A FRAMEWORK FOR TELEPRESENT GAME-PLAY IN LARGE VIRTUAL ENVIRONMENTS

Patrick Rößler, Frederik Beutler, and Uwe D. Hanebeck
Intelligent Sensor-Actuator-Systems Laboratory
Institute of Computer Science and Engineering
Universität Karlsruhe (TH)
Karlsruhe, Germany
Email: {patrick.roessler | beutler | uwe.hanebeck}@ieee.org

Keywords: Extended Range Telepresence, Motion Compression, Virtual Reality.

Abstract: In this paper we present a framework that provides a novel interface to avatar control in immersive computer games. The user's motion is tracked and transferred to to the game environment. This motion data is used as control input for the avatar. The game graphics are rendered according to the avatar's motion and presented to the user on a head-mounted display. As a result, the user immerses into the game environment and identifies with the avatar. However, without further processing of the motion data, the virtual environment would be limited to the size of the user's real environment, which is not desirable. By using Motion Compression, the framework allows exploring an arbitrarily large virtual environment while the user is actually moving in an environment of limited size. Based on the proposed framework, two game applications were implemented, a modification of a commercially available game and a custom designed game. These two applications prove, that a telepresence system using Motion Compression is a highly intuitive interface to game control.

1 INTRODUCTION

Telepresence gives a human user the impression of being present in another environment. This is achieved by having a robot gather visual data of the remote environment and present it to the user, who is wearing a head-mounted display. The user thus perceives the remote environment through the eyes of the robot. In order to extend telepresence to an intuitive user interface, the motion of the user's head and hands is tracked and transferred to the robot that replicates this motion. As a result, the user identifies with the robot, i. e., he is telepresent in the remote environment.

Of course, this technique is also applicable to virtual reality games, where the user controls an avatar instead of a robot. Using telepresence as an input to the computer, the user experiences a high degree of immersion into the game's virtual environment and identifies fully with the avatar. Thus, telepresence techniques provide an appropriate interface for intuitive avatar control.

Common input devices for avatar control like keyboards, mice, joysticks, and game pads, all lack the ability of controlling the avatar intuitively. A possible approach for controlling an avatar in a game environment by means of immersive interfaces is Augmented Reality. This approach is applied in the ARQuake project (Piekarski and Thomas, 2002), where virtual objects from the game are superimposed onto a real environment. This system, however, only allows virtual environments that feature the same layout as the real environment.

CAVE Quake II (Rajlich, 2001) uses the CAVE Environment (Cruz-Neira et al., 1993), where images are projected onto walls of a box surrounding the user, in order to provide a realistic impression of the first person game Quake II to the user. The motion of a tool called wand is tracked for avatar movement. Interfaces like this, however, are known for producing an impression that resembles flying, rather than walking.

Other approaches use walking-in-place metaphors (Slater et al., 1994) or complex mechanical setups (Iwata, 1999), (Iwata et al., 2005) to allow free natural locomotion in virtual environments. However, it is not known, that these systems feature the typical user motion of computer games.

This paper presents a framework that combines immersive computer games and extended range telepresence by means of Motion Compression (Nitzsche et al., 2004). Motion Compression allows the user in a confined user environment to control the avatar in an arbitrarily large virtual world by natural walking. Fig. 1 shows a user wearing a non-transparent head-mounted display while playing an immersive game.

According to several sources, (Peterson et al., 1998), (Darken et al., 1999), and (Tarr and Warren, 2002), there is evidence, that using full body motion, e. g. normal walking, results in better navigation in the virtual environment than using common input devices. By using the approach presented in this paper, the user is expected to feel present in the virtual world and identify well with the avatar under control.

Figure 1: User playing an immersive game with a telepresence interface.

The remainder of this paper is structured as follows. Section 2 reviews Motion Compression, as it is a major part of the proposed framework. An overview on the framework is given in section 3. Section 4 describes the tracking system and section 5 presents two different games that use this framework. An experimental evaluation of the suitability for intuitive avatar control is given in section 6. Finally, conclusions are drawn in section 7.

2 MOTION COMPRESSION

The Motion Compression algorithm transforms the user's position and orientation in the *user environment*, i. e., the physical world surrounding the user, into the *target environment*, which in this application is the virtual environment of the game (Fig. 2). The target environment is perceived visually by the user wearing a non-transparent head-mounted display, which makes the physical world invisible for him. The effect is that he moves in the user environment but feels present in the virtual environment instead.

