Distributed Consistency Maintenance Scheme for Interactive Mixed Reality Environments

Felix G. Hamza-Lup[1], Charles E. Hughes[1], Jannick P. Rolland[1,2]
(1) School of Computer Science
(2) College of Optics and Photonics: CREOL/FPCE
University of Central Florida

ABSTRACT

Advances in computer networks and rendering systems facilitate the creation of distributed collaborative environments in which the distribution of information at remote locations allows efficient communication. Particularly challenging are distributed interactive Mixed Reality (MR) environments that allow knowledge sharing through 3D information.

In a distributed interactive MR environment the dynamic shared state represents the changing information that multiple machines must maintain about the shared virtual components. One of the challenges in such environments is maintaining a consistent view of the dynamic shared state in the presence of inevitable network latency and jitter. A consistent view of the shared scene will significantly increase the sense of presence among participants and facilitate their interactivity.

In further consideration of the latency problems and in the light of the current trends in interactive distributed applications, we propose a hybrid system architecture for sensor based distributed environments that has the potential to improve the system real-time behavior and scalability.

Keywords: *Mixed Reality, Human Computer Interaction, Distributed Systems, Sensors, Distributed Architectures*

1. INTRODUCTION

One of the challenges in distributed interactive virtual environments is maintaining a consistent view of the shared state in the presence of inevitable inconsistency factors. It is only recently that researchers have begun to examine systematically the effects of consistency on the sense of presence. A consistent view in a shared scene may facilitate users interaction and thus significantly increase the sense of presence among participants [1]**.** One way in which interaction is related to presence is its ability to strengthen participant's attention and involvement [2].

The interactive and dynamic nature of a collaborative virtual environment is constrained by latency. Latency generators can be roughly grouped in two categories: computing system latency and network infrastructure latency.

In a Mixed Reality (MR) environment, in describing the equipment (e.g. head-mounted display) that provides stereoscopic visualization and body parts tracking, the latency is increased with the time elapsed from detecting the body part motion to the time the appropriate image is displayed. The computing system latency includes rendering delays (e.g. image-generation delay, video sync delay, frame delay, internal display delay), mismatches in data speed between the microprocessor and input/output devices, sensor delays (e.g. tracker delays), inadequate data buffers [3]. However, with the advances in hardware technology these delays are bound to become much smaller that the latency caused by the network infrastructure.

In this work we introduce and assess a new architecture for sensor based interactive distributed MR environments that falls in-between the atomistic peer-to-peer model and the traditional client-server model. The architecture facilitates the development of distributed MR collaborative applications in which data collected from real-time sensors is shared among the users. Each node in the system is autonomous and fully manages its resources and connectivity. The dynamic behavior of the nodes is dictated by the human participants who manipulate the sensors attached to those nodes.

The contribution of this paper consists of a shared state maintenance scheme involving a hybrid architecture for sensor based MR environments combined with a dynamic shared state maintenance algorithm. We built and deploy a prototype distributed MR environment using the proposed shared state maintenance scheme and assess the levels of consistency achieved while the number of participants varies.

2. RELATED WORK

A common approach for dynamic shared state maintenance is to keep the state for each participant in a file. A networked file system can be employed to provide distributed access to the centralized information. Examples of such systems can be found in [4],[5]. However such approaches suffer from scalability issues.

Several research efforts have been directed towards dead-reckoning algorithms to address the update rate problem by maintaining a loose consistency of the shared state. The idea behind dead reckoning is to transmit state updates less frequently and to use the information contained in the available updates to approximate the true shared state, in an effort to improve the scalability. Dead reckoning protocols for distributed Virtual Reality (VR) environments imply two phases: *prediction* and *convergence*. In the prediction phase the current state is computed at each participant based on the previous information. In the convergence phase the inaccuracies in the predicted state are corrected and a smooth transition is assured. To obtain better prediction results several research efforts have been concentrating on matching the virtual entity characteristics with the prediction algorithm to obtain an object-specialized prediction [6-10].

