Real-time Interaction in VR with a Distributed Multi-Agent System

Hans J.W. SPOELDER[1] Luc RENAMBOT[2] Desmond GERMANS[1] Henri E. BAL[1,2]
Frans C.A. GROEN[3]
[1]Division of Physics and Astronomy
[2]Division of Mathematics and Computer Science
Faculty of Sciences, Vrije Universiteit, Amsterdam
[3]Informatics Institute, University of Amsterdam, Amsterdam

Keywords: Interactive Virtual Reality, Distributed interactive simulation, Collaborative visualization

Abstract

We describe a Virtual Reality system that allows users at different locations to interact with a distributed multi-agent system. We use RoboCup (robot soccer) as a case study. A human player who is immersed in a CAVE can interact with the RoboCup simulation in its natural domain, by playing along with a virtual soccer game. The system supports distributed collaboration by allowing humans at different geographic locations to participate and interact in real time. The most difficult problem we address is how to deal with the latency that is induced by the multi-agent simulation and by the wide-area network between different CAVEs. Our navigation software anticipates the movements of the human player and optimizes the interaction (navigation, kicking). Also, it sends a minimal amount of state information over the wide-area network.

1. Introduction

Multi-agent systems are becoming increasingly important in our society. The majority of such systems is in some way related to Internet applications, predominantly in the field of electronic commerce. Real-world multi-agents (robots) are still in their infancy, but also of growing importance. Their application area is broad, covering among others cleaning, public safety, pollution detection, fire fighting, traffic control, and games. Both real-world and cyber-world based multi-agents should be able to cooperate, develop optimal sensing and action strategies, and show an adaptiveness towards their task. Multi-agent systems are by nature distributed and rely often on a broad range of different sensors. The interaction of a human with a multi-agent system in real-time poses several intriguing problems, like the representation of the world as seen by the agents, the type of interaction, and the possibility of sharing the virtual world with other (possibly remote) users. In this paper, we study the use of Virtual Reality (VR) techniques for real-time interaction between humans and a distributed multi-agent system. In the virtual reality of the sensor data, the human can interact with the agents to study and control the multi-agent system.

To facilitate progress in this research area, it is useful to define a standard problem. In the Artificial Intelligence community, Robot Soccer –better known as RoboCup– has been chosen as a standard problem in which a wide range of technologies are integrated and examined [8]. The goal of RoboCup is to let teams of cooperating autonomous agents play a soccer match, using either real agents (robots) or simulated players. We will show that RoboCup also is a useful and challenging application for studying real-time interaction with a distributed simulation. Our objective is to construct a distributed VR environment in which humans at different geographic locations can play along in real-time with a running RoboCup simulation in a natural way, almost as if they were participating in a real soccer match [24].

Our work on RoboCup differs from earlier case studies like virtual tennis [14] in that humans interact with a running distributed simulation program and not just with other humans. A human can take over the role of any of the simulated soccer players, which is useful for testing strategies that have not been coded in the simulation program yet. Interacting with a simulation program, however, is even harder to implement than interaction between humans. A simulation program will unavoidably introduce delays, which make a realistic real-time visualization and interaction a challenging problem, as we will discuss in depth in the paper.

The paper is organized as follows. In Section 2, we discuss related work. Section 3 introduces some basic concepts of RoboCup. In Section 4, we present an overview of our virtual RoboCup environment. Some difficult implementation issues on real-time navigation, interaction, and remote collaboration are discussed in depth in Section 5. Finally, we draw conclusions and discuss future work in Section 6.

The main contributions of the paper are as follows:

1. We propose RoboCup as a useful case study for VR research on real-time interactions between humans and multi-agent systems (or distributed simulation programs in general).

2. We describe a prototype distributed implementation that allows humans at different geographic locations to participate in the same virtual soccer match. This prototype has been demonstrated at the RoboCup'99 tournament in Stockholm, by playing a distributed virtual match using CAVEs [2] in Amsterdam and Stockholm.

3. We discuss one difficult and important problem in detail: how to achieve a natural interaction in the presence of latencies. The latency problem is caused by the delay that the simulation introduces in generating the virtual reality. Also, when multiple CAVEs are used, the wide-area interconnection network increases the latency even further. The latency problem is general and also appears in other VR applications in which humans interact with simulations [22]. The paper describes several experiments that give more insight into this problem.

2. Related work

Interactive and collaborative visualization radically change the way scientists use computer systems [4]. With interactive visualization, a user can interact with a program in its visual domain. Distributed collaboration allows multiple users at different geographic locations to cooperate, by interacting in real time through a shared application [11, 22].