The Motion Compression algorithm is partitioned into three functional modules: *path prediction*, *path transformation*, and *user guidance*.

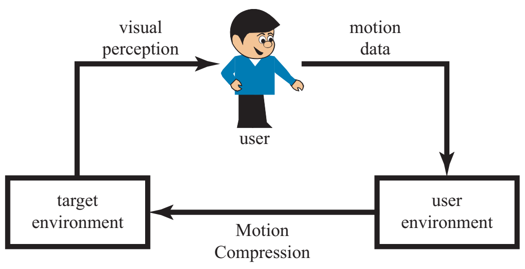

Figure 2: Overview of the different Motion Compression environments.

The *path prediction* unit predicts the path the user wants the avatar to follow in the target environment. This prediction is based on the user's view direction and, if available, additional information on the target environment. The resulting path is called *target path*.

Path transformation maps the target path onto the *user path* in the user environment. Since the user environment is in most cases smaller than the target environment, the target path cannot always be mapped directly. Motion Compression aims at giving users a realistic impression of the virtual environment by keeping distances and turning angles in the user environment and target environment locally equal. Thus, only the curvature of the path is changed. A possible target path and the corresponding user path is illustrated in Fig. 3. To give the user a realistic impression of controlling the avatar, the resulting curvature deviation is kept at a minimum.

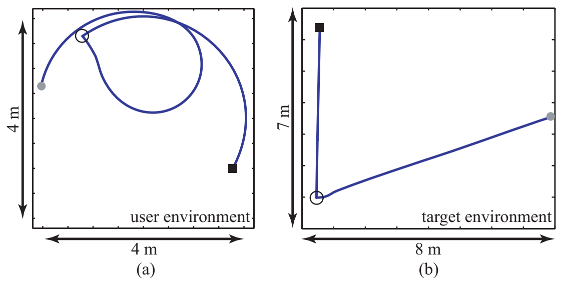

Figure 3: User path (a) and corresponding target path (b).

When walking, humans continuously check if they are on the direct way to their desired target and adjust their direction accordingly. This behavior is exploited in *user guidance*. While moving along, the avatar's orientation in the target environment is changed in such a way, that the user follows the path by correcting the perceived deviations.

As a result of the three processing steps Motion Compression provides a linear, but location-variant transformation from the user's position to the avatar's position in the target environment.

3 SOFTWARE FRAMEWORK

We designed a software framework, which allows connecting arbitrary game environments to Motion Compression control. This framework, however, is not limited to game applications, but can also be used for controlling teleoperators in telepresence scenarios (Rößler et al., 2005). In order to provide an extensible interface that may be adapted to future applications, we decided to base the framework on the well known CORBA middleware standard. Another advantage of CORBA is, that it is platform independent and available for virtually any programming language.

As shown in Fig. 4, the core of the setup is a CORBA server, the MC Server, which contains an implementation of the Motion Compression algorithm. In order to provide an up-to-date target position, the server runs asynchronously and constantly accepts updates of the user position from the tracking subsystem, which acts as a CORBA client. Based on these position updates MC Server calculates the current transformation and target position. The target position is made available to be fetched by the game client.

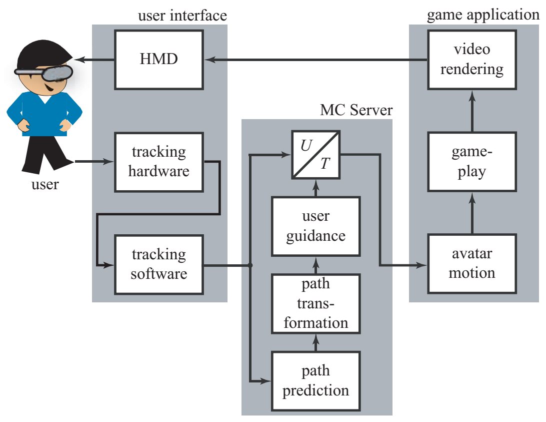

Figure 4: Data flow in the software framework.

The communication between the game application, which is a CORBA client, and the MC Server requires a data connection. This connection is established when the game is started. During the game, the connection is used to continuously refresh the target position and the avatar is moved accordingly. The connection is maintained until the game is quit. The cooperation of the game application and the MC Server is illustrated in detail in Fig. 5. The game is also responsible for game-play and rendering the first person view of the game-environment. These rendered images are displayed to the user on a head-mounted display.