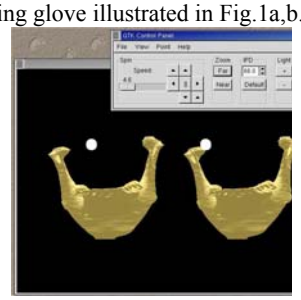
While traditional distributed virtual environments separate graphical and application state, several research projects like Studierstube [11], Repo-3D [12] and Avocado [13] try to simplify the development of distributed virtual environments by unifying the graphical and non-graphical state into a single data structure shared over the network. In DIV [14] for example, distribution is performed implicitly through a mechanism that

keeps multiple local replicas of a scene graph synchronized without exposing this process to the application programmer. The scene graph changes are propagated using reliable multicast.

With advances in computer graphics and tracking systems the research community has shifted attention to collaborative environments that span the entire virtuality continuum, e.g. Mixed Reality (MR) [15] and a subset of MR, Augmented Reality (AR). As a common feature, the applications emerging from these projects make extensive use of sensors (tracking, haptic etc) although they are based on local collaboration and are built on the traditional client-server architecture. None of the research efforts concurrent to these projects consider the possibility of collecting and distributing real-time sensory information from multiple remote sensors in the collaborative environment.

3. HUMAN COMPUTER INTERACTION - ACTION FREQUENCY PATTERNS

Let's consider the following application scenario. A surgeon located in his office is analyzing a 3D model of the mandible of one of his patients. He would like to discuss the surgical procedure that will follow shortly with one of his colleagues, whose office is in another building. As part of the discussion, they have to analyze the 3D model of the patient's mandible. They use the 3D distributed visualization platform implemented on the hospital's local area network. For stereoscopic visualization, each office is equipped with a head mounted display HMD [16] and a sensing glove [17]. In this scenario, the distributed visualization platform allows one participant to modify the position and orientation of the 3D model from a mouse-driven graphical user interface (GUI) or through the sensing glove illustrated in Fig.1a,b.



(a) GUI based interaction.

(b) Glove based interaction.

Fig.1. Interaction with the virtual world

There are two problems that arise from this scenario. The first is related to the network latency. As one of the participants manipulates the 3D model, the network latency desynchronizes their common viewpoints. Moreover, since network jitter is also present, the position/orientation drift among the views increases in time while the participants are not aware of the inconsistency of their viewpoints.

The second problem pertains to the nature of the interaction with the objects in the shared scene. In this scenario, the 3D model can be either manipulated from a GUI through discrete and predictable actions, or using the glove-like peripheral device which updates the environment at a higher rate and allows relatively unpredictable actions. The participant acting on the GUI through the mouse, for example, cannot exceed a certain frequency of actions mainly because of his motor reaction time. The action frequency is limited by the fact that the human-computer response time included perceptual (i.e. user perceives the items on the display or auditory signals), cognitive (i.e. user retrieves information from his own memory), and motor cycle times which can add up to an average of 240ms [18]. At the same time, since the position and the orientation of the object are set through the interface, predictable actions are applied on the object (e.g. by pressing the GUI's "Rotate around OX axis" button).

In contrast to the GUI, the glove-like peripheral device generates updates at high frequencies e.g. a P5 glove [19] has an optical tracking system attached that has a refresh rate of 60Hz and is going to capture the participant actions at a higher frequency. As a result, we have two types of interaction with the 3D model that have distinct patterns.

While the network latency problem is well known in distributed interactive environments, the second problem is more subtle and requires further analysis. Based on the above observations, we proposed a novel criterion for categorization of distributed interactive applications [20].

4. HYBRID NODES WITH REAL-TIME SENSORS

Distributed interactive MR environments involve the interaction of several remote participants while a significant amount of data is collected from sensors and devices attached to the participants. While some of the participants actively modify the shared scene, other participants are passive, in the sense that they do not interact with the shared scene. From this point of view we define two categories of participants: ***active participants*** and ***passive participants***.

An active participant triggers the virtual object state changes from a graphical interface or through a sensor attached to his/her node. Passive participants do not trigger any modifications of the shared scene. They just receive visual, haptic and/or audio feedback from the environment. The active/passive attributes of the participants can dynamically change during collaboration. An active participant can become passive and vice-versa depending on the collaboration needs.

A node in the distributed system is viewed as a computing device that allows a participant to interact with the virtual components of the environment. Without loss of generality, we will consider that each node has a set of ***sensors*** that provide position and orientation information and peripheral devices (e.g. mouse, keyboard). The discussion can easily be extended to other types of sensors (e.g. haptic) and other devices that can be part of the distributed system's resources. The system must ensure that the data captured by each node's sensors is distributed with minimum delay to all interested participants to maintain the environment consistency. Moreover, each node in the system will need to exchange its sensory data with all or a predefined subset of the system's nodes.