Many existing applications restrict the interaction to the visualization process (e.g., the direction of view, the zoom factor). A more advanced form of interaction (i.e., *steering*) allows the user to interact with the simulation process. Several systems exist that support steering [15, 17], but they typically provide low-level interactions and require users to monitor or change the application program's internal variables. In [21], the authors describe how robots can be steered from a Virtual Reality environment. Our RoboCup application allows the user to interact with the simulation program in a high-level, natural way. We think that this capability is beneficial to many scientific applications. Examples of such high-level interaction are reported in the literature for simulation of molecular docking and molecular dynamics [10, 12]. Although RoboCup may seem very different from scientific applications, it in fact can be used to study many issues: distributed multi-agent systems, teleimmersion, remote collaboration and man-multi-agent interaction.

Our application is collaborative in that it allows real-time interaction between humans at different geographic locations, which is a challenge for the implementation. The system-induced latency and the network latency between the different sites become a difficult problem. Some applications address the network latency problem by using dedicated ATM links [6, 9]. In [16], the authors compare the performance of the users in achieving different tasks under several network conditions (varying latency and jitter) in a collaborative virtual environment. They show that high latency reduces performance; the variability of the latency makes the use of prediction difficult and introduces a lack of hand-eye coordination. An interesting performance metric is the complete time (lag) in system response, including the simulation, tracking, rendering, network, and synchronization. Lag over 200msec makes interactivity difficult [25]. We therefore try to minimize the amount of communication between the user and the simulation program. In particular, we couple remote users at the level of the simulation program and not at the visual level, allowing us to transfer the (small) simulation state instead of (large) images. Similar to our problem is the problem of remote modeling of objects in a large distributed system (large-scale military simulations or networked games). However, in RoboSoccer, there is only one centralized world state (at the Soccer Server), so techniques like *Dead Reckoning* or *Position-History Based Dead Reckoning* [23] cannot be applied.

Our research also is related to other work on virtual games. Molet et al. implemented a collaborative environment that is used for a virtual tennis game between humans at different locations [14]. We do not study interaction between humans alone. Rather, we study interaction between humans and simulation programs. Also, we put much less emphasis on realistic modeling of human bodies. The RoboCup application raises interesting problems for the human-computer interaction, for example which input and output devices to use and how to manipulate objects in virtual environments. Another issue is the movement of the user in a limited area, as a soccer field is much larger than the CAVE. We do not investigate this problem in our research and simply use the wand (i.e., a 3D mouse) to navigate, although this is not a natural way to navigate. One approach is to scale the user movements to the size of the field when he goes out of a delimited zone [14]. Other approaches have been proposed such as neural networks to detect walking movement [26] or hardware devices to allow locomotion [5].

3. RoboCup

RoboCup (or The Robot World Cup) attempts to promote intelligent robotics research by providing a common task for evaluating various theories, algorithms, and agent architectures. RoboCup comprises two sub-fields: real robots and simulations. This paper deals with the simulation part of RoboCup.

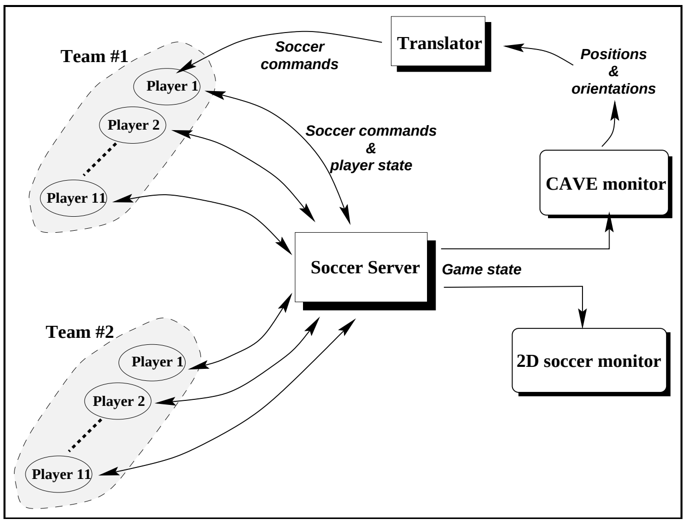

Figure 1. Software architecture of the RoboCup application

The software architecture of the RoboCup application is shown in Figure 1. The *Translator* and *CAVE monitor* are developed by us and are described later in the paper; the remaining components are part of existing RoboCup software and are described in [8].