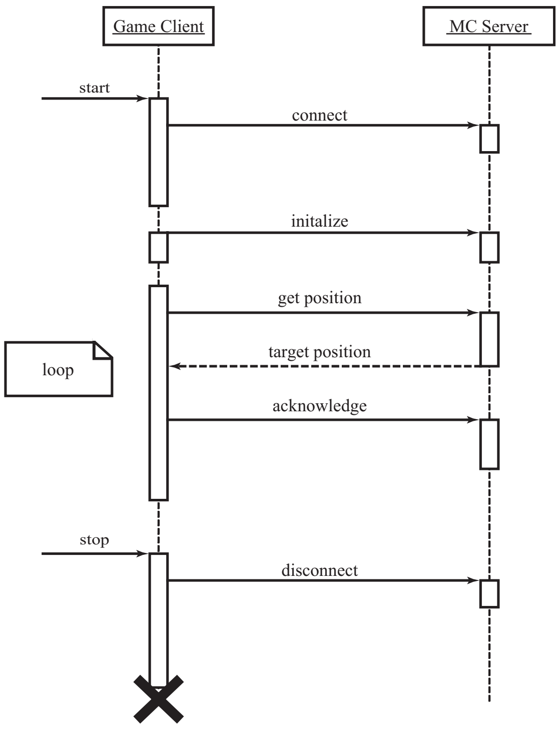

Figure 5: Collaboration of the game client and the MC Server module.

4 TRACKING SYSTEM

For the estimation of the user's posture, i. e., translation and orientation, an acoustic tracking system is used. This system consists of four loudspeakers, which are placed in the corners of the user environment, emitting distinct acoustic signals.

These signals are received by four microphones attached to the head-mounted display. In order to estimate the time delay between sending and receiving the signal, the cross correlation between the filtered signal and the transmitted signal is calculated. The estimated time delay is converted to the range based on the velocity of sound.

Based on the arrangement of four loudspeakers and four microphones 16 estimated ranges are available. These range estimates are used in a gradient descent algorithm to estimate the posture of the user's head. The initial values for the gradient descent algorithm are calculated by means of a closed form solution presented in (Beutler and Hanebeck, 2005). The tracker data is fused with information from a gyroscope cube by means of a Kalman Filter, resulting in more accurate estimates for the orientation. Fig. 6 shows the setup of the tracking system attached to the head-mounted display.

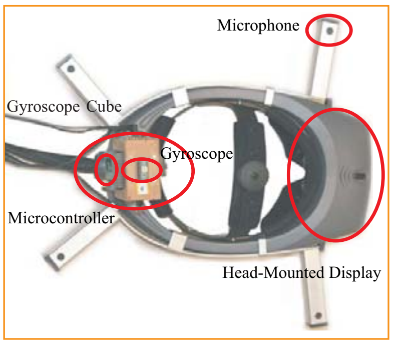

Figure 6: Top view on the hardware setup consisting of four microphones and a gyroscope cube mounted on a head-mounted display.

5 GAME APPLICATIONS

In order to prove the applicability of the proposed software framework, two game applications were implemented.

5.1 Quake

The first game application is a modification of the commercially available first person game Quake III Arena (id Software, 2001). The resulting modification is called *MCQuake*.

In *MCQuake* the modifications include a CORBA client that implements the interface required by the software framework. In order to control the avatar, *MCQuake* simulates normal user input from the positions received from MC Server. Based on the the last position of the avatar and on the commanded position a motion vector is calculated, which is handed to the game engine. The game engine now moves the avatar accordingly. By doing a collision detection, it prevents the avatar from moving through walls.

The height of the target position has also to be mapped onto the avatar's position in the virtual environment. The virtual environment supports two kinds of height information, which are mapped differently as described below.

The first kind of height information is the absolute height of the avatar. This height is unrestricted allowing the avatar to climb stairs. Absolute height is handled by the game engine itself. If, for example, the user maneuvers the avatar over a set of stairs, the avatar's absolute height changes with the height of the floor beneath him.

The second kind of height information, called view height, is relative to the floor the avatar moves on. In the game, however, it is restricted to only two different values used for crouched and normal movement. A threshold applied to the user's view height in the user environment, determines which kind of movement is to be used.

Given these mappings, the user is now able to control the avatar in an intuitive way in arbitrarily large virtual environments by normal walking. Of course, *MCQuake* also supports other kinds of motion, like running and strafing, which are very common in first person games. For Motion Compression, there is no difference between those and normal walking as motion is always handled as a sequence of position updates.