A pure centralized data distribution approach (e.g. client-server) would not be efficient because of the additional delay associated with the data collection stage, followed by the data distribution. An atomistic peer-to-peer approach would not fit either because of the additional overhead in data distribution. As a fundamental property, the nodes in a sensor based distributed system may act as data producers, consumers and distributors, simultaneously. Based on this observation we define four types or running modes for the distributed system's nodes: active nodes, passive

nodes, active forward nodes and passive forward nodes. A description of each type of node and the processes that run on them is available in [21]. In what follows we briefly describe the states of a node in a distributed interactive MR environment.

Let's denote the states as: {A, P, AF, PF}. "A" stands for "active", "P" for "passive", "AF" for "Active Forwarder" and "PF" for "Passive Forwarder". Let's denote the conditions that trigger the change in state using a binary representation {00, 01, 10, and 11}

- 00 - the state changes from **inactive** to **active**
- 01 - the state changes from **active** to **inactive**
- 10 - the state change from forwarding **off** to **on**
- 11 - the state change from forwarding **on** to **off**

The behavior of a node can be represented as a state machine as illustrated in Fig.2.

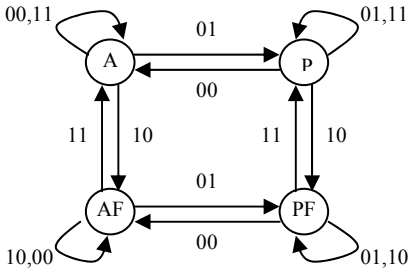


Fig. 2 State machine representing the hybrid node

Based on the above discussions, we now introduce a data distribution scheme targeted towards the development of sensor based distributed MR environments.

Hybrid Data Distribution Scheme

The ***dynamic membership*** property of the environment as well as the constraints of the interactive environment pointed us towards a hybrid between a client-server and peer-to-peer model. The proposed algorithm for data distribution is driven by the assumption that each active node manages a small network of potentially real-time sensors as illustrated in Fig. 3.

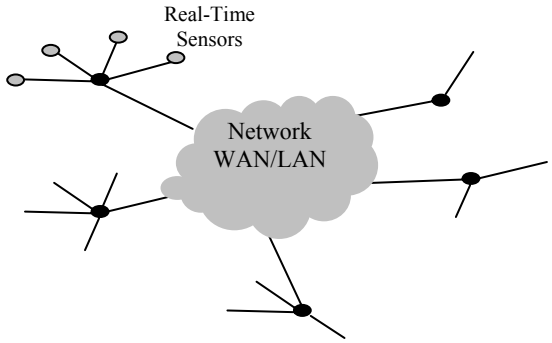


Fig. 3 Distributed Interactive Virtual Environments Nodes and Sensors

The algorithm builds an overlay network at the application level in which we employ the Core-Based Trees [22] techniques for multicast tree construction. Each node that becomes active (i.e. "A" node) will advertise the availability of information to all participants. Interested participants will join the multicast tree cored at the "A" node. At a particular moment in time the distributed interactive environment could contain several core-based trees rooted at the "A" nodes. Fig. 4 provides a snapshot in time of the distributed system.

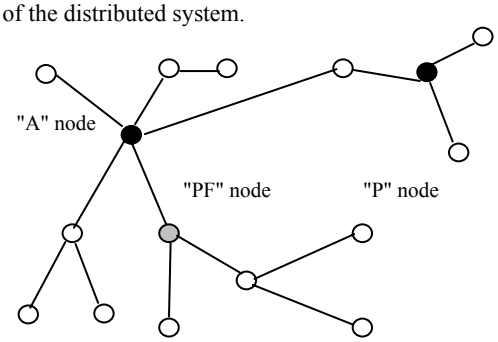


Fig.4 Snapshot of a Distributed Interactive MR Environment.

Consistency Maintenance Algorithm

In a MR environment the proportion of virtual and real changes continuously based on the participant's gaze and the interactions in the environment. As opposed to Virtual Reality, a MR environment might be less demanding on rendering and more demanding on sensor and other data (e.g. objects position and orientation) distribution. Assuming a fairly homogenous system, our main focus is the delay caused by the infrastructure where the application is deployed.