Mode	Camera	Interaction	View
Monitor	event driven (coupled to ball)	–	2D
Workbench	void	point and select	3D non-immersive
CAVE	human	play along	3D immersive

Table 1. Overview of different visualization modes and their characteristics

A central role in the simulation is played by the so called *Soccer Server*, which keeps track of the state of the game and provides the players with information on the game. The server can be seen as a (virtual) world on which ideal sensors of the players can operate. The players are individual processes that can request state information from the server and autonomously calculate a change (or more globally, a behavior). The server also enforces the rules of RoboCup and ignores invalid commands from the players. For this reason, participants in a RoboCup tournament are not allowed to make any changes to the server. Our RoboCup VR system therefore also uses the unmodified server software. The players communicate with the server and send it soccer commands. The commands are the intentional behavior of the player, and are expressed in a simple language, consisting of dashes (accelerations), turns, and kicks:

- dash [dash-amount]
- turn [turn-angle]
- kick [kick-force, kick-direction]

The server discretizes time into slots of 100msec. To prevent server overloading by unfair play, only the last command of a player to the server within a given time slot is executed; other commands are ignored by the server. Also, the kick command requires the player to be within a certain distance from the ball (1 meter by default), else the command has no effect.

With N players per team, the state of the server is defined by $2N + 1$ M-dimensional coordinates (one for each player and one for the ball). The current implementation of RoboCup is two-dimensional, so the dimension M is two. A two dimensional visualizer (called the 2D soccer monitor) is provided in the standard RoboCup software distribution[1]. The bandwidth required to transmit the state of the game once every 100msec is $10 * (2N + 1) * M * S$ bytes per second, where S is the storage format of the positions. This bandwidth is small enough for current Internet connections (or even telephone lines).

Thus we can summarize the RoboCup application as a multi-agent system (the agents being the players), centered around a virtual world (the Soccer Server), with a discrete time and discrete interaction space.

4. Overview of the Virtual RoboCup System

As stated in the introduction, our objective is to construct an environment in which humans can play along with a running RoboCup simulation in a way as natural as possible. Below, we will describe the functionality of our system. In Section 5 we will discuss the implementation.

Our virtual RoboCup system not only uses immersive environments (like the CAVE), but integrates various visualization devices, as shown in Figure 2. We identify three different modes in which a human can observe the RoboCup multi-agent system: the monitor mode, the workbench mode, and the CAVE mode (see also Table 1). For each mode, we approach the human notion of a soccer match as closely as possible.

The first mode is the monitor mode, which allows spectators at different locations to passively watch the match on their graphical workstations. We have developed an automated camera system that tracks the ball and switches

[1]The original RoboCup simulation software can be found at *http://www.robocup.org*

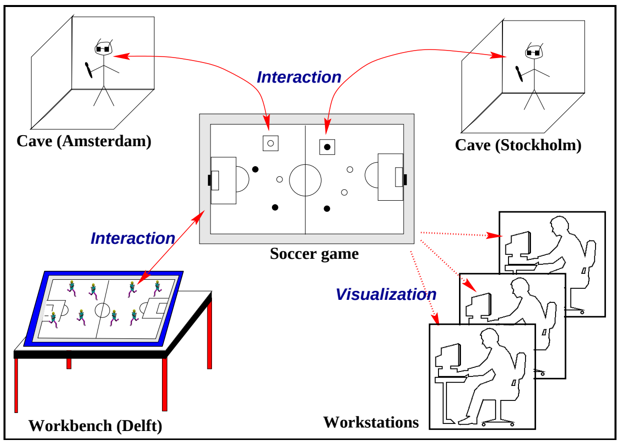

Figure 2. Interactive and collaborative visualization of a soccer match

between camera positions according to a user-definable algorithm. This mode thus is an approximation of the standard TV coverage [3, 7]. The second mode is implemented on a workbench[2] and provides a 3D overview of the soccer match. Its main feature is that visualization is miniaturized to allow a human to overview the state (without a predefined viewing angle). The goal here is to allow a coach to steer the game and give instructions to the players. We have implemented the 2D camera system and we have a partial implementation of the workbench software. In the remainder of the paper, however, we focus on the third mode, the CAVE mode.

The CAVE mode allows the user to be immersed in the game and to interact with it. We have implemented a new RoboCup monitor, the *CAVE monitor*, which uses the same information and communication as the original 2D monitor (see Figure 1), but now visualizes the state of the game in a CAVE. As a starting point in the visualization, we have built a virtual stadium and a parameterized soccer player. Animation of the players is based on the 'walker' motion data included in the GLUT (OpenGL Utility Toolkit) distribution. The state of this walker is characterized by five degrees of freedom. Its movement is governed by a set of basic points that are interpolated by splines. The movement as a whole is periodical. We discriminate three different modes of movement: standing still, walking, and running. The visualization of the movement is adjusted by linearly interpolating between the three modes.

