith, B. E. Smith, and S. K. Howard, Time for a new paradigm in pediatric medical education: Teaching neonatal resuscitation in a simulated delivery room environment, Pediatrics 106 (2000), e25.

[2] A. Holobar, M. Divjak, D. Korošec, and D. Zazula, Training scenario prototyping for VR-based simulation of neonatal decision-making, Comput Appl Eng Educ 15 (2007), 317−328.

[3] D. Whitelock, D. Romano, A. Jelfs, and A. P. Brna, Perfect presence: What does this mean for the design of virtual learning environments? Educ Inf Technol 5 (2000), 277−289.

[4] M. Ponder, B. Herbelin, T. Molet, S. Schertenlieb, B. Ulicny, G. Papagiannakis, N. M. Thalmann, and D. Thalmann, Immersive VR decision training: Telling interactive stories featuring advanced virtual human simulation technologies, in Proceedings of Workshop on Virtual Environments, Zurich, Switzerland, Vol. I, (2003), 97−106.

[5] F. Mantovani, G. Castelnuovo, A. Gaggioli, and G. Riva, Virtual reality training for health-care professionals, Cyberpsychol Behav 6 (2003), 389−395.

[6] M. Lombard and T. Ditton, (1997), At the heart of all: The concept of presence, J Comput Med Commun 3 (electronic edition).

[7] M. Slater, Measuring presence: A response to the Vitmer and Singer presence questionnaire, Presence 8 (1999), 73−79.

[8] G. Riva, From technology to communication: Psychosocial issues in developing virtual environments, J Vis Lang Comput 10 (1999), 87−97.

[9] F. Mantovani, VR learning: Potential and challenges for the use of 3D environments in education and training. In Towards cyberpsychology, mind, cognitions and society in the internet age. G. Riva and C. Galimberti (Eds.), IOS Press, Amsterdam, 2001, 207−225.

[10] R. S. Kalawsky, S. T. Bee, and S. P. Nee, Human factors evaluation techniques to aid understanding of virtual interfaces, BT Technol J 17 (1999), 128–141.
[11] M. Green, Shared virtual environments: The implications for tool builders, Comput Graph 20 (1996), 185–189.
[12] D. England and P. Gray, Temporal aspects of interaction in shared virtual worlds, Interact Comput 11 (1998), 87–105.
[13] S. Stansfield, D. Shawver, A. Sobel, M. Prasad, and L. Tapia, Design and implementation of a virtual reality system and its application to training medical first responders, Presence-Teleoper Virtual Environ 9 (2000), 524–556.
[14] R. Carey and G. Bell, The Annotated VRML 2.0 Reference Manual. Addison–Wesley Developers Press, Berkeley, 1997.
[15] Humanoid Animation Working Group of the WEB3D Consortium, at URL http://h-anim.org/, 2003.
[16] N. Taffiner, C. Sutton, R. J. Fishwick, I. C. McManus, and A. Darzi, Validation of virtual reality to teach and assess psychomotor skills in laparoscopic surgery: Results from randomized controlled studies using MIST VR laparoscopic simulator, Stud Health Technol Inform 50 (2003), 124–130.
[17] Polhemus, Polhemus Fastrak motion tracking device, at URL http://www.polhemus.com/fastrak.htm, 2003.
[18] B. Peterson, M. Wells, T. Furness, and E. Hunt, The effects of the interface on navigation in virtual environments, in Proceedings of Human Factors and Ergonomics Society Annual Meeting Santa Monica, CA (1998), 1496–1505.
[19] Sun, Java Remote Method Invocation, at URL http://java.sun.com/products/jdk/rmi/, 2003.
[20] C. Bouras and T. Tsiatsos, Distributed virtual reality: Building a multi-user layer for the EVE Platform, J Netw Comput Appl 27 (2004), 91–111.
[21] Sun, Java Media Framework API Specification v2.0, at URL http://java.sun.com/products/java-media/jmf/index.html, 2002.
[22] J. Blauert, Spatial hearing: The psychophysics of human sound localization, MIT Press, London, 1999.

BIOGRAPHIES

Aleš Holobar received his BS and PhD degree in computer science from the University of Maribor, Slovenia, in 2000 and 2004, respectively. From 2000 to 2006, he was a researcher with the Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia. He is currently a Marie Curie fellow at Politecnico di Torino, Italy. His research interests include virtual reality, conceptual learning and signal processing, with current activities focused on blind source separation and biomedical signal processing.

Matjaž Divjak received MSc and PhD degrees in computer science from the University of Maribor, Slovenia, in 2003 and 2005, respectively. He worked as a teaching assistant with the Faculty of Electrical Engineering and Computer Science in Maribor from 2000 to 2005. Currently he is a postdoc researcher at INRIA in Nancy, France. His main research interests include computer vision, image processing and interactive VR environments.

Iztok Prelog graduated from the Faculty of Electrical Engineering and Computer Science, University of Maribor, where he was conducting research in the areas of conceptual learning and virtual reality. He is currently working as a software architect and lead software engineer at HERMES SoftLab, Slovenia. His current research focus is in the field of agent-based enterprise system management, especially management of distributed applications in ubiquitous computing environments.

Dean Korošec received his PhD from Ecole Centrale de Nantes, France, and the University of Maribor, Slovenia, in 1999. After spending 10 years on various research and applied projects at University of Maribor, he became project manager at Nova KBM in 2003. He is currently head of informatics at Nova KBM, second largest Slovenian bank, and director of M-Pay, joint company of Nova KBM and Mobitel. His main research interests include signal processing applications in medicine, simulated medical training and mobile payment systems.

Damjan Zazula received Dip Ing, master's and doctorate degrees in electrical engineering from the University of Ljubljana, Slovenia, in 1974, 1978 and 1990, respectively. After being involved in industrial R&D for 12 years, he joined the Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia, in 1987. Currently he holds a full professor position in computer science, while from 1998 to 2003 he was also appointed an associate dean of research. Dr. Zazula has spent several months as a visiting professor at the ETH in Zurich, Switzerland, and Ecole Centrale de Nantes, France. His main research interests are compound signal decomposition, biomedical imaging and virtual training tools. He is a member of IEEE Signal Processing Society, EURASIP, IAPR, Slovenian Technical Society, Slovenian Society of Pattern Recognition and Slovenian Society of Biomedical Engineering.