To compensate for these communication delays and for the variation in the delay (i.e. jitter) we have proposed a combination of distributed monitoring combined with a delay compensation method [23].

5. EXPERIMENTAL SETUP

To investigate the efficacy of the consistency maintenance scheme we have developed a prototype application using the Distributed Artificial Reality Environment DARE [14] and VESS [24] frameworks.

System Setup

The application was deployed in a distributed system containing six nodes interconnected on a 100 Mbps local area network. Additionally two optical tracking sensors were used to insert information into the environment at 60 Hz regarding the position and orientation of the virtual objects in the scene.

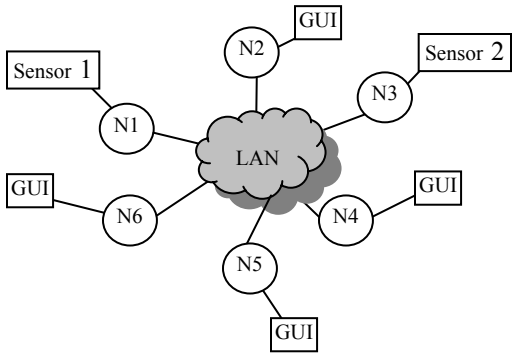


Fig.5 Six participants setup

The participants at nodes N1 and N3, respectively, interact on the shared scene through optical tracking sensors attached to the nodes as shown in Fig. 5. Participants at the other sites may interact through a GUI. From the hardware point of view each

site consists of one head-mounted display , a Linux based PC and an ARC display [24] as illustrated in Fig. 6.

(a) Optical See-through HMD (b) ARC Display

Fig. 6 Hardware components

Distributed Interactive AR/MR

At each location, the real environment is augmented with floating 3D objects seen through the HMD as illustrated in Fig.7.

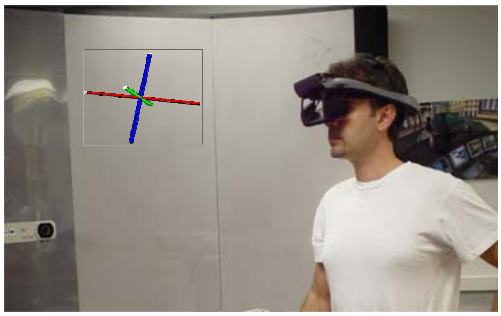


Fig.7 Interactive 3D Visualization participant and virtual 3D cross.

Participants can interact with the 3D models in two ways. Using a GUI they can point in the virtual space to different parts of the virtual objects and they can manipulate them. An alternative interaction method is using a tracking probe attached to an optical sensor (e.g. NDI Polaris Tracking system). By manipulating the probe the user can change the position and orientation of the virtual objects. The application captures the information form the tracking sensors at 60 Hz.

The distributed visualization application implemented on a hybrid node infrastructure is a simple example of a distributed interactive MR environment that utilizes sensors. We plan to experiment with other interactive distributed MR applications in the near future.

6. EXPERIMENTAL RESULTS AND ASSESSMENT

To assess the consistency maintenance scheme, the amount of orientation drift for a shared 3D object between the node that acts on the object and each of the other participating nodes is computed. We have focused our experiments on the assessment of the orientation drift. A similar assessment can be done for the object's position.

To emphasize the process we describe a simple scenario. We use two nodes A and P, sharing the same virtual 3D scene. A GUI is available at A, which allows the participant to change the 3D model *orientation* by applying rotations around the Cartesian axes. In this way, one participant generates events from the interface, and each time an event is generated, the object's orientation at both sites A and P is recorded. Because of the network latency, different vectors at each node will describe the orientation of the object. The rotations can be easily expressed using quaternion notation.