The visualization system has to compute several quantities from successive states of the game. The reason is that the soccer server tries to reduce the required communication bandwidth by sending a minimal amount of information to the visualization system. For example, the direction of movement and velocity of the players and events like kicking the ball have to be determined from two successive states of the game. Likewise, determining the acceleration requires three successive states.

In addition to this visualization software, we have developed software to track the behavior of a human in the CAVE. We use three trackers to handle the interaction of the human with the VR software. One tracker is connected to the viewing glasses and is used to monitor positional changes of the human player inside the CAVE. The second tracker is connected to the wand, which can be used for global movements over the soccer field. The third tracker is attached to the foot of the human player and is used to recognize a kick.

The most difficult problem in realizing a virtual RoboCup system is caused by the latency of the simulation program (the Soccer Server). If the human player moves over the virtual soccer field, these moves happen almost instantaneously for the human. In contrast, the Soccer Server will require some time to process the change of position. As a consequence, the position of the human in the CAVE may be different from the position stored in

[2]A workbench uses one projector to generate a 3D image on a slightly tilted projection plate. The users see the 3D scene as if it comes out of a table.

the server. Such a difference will especially affect the kick command, since the RoboCup rules require the player to be within a certain distance from the ball to be able to kick it. The human player may thus think he is close enough to the virtual ball, but the server (the simulation) may have different information and ignore the kick. This problem is a typical example of how a delay introduced by a simulation program can harm a natural and real-time interaction [16].

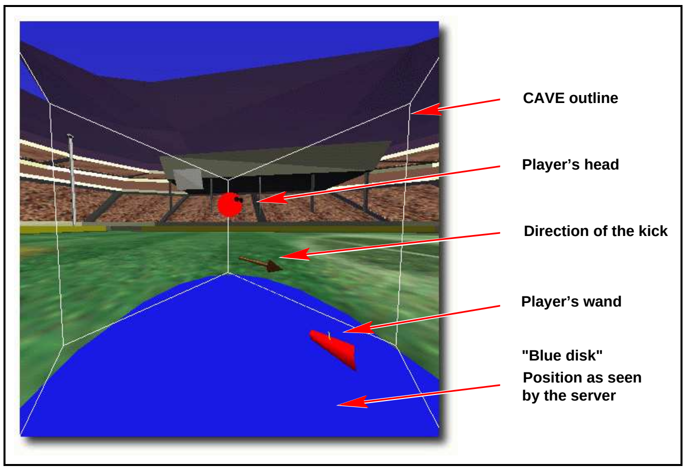

Figure 3. Interaction radius and human position according to Soccer Server

To study this problem, we introduce a visual cue for the human, called a *blue disk*. The center of the disk corresponds to the human position as currently stored by the server, and the radius corresponds with the interaction area (i.e., the area in which the human is allowed to kick). This is illustrated in Figure 3. Usually, the blue disk will stay close to the human player. If not, the lack of tracking makes the user aware that, according to the server, he is making an illegal move. In practical experiments, we have determined that the human user can easily recognize the disk and does not experience its presence as distracting. More implementation details on this problem will be given in Section 5.

Our virtual RoboCup environment also allows visualization front ends at different geographic locations to be coupled to the same simulation. In this respect, the environment is an example of a distributed collaborative application, which is becoming an important class of scientific applications [4]. We have explored this possibility in a virtual soccer match played on August 3, 1999 using the CAVEs in Amsterdam and Stockholm. For this match, human players in both cities joined the team of simulated players. The large distance (1300 kilometers) between the two CAVEs introduces a similar latency problem as described above: the actual position of the human in the remote CAVE may be different from the position currently stored at the Soccer Server. This holds especially for the human player in Stockholm, because the server and the simulated players were run in Amsterdam. We will describe and study this latency effect in Section 5. In the visualization, the remote human position is shown by a pyramid and the corresponding player is shown by an ordinary player. Arrows denote the viewing direction of the human in the remote CAVE.

5. Implementation and Performance Issues

Our RoboCup system uses the standard Soccer Server that is part of the existing RoboCup software. We use player processes that were developed at the University of Amsterdam. We extended this software in several ways. First, we have implemented a CAVE monitor, which visualizes the soccer game in a CAVE, as discussed in the previous section. In addition, we adapted the virtual player process to allow a human to take over the identity of one of the players, participate in the game, and interact with the simulation. For this purpose, we use a *Translator* process (see Figure 1) that translates tracker changes into soccer commands and transmits these to the Soccer Server (just like the normal virtual players do). Finally, we coupled two CAVEs at different locations. The two CAVEs are connected to the same Soccer Server, so the two humans participate in the same game. The two CAVEs exchange tracking data to achieve this coupling, using the network functionalities of the CAVE library.