5.2 PacMan

A second game application is a custom build first person telepresence version of the arcade game classic PacMan, called *paMCan* (PacMan with a Motion Compression driven artificial environment) as shown fin Fig. 7. While *MCQuake* is written in C, *paMCan* is written completely in python. In order to obtain high quality graphics output, cgkit (Baas, 2005) was used as graphics back end. For *paMCan* the path prediction module was modified in such a way, that it uses not only view direction, but also the virtual map layout.

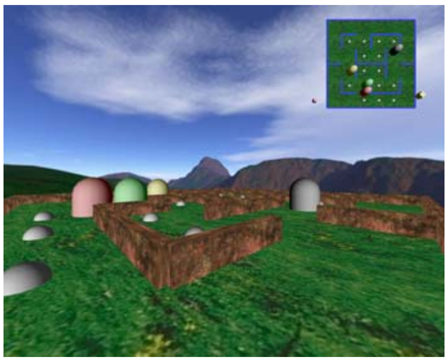

Figure 7: An impression of the *paMCan*-game.

Motion commands are handled similar to *MCQuake*. The game application computes a motion vector and moves the avatar accordingly if no obstacles block the way. However, if there are obstacles, the motion vector is modified in accordance with the map information. This prevents the avatar from moving through walls. Although modifying the motion vector leads to a displacement of the commanded avatar positions and the actual avatar position, this has no effect on user and game. In *paMCan* view height is the only height information and is mapped directly.

6 EXPERIMENTAL EVALUATION

In order to gain a high degree of realism the setup uses a high quality head-mounted display with a resolution of 1280×1024 Pixels per eye and a field of view of $60°$. Both the game engine and the MC Server, run on a multimedia Linux-PC, which allows a frame rate of approximately 80 images/sec for *MCQuake* and 60 images/sec for *paMCan*.

The acoustic tracking system currently provides 15 estimates per second for the position and 50 estimates per second for the orientation, which is enough for the given application. Fig. 8 illustrates a motion trajectory recorded by the tracking system. The vectors are directed in view directions. It can be observed, that the tracking system has a good relative accuracy, which is very important for the given application to avoid shaking images. Absolute accuracy, or ground truth, is of less importance. If the tracking system detects outliers, it discards the measurement and provides no estimate.The tracking system again provides reasonable estimates after some acoustic measurements.

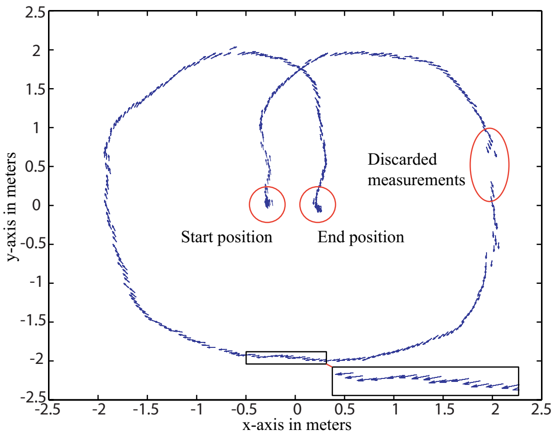

Figure 8: The estimated translation sequences in a test run with a predefined motion trajectory.

In order to test the users' ability to navigate properly in the virtual environment, an environment well-known to the users was chosen. Hence the map from MensaQuake (The MensaQuake Project, 2002) was loaded into *MCQuake*. This map is a realistic model of the student cafeteria of the University of Karlsruhe (Fig. 9).

The experiment compares the time a user needs to navigate his avatar along a specified path in the virtual cafeteria using *MCQuake* and Quake with keyboard and mouse as inputs. In addition the user was asked to walk the same path in the real cafeteria. In order to avoid effects of user adaptation, a user was chosen for the experiment, who was experienced in both, using Motion Compression and Quake with standard input devices. He was also familiar with the cafeteria. The path was partitioned into three parts (a), (b), and (c), as shown in Fig. 10.