Let q_A express the rotation of an object at the A node and let q_P express the rotation of the same object at the P node. Both participants see the same virtual scene and the object should have exactly the same orientation. To quantify the difference between the orientations of the object as rendered on each node, we can compute the correction quaternion q_E every time the user triggers a new action. The correction can be expressed as

$$q_A = q_E q_P \quad (1)$$

and thus

$$q_E = q_A q_P^{-1} \quad (2)$$

where

$$\alpha = 2\cos^{-1}(\omega_E) \quad (3)$$

The angle α represents the drift between the orientations of the 3D model as seen by the two participants. To investigate the effects of the network latency, we performed experiments at different action velocities, given that the drift value for an object is the product between the action velocity applied on that object and the network latency. For example, let's assume a simple scenario consisting of two nodes representing an active and a passive participant. Suppose the average delay between these nodes is t_{ij}= 0.2ms and the active participant applies an action (e.g. a rotation around axis) on a 3D object in the shared scene with the angular velocity ω = 10(degrees/s). The angular drift in this case will be given by:

$$\alpha_{ij} = \omega t_{ij} = 0.002 \text{ (degrees)}$$

On a higher delay infrastructure in which the delay is 15ms, the same action would produce a drift:

$$\alpha'_{ij} = \omega t'_{ij} = 0.02 \text{ (degrees)}$$

While keeping the same delay (t_{ij}= 0.2 ms), the drift increase can be simulated by increasing the action velocity, i.e. ω = 100 degrees/sec. Therefore in order to simulate higher latency networks in our experiments we vary the action angular velocity from 1(degree/s) to 100(degrees/s). In this way we can simulate network latencies of up to 20(ms) on a 0.2(ms) average delay local area network.

Two participants

In the first set of experiments we've computed the orientation drift as the participant applies actions on the virtual objects without any compensation. The reason for these measurements is to obtain a reference for the drift magnitude and see its behavior as a participant interacts on the shared scene. The variation of the orientation drift value and its trend line (i.e. third order polynomial fit), while a set of fifty consecutive actions (i.e. random rotations) were applied on the virtual object, is illustrated in Fig.8.

Without any compensation the drift accumulates and the drift angle reaches over 210 degrees after 50 consecutive random rotations around the object axes when the rotations angular velocity is 100 degrees/second.

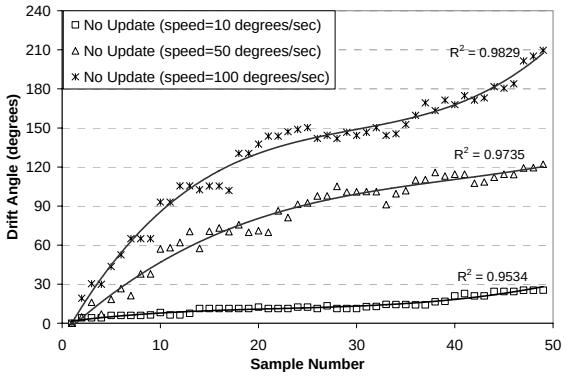


Fig. 8 Drift behavior with no drift compensation

In a second set of experiments we update the position of the virtual object after each change in action attributes. These changes are generated by the participant as he/she interacts on the virtual object. As illustrated in Fig. 9, the orientation drift is maintained at a fairly constant value. The trend lines use a high order polynomial fit. The drift angle is maintained at an average value of 11 degrees after 50 consecutive random rotations with angular velocity of 100 degrees/second.

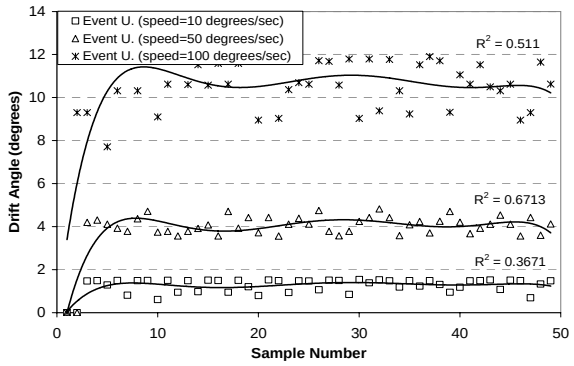


Fig.9 Drift behavior using the Event Updates method

In a third set of experiments, we have employed the adaptive scene synchronization algorithm to compensate for the communication delay and jitter. As shown in Fig. 10 the drift value is significantly decreased and kept at a constant level. A higher order polynomial fit was used for the trend lines. As in the case of event update the trend line has a sinusoidal shape which has a negative effect on the correlation coefficient (**R**). The sinusoidal shape of the trend line can be explained as an effect of the buffering and other system threads at the network and operating system level.