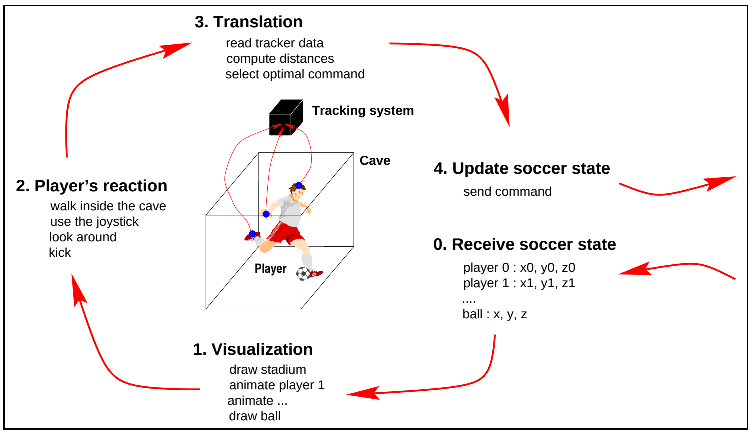

Figure 4. Temporal difference

The major problem in implementing the system is its inherent delay, as illustrated in Figure 4. At each cycle, the CAVE monitor receives state information from the Soccer Server. This data is expanded from 2D to 3D and visualized in the CAVE. The user then reacts to the world he is immersed in. These movements are converted by the translator process into commands, which are sent back to the Soccer Server. All these steps are processed in a pipelined manner, so the user reacts on the previous state of the Soccer Server, and his movements are processed during the next simulation step. This inherent lag of the system significantly influences the navigation, the interaction, and the collaboration, as we describe below.

To study and optimize our system, we have done several experiments, using the CAVE located at SARA Amsterdam [20]. This CAVE is connected to an IBM SP/2, which runs the RoboCup simulation (using multiple processors for the players and other processes). For the wide-area experiments, we use a second CAVE, located at Stockholm [18], which also is connected to a local SP/2.

Our goal is to let the human player in the CAVE be able to kick the ball when he wants. This is only possible if, according to the Soccer Server, the human is close enough to the ball. Thus, we have to reduce the distance between the position of the user in the CAVE and the corresponding position in the simulated world as much as possible, to offer natural interaction. One obvious solution could have been to sample a joystick (mounted on the wand) and to emit the directly corresponding commands (forward motion means *dash*, sideways motion means *turn*). The visualization is then updated for the next cycle with the monitor data. When using such an approach, however, the user always reacts on a past view. Also, his movements are limited, so he experiences non-natural interaction. Moreover, the refresh rate of the monitor data from the Soccer Server is only 10Hz, which is too slow to provide smooth visualization. Below, we explain our solution to this problem in more detail.

We have tested our system with two 'benchmark' trajectories. One is the path of a simulated player of the 'CMUnited98'-team (*CMU-track*), taken from the final of the 1998 RoboCup Simulator Tournament. This trajectory is originally built out of simulation-native turn and dash-commands. It is ideally suited to test the algorithm. The second benchmark is the path of a human in the CAVE that uses a joystick (mounted on the wand) to steer the player around the soccer field (*Joystick-track*). Because this trajectory is not native to the Soccer Server, we expect it to be harder to follow. We replayed these two trajectories, varied the dash-factor parameter of our algorithm (see below), and measured the average distance between the user and the corresponding simulated player.

5.1. Navigation

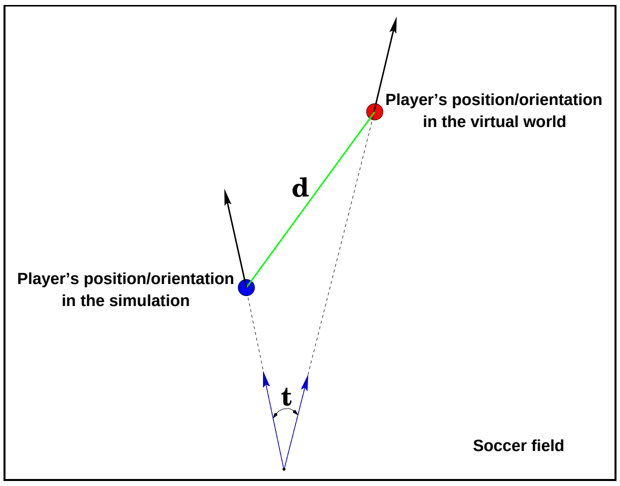
Player's position/orientation in the virtual world
d
Player's position/orientation in the simulation
t
Soccer field

Figure 5. Difference between human and represented player

As described above, the translator process receives changes in the position and orientation of the human player in the CAVE. It compares this information with the position and orientation stored in the simulation (see Figure 5). It then tries to emit a soccer command that reduces the difference between the two (i.e., which will bring the simulated position and orientation close to those of the human). Since the Soccer Server will accept only one command per 100 msec, the translator waits until the Soccer Server has acknowledged the previous command before sending a new command.