Figure 9: Impression of the student cafeteria in *MCQuake*. (The MensaQuake Project, 2002)

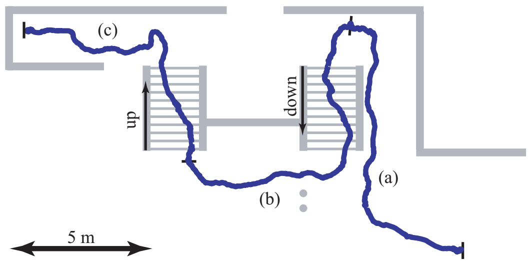

Figure 10: Actual user path from the completion time experiment in the virtual cafeteria.

Table 1 gives a comparison of the completion times gathered in this experiment. When using Quake with standard inputs the avatar reaches his goal much faster than in *MCQuake*. This is a result of unrealistically high walking speed in standard Quake, even when running mode is turned off. A comparison with walking in the real cafeteria shows, that *MCQuake* provides a more realistic motion. Thus, the gaming experience is more realistic than with common game control.

	(a)	(b)	(c)	total
Quake	4.8 s	4.7 s	4.1 s	13.4 s
real	8.9 s	14.0 s	15.4 s	38.2 s
MCQuake	**15.0 s**	**15.1 s**	**14.4 s**	**44.5 s**

Table 1: Average time for a specified path from three runs in the virtual and real cafeteria, respectively.

It can be observed, that users using Motion Compression for the first time start with a few cautious steps. After several minutes of adjustment, however, they adapt to the system and are able to navigate intuitively through the target environment.

When playing *paMCan* the users adapted to the system even faster. In fact, all three testers stated, that they did not notice the influence of Motion Compression at all. This fact may be a result of *paMCan's* dynamic environment, which provides much more distractions to the user than most other target environments. In *paMCan* the user has to collect pills, escape from ghosts, and navigate through a narrow maze, leaving him less time to focus on the inconsistency of visual and proprioceptive feedback.

7 CONCLUSIONS

Telepresence techniques were designed for controlling robots remotely. Since the remote environment can easily be replaced by a virtual environment, telepresence techniques can also be used to control an avatar in a first person game.

This paper presented a CORBA-based framework for telepresent game-play in large virtual environments using Motion Compression. The algorithm allows a user to control the avatar intuitively through large virtual environments, by actual locomotion in a limited user environment. This framework was tested with two different game applications, *MCQuake* and *paMCan*. Motion Compression proved to be very intuitive as an input for the virtual reality games. As a result, users had a realistic impression of the virtual environment and, thus, experienced a high degree of presence.

In order to give the users the possibility to experience the virtual environment with all senses, we will implement a haptic feedback device, which allows to feel obstacles and weapon recoil. This will lead to an even higher degree of immersion.

The authors believe, that this new kind of gaming experience will lead to a revolution in how people experience computer games. We expect systems like this to become omnipresent in gaming halls in the next couple of years. As soon as the hardware is affordable, people might even start installing these systems in their homes.

ACKNOWLEDGEMENTS

The authors thank two teams of students for the implementation of the games. Henning Groenda and Fabian Nowak implemented *MCQuake*. *paMCan* was written by Jens Kübler, Jan Wassenberg, and Lutz Winkler.

REFERENCES

Baas, M. (2005). cgkit – The Python Computer Graphics Kit. `http://cgkit.sourceforge.net/`.

Beutler, F. and Hanebeck, U. D. (2005). Closed-Form Range-Based Posture Estimation Based on Decoupling Translation and Orientation. In *Proceedings of IEEE Intl. Conference on Acoustics, Speech, and Signal Processing (ICASSP05)*, pages 989–992, Pennsylvania, PA, USA.

Cruz-Neira, C., Sandin, D. J., and DeFanti, T. A. (1993). Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. In *Proceedings of the 20th ACM Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1993)*, pages 135–142, Anaheim, CA, USA.

Darken, R. P., Allard, T., and Achille, L. B. (1999). Spatial Orientation and Wayfinding in Large-Scale Virtual Spaces ii. *Presence*, 8(6):iii–vi.

id Software (2001). Quake III Arena. `http://www.idsoftware.com/games/quake/quake3-arena`.

Iwata, H. (1999). The Torus Treadmill: Realizing Locomotion in VEs. *IEEE Computer Graphics and Applications*, 19(6):30–35.

Iwata, H., Yano, H., Fukushima, H., and Noma, H. (2005). CirculaFloor. *IEEE Computer Graphics and Applications*, 25(1):64–67.

Nitzsche, N., Hanebeck, U. D., and Schmidt, G. (2004). Motion Compression for Telepresent Walking in Large Target Environments. *Presence*, 13(1):44–60.

Peterson, B