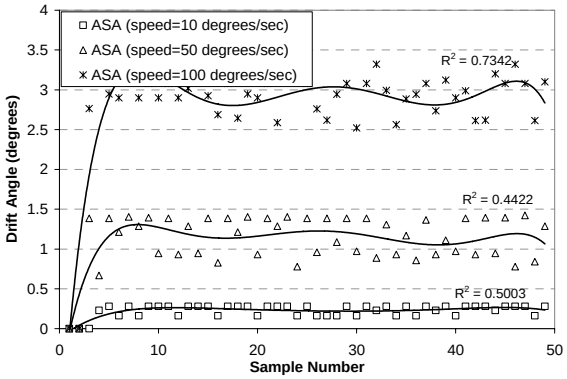


Fig.10 Drift behavior using the Adaptive Synchronization Algorithm

When the adaptive synchronization algorithm is used, the drift angle is maintained constant at a value that is two orders of magnitude lower that the average drifts without update and approximately four times smaller that the average drift for the event update (i.e. frequent state regeneration approach).

Scalability Analysis

To start investigating the scalability of the approach, we have increased the number of participants consecutively to three, four, five and six. To quantify the scalability of the adaptive synchronization algorithm regarding the number of participants we define a metric analyzing the relationship between the number of participants in the system and the drift values among their views.

Let ψ_i be the average drift value over all the participants, when $i+1$ participants are in the system. Without loss of generality, let us consider an action velocity of 100 degrees per second. In the case of a two participants setup, results show that the average drift is ψ_1 equal 2.83 degrees, while in the case of a six participants setup the average drift is ψ_5 equal 3.17 degrees. An algorithm with a low degree of scalability would have at least a linear increase in drift, i.e. ψ_n equal $n^* \psi_1$. On the other end, a high degree of scalability would mean $\psi_n \approx \psi_1$. Using this metric, in the six participant setup, a low degree of scalability would translate to ψ_5 equal $5^*\psi_1$ or 14.15 degrees. However, the experimental results and trend lines in Fig. 11 show that $\psi_5 \approx \psi_1$. Thus, the algorithm gives promising results in terms of scalability regarding the number of participants. The trend lines have been potted using linear regression. The slope of the regression line increases slightly with the action speed.

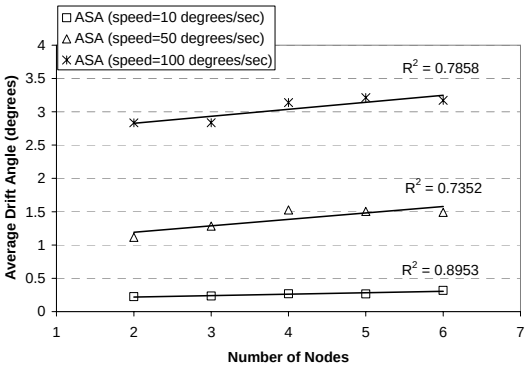


Fig.11 Orientation drift behavior as the number of participants increases

7. CONCLUSION

We have proposed a novel approach for managing distributed sensors as part of an interactive distributed MR application that combines a hybrid architecture with an adaptive delay compensation algorithm. The proposed approach allows dynamic behaviors for the distributed systems nodes allowing interactive data capturing and distribution. Furthermore we avoid a complex architecture as we believe simplicity is a key component in developing interactive real-time applications.

A preliminary assessment of scalability regarding the number of passive participants gives promising results. We are in the process of investigating the scalability of the system regarding active participants.

8. ACKNOWLEDGEMENTS

We wish to thank our sponsors: The Link Foundation, NSF/ITR IIS-00-820-16, Office of Naval Research Grant N000140310677, and the US Army Simulation, Training, and Instrumentation Command (STRICOM) for their invaluable support for this research.

9. REFERENCES

[1] Meehan, M., Razzaque, S., Whitton, M. C., et al. **Effect of Latency on Presence in Stressful Virtual Environments**. in *IEEE Virtual Reality*. 2003. Los Angeles, CA.

[2] Lombard, M. and Ditton, T., **At the Heart of it all: The concept of presence.** Journal of Computer-Mediated Communication, 1997. **3**(2).

[3] Swindells, C., Dill, J. C., and Booth, K. S. **System Lag Tests for Augmented and Virtual Environments**. in *13st ACM Symposium on User Interface Software and Technology*. 2000. San Diego, CA.