An interesting problem is which command should be emitted by the translator. Both the position and orientation may differ between the virtual world and the simulation. Ideally, the translator thus should emit a turn-command to cover the difference in orientation (the angle t) and at the same time a dash command to cover the distance d, as illustrated in Figure 5. Unfortunately, the Soccer Server allows only one command per cycle, so we need a way to serialize turn and dash-commands.

We tried several strategies and found that the most accurate strategy is simply to alternate the turn and dash-commands. Unlike other strategies, this gives two steady streams of commands (with a fixed time between each consecutive command) for both the angle and the distance. This property allows us to optimize the turn and dash-parameters by using simple extrapolations and by partially modeling the inverse of the Soccer Server. Taken this into account, the algorithm is left with one variable parameter, the dash-factor. The algorithm is constructed in such a way that this parameter should have a value of 100.0 momentum units to keep the distance minimal.

Figure 6 shows the results for both trajectories in local and wide-area setups. The shaded area indicates the *interaction range*, the distance at which the player is allowed to kick the ball. It has been set to one meter by the Soccer Server.

As expected, the wide-area results show a larger average distance. In this case, the algorithm needs to recover from network latency-effects and glitches common to the Internet. Furthermore, the algorithm clearly prefers motions that are native to the Soccer Server. The average distances for the CMU-track are smaller than those of the joystick-track.

Due to the large amount of noise in the system, even when using average distances, it is not clear which dash-factor is optimal for the tested trajectories. The results show roughly that a dash-factor below 80 increases the

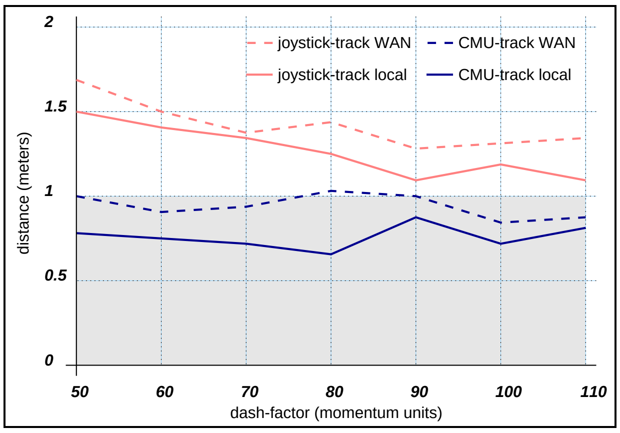

Figure 6. Dash-factor influence on average distance

average distance. For these low-strength cases, the algorithm can not keep up with the speed of the human in the CAVE.

5.2. Interaction

Full interaction with a RoboCup soccer game consists furthermore of being able to kick the virtual ball. For this, we added a third tracker to the CAVE setup at SARA. The VR software recognizes a kick-command using this tracker, attached to the preferred foot of the human. A kick-command is issued when three conditions are met. Firstly, the tangential speed of the foot must be above a set threshold. Furthermore, the human must be within kicking-range of the ball, and finally, the foot has to move towards the ball. The force-parameter of the kick-command is derived from the instantaneous speed of the foot, and the angle-parameter is derived from the instantaneous direction of the foot. Both speed and direction are sampled at the point where the speed reaches a local maximum. Using this approach it is possible to give both hard and soft (dribbling) kicks. Since the accuracy of the trackers we use (Ascension Flock-of-Birds magnetic trackers) is limited, especially when the tracker is at a highly asymmetric position as is the case for the foot tracker, the detection of the kick from this data is rather crude.

Figure 7 shows the distribution of distances measured for the CMU-track both locally and in a wide-area setup. It shows that 50% of the time (indicated by the line marked ’50%’ in Figure 7), the distance is below 1 meter and 90% of the time (indicated by the line marked ’90%’ in Figure 7), it is below 1.6 meters making natural interaction possible in most but not all cases. The tail (indicated as ’large latency errors’ in Figure 7) is larger for the wide-area case. This is expected, because the algorithm recovers more from latency problems in the wide-area setup.