[4] Callaghan, B., Pawlowski, B., and Staubach, P., **NFS version 3 protocol specification. Request for Comments (RFC) 1813**. 1995, Information Sciences Institute: Marina del Rey, CA.

[5] Campbell, R. **Managing AFS: The Andrew File System**. 1998. Saddle River, NJ: Prentice-Hall.

[6] Katz, A. and Graham, K. **Dead reckoning for airplanes in coordinated flight.** in *Tenth Workshop on Standards for the Interoperability of Defense Simulations*. 1994. Orlando, FL: Institute for Simulation and Training.

[7] Pratt, D. R., **A software architecture for the construction and management of real-time virtual worlds.**, in *Computer Science*. 1993, Naval Postgraduate School: Monterey, CA.

[8] Friedman, M., Starner, T., and Pentland, A. **Device Synchronization using an optimal linear filter.** in *Symposium on Interactive 3D graphics*. 1992. Cambridge, MA: ACM SIGGRAPH.

[9] Schmalstieg, D. and Hesina, G. **Distributed Applications for Collaborative Augmented Reality**. in *IEEE Virtual Reality 2002*. 2002. Orlando, Florida.

[10] MacIntyre, B. and Feiner, S. **A Distributed 3D Graphics Library**. in *ACM SIGGRAPH*. 1998.

[11] Tramberend, H. **Avocado: A Distributed Virtual Reality Framework**. in *IEEE Virtual Reality*. 1999. Houston, TX.

[12] Hesina, G., Schmalstieg, D., Fuhrmann, A., et al. **Distributed Open Inventor: A Practical Approach to Distributed 3D Graphics**. in *ACM Virtual Reality Software and Technology*. 1999. London.

[13] Tamura, H. **Overview and final results of the MR project**. in *International Symposium on Mixed Reality*. 2001. Yokohama, Japan.

[14] Rolland, J. P., Biocca, F., Hua, H., et al., **Teleportal Augmented Reality System: Integrating virtual objects, remote collaborators, and physical reality for distributed networked manufacturing.**, in *Virtual and Augmented Reality Applications in Manufacturing*, S.K.O.a.A.Y.C. Nee, Editor. 2004, Springer-Verlag London Ltd. p. 400.

[15] Sturman, D. J. and Zeltzer, D., **A survey of glove-based input.** IEEE Computer Graphics and Applications, 1994. **14**(1): p. 30-39.

[16] Eberts, R. E. and Eberts, C. G., **Four Approaches to Human Computer Interaction**, in *Intelligent interfaces: theory, research, and design*, P.A. Hancock and M.H. Chignell, Editors. 1989, North-Holland. p. 69-127.

[17] Essential Reality, i., **P5 Manual**. 2002.

[18] Hamza-Lup, F. and Rolland, J. P., **Scene Synchronization for Real-Time Interaction in Distributed Mixed Reality and Virtual Reality Environments.** PRESENCE: Teleoperators and Virtual Environments, 2004. **13**(3).

[19] Hamza-Lup, F., Hughes, C., and Rolland, J. P. **Hybrid Nodes with Sensors - Architecture for Interactive Distributed Mixed and Virtual Reality Environments**. in *8th World Multiconference on Systemics, Cybernetics and Informatics*. 2004. Orlando, FL.

[20] Ballardie, A. J., Francis, P. F., and Crowcroft, J. **Core Based Trees**. in *ACM SIGCOMM*. 1993.

[21] Hamza-Lup, F. and Rolland, J. P. **Adaptive Scene Synchronization for Virtual and Mixed Reality Environments**. in *IEEE VR 2004*. 2004. Chicago, MI.

[22] Hamza-Lup, F. G., Davis, L., Hughes, C. E., et al., **Where Digital Meets Physical - Computer-Based Distributed Collaborative Environments**, in *CrossRoads*, ACM, Editor. 2003.

[23] Daly, J., Kline, B., and Martin, G. **VESS: Coordinating Graphics, Audio, and User Interaction in Virtual Reality Applications**. in *IEEE Virtual Reality*. 2002. Orlando, FL.

[24] Davis, L., Rolland, J. P., Hamza-Lup, F., et al., **Enabling a Continuum of Virtual Environment Experiences.** IEEE Computer Graphics & Applications, 2003. **23**(2): p. 10-12.