Currently we are preparing experiments to test the ability of experienced and inexperienced users to play soccer in our virtual environment. It should consist of some local exercises (dribbling, following a path, moving around objects, etc.) and some global tasks (running across the field with the ball, kicking accurately over long distances, etc.). The experiments should give more insight on the effectivity of natural interaction in the RoboCup environment.

5.3. Collaboration

During the RoboCup’99 event, we played a soccer match between two human players in two CAVES (in Amsterdam and in Stockholm). The Soccer Server was located in Amsterdam, so the Amsterdam-player experienced

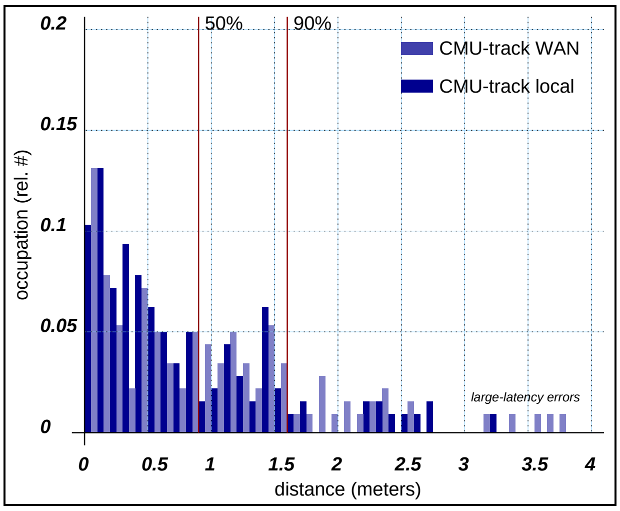

Figure 7. Distance-histogram for CMU-track

local latency effects, whereas the Stockholm-player experienced wide-area latency effects. Even though this difference was noticeable in the soccer experience on both sides, the game was a success.

6. Conclusion and Future work

We have described a Virtual Reality system that allows users at different locations to interact with a distributed multi-agent system. For our case study, RoboCup, the interaction is natural and is similar to that of a human soccer player: the user in a CAVE can kick a virtual ball.

The most difficult problem we addressed is how to deal with the latency that is induced by the multi-agent simulation and by the wide-area network that connects different CAVEs. This latency causes a difference between the real position of the human in the virtual space and the position stored by the simulation. Latency is unavoidable, so to some extent it is a problem one has to live with. We reduce the impact of the latency problem in several ways. Our navigation software anticipates the movements of the human player and tries to reduce the difference in the positions. Also, we provide visual feedback to the user, showing the current position according to the simulation. Finally, we exploit the principle of compression and send a minimal amount of state information over the network. Rather than sending complete images, we transmit the state of the soccer game (which is much smaller) and expand this state locally to 3D images. This is especially important for communication over (slow) wide-area networks.

In our current and future work, we also study other scientific applications that can exploit interactive and collaborative visualization [19]. Example applications we intend to investigate are simulation of nonlinear systems (e.g., lasers), based on our earlier work described in [13], visualization of molecular dynamics, and interactive visualization of the cornea of the human eye [27, 28]. Such scientific applications will exhibit similar problems as those identified for RoboCup. For example, visualization of the cornea would be very useful during eye surgery, but the modeling of the cornea is a computation-intensive process, resulting in latency problems. We are currently studying how parallel cluster computing [1] may reduce the latency and allow the visualization to be done in real-time.

References

[1] H. Bal, R. Bhoedjang, R. Hofman, C. Jacobs, K. Langendoen, T. Rühl, and F. Kaashoek. Performance Evaluation of the Orca Shared Object System. *ACM Transactions on Computer Systems*, 16(1):1–40, Feb. 1998.

[2] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart. The CAVE: audio visual experience automatic virtual environment. *Communications of the ACM*, 35(6):64–72, June 1992.

[3] S. M. Drucker and D. Zeltzer. CamDroid: A system for implementing intelligent camera control. In *1995 Symposium on Interactive 3D Graphics*, pages 139–144, Apr. 1995.

[4] A. Foster and C. Kesselman. *The Grid: Blueprint for a New Computer Infrastructure*. Morgan Kaufman, 1998.

[5] H. Iwata. Walking About Virtual Environments on an Infinite Floor. In *IEEE Virtual Reality'99*, pages 286–293, 1999.

[6] A. Johnson, J. Leigh, and J. Costigan. Projects in VR: Multiway tele-immersion at Supercomputing 97. *IEEE Computer Graphics and Applications*, 18(4):6–9, July/Aug. 1998.

[7] P. Karp and S. Feiner. Automated presentation planning of animation using task decomposition with heuristic reasoning. In *Graphics Interface '93*, pages 118–127, May 1993.

[8] H. Kitano, M. Veloso, P. Stone, M. Tambe, S. Coradeschi, E. Osawa, I. Noda, H. Matsubara, and M. Asada. The RoboCup Synthetic Agents Challenge 97. In M. Pollack, editor, *15th International Joint Conference on Artificial Intelligence*, pages 24–29, 1997.

[9] W. Lamotte, E. Flerackers, F. Van Reeth, R. Earnshaw, and J. De Matos. Visinet: Collaborative 3D Visualization and VR over ATM Networks. *IEEE Computer Graphics & Applications*, 17(2):66–75, Mar.-Apr. 1997.

[10] J. Leech, J. Prins, and J. Hermans. SMD: Visual Steering of Molecular Dynamics for Protein Design. *IEEE Computational Science & Engineering*, 3(4):38–45, Winter 1996.

[11] J. Leigh, A. Johnson, T. DeFanti, and M. Brown. A Review of Tele-Immersive Applications in the CAVE Research Network. In *IEEE Virtual Reality'99*, pages 180–187, 1999.

[12] D. Levine, M. Facello, P. Hallstrom, G. Reeder, B. Walenz, and F. Stevens. Stalk: An Interactive System for Virtual Molecular Docking. *IEEE Computational Science*, 4(2):55–65, April-June 1997.

[13] C. Mirasso, M. Mulder, H. Spoelder, and D. Lenstra. Visualization of the Sisyphus Attractor. *Computers in Physics*, 11(3):282–286, May/June 1997.

[14] T. Molet, A. Aubel, T. Gapin, S. Carion, E. Lee, N. Magnenat-Thalmann, H. Noser, I. Pandzic, G. Sannier, and D. Thalmann. Anyone for Tennis? *Presence*, 8(2):140–156, Apr. 1999.

[15] J. Mulder, J. van Wijk, and R. van Liere. A Survey of Computational Steering Environments. *Future Generation Computer Systems*, 13(6), 1998.

[16] K. S. Park and R. Kenyon. Effects of Network Characteristics on Human Performance in a Collaborative Virtual Environment. In *IEEE Virtual Reality'99*, pages 104–111, 1999.

[17] S. Parker, M. Miller, C. Hansen, and C. Johnson. An Integrated Problem Solving Environment: the SCIrun Computational Steering System. In *Hawaii International Conference of System Sciences*, pages 147–156, Jan. 1998.

[18] Center for Parallel Computers, Royal Institute of Technology, Stockholm. http://www.pdc.kth.se.

[19] L. Renambot, H. Bal, D. Germans, and H. Spoelder. CAVEStudy: an Infrastructure for Computational Steering in Virtual Reality Environments. Technical report, Vrije Universiteit Amsterdam, Faculty of Sciences, Mar. 2000.

[20] SARA, Academic Computing Services Amsterdam. http://www.sara.nl.

[21] K. Simsarian, L. Fahlen, and E. Frecon. Virtually Telling Robots What To Do. In *Informatique Montpellier 1995, Interface to Real and Virtual worlds.*, 1995.

[22] S. Singhal and M. Zyda. *Networked Virtual Environments: Design and Implementation*. Addison-Wesley, 1999.

[23] S. K. Singhal. Effective remote modeling in large-scale distributed simulation and visualization environments. Ph.D. Thesis CS-TR-96-1574, Stanford University, Department of Computer Science, Sept. 1996.

[24] H. Spoelder, L. Renambot, D. Germans, H. Bal, and F. Groen. Man Multi-Agent Interaction in VR: a Case Study with RoboCup. In *IEEE Virtual Reality 2000 (poster)*, New Brunswick, NJ, Mar. 2000.

[25] V. E. Taylor, J. Chen, T. L. Disz, M. E. Papka, and R. Stevens. Interactive Virtual Reality in Simulations: Exploring Lag Time. *IEEE Computational Science & Engineering*, 3(4):46–54, 1996.

[26] M. Usoh, K. Arthur, M. Whitton, R. Bastos, A. Steed, M. Slater, and F. Brooks. Walking $>$ Walking-in-Place $>$ Flying, in Virtual Environments. In *Computer Graphics, ACM SIGGRAPH '99 Proceedings*, 1999.

[27] F. Vos and H. Spoelder. Visualization in Corneal Topography. In *IEEE Visualization'98*, pages 427–430, Oct. 1998.

[28] F. Vos, G. van der Heijde, H. Spoelder, I. van Stokkum, and F. Groen. A new PRBA-based Instrument to Measure the Shape of the Cornea. *IEEE Trans. on Instrum. Meas.*, 46(4):794–797, 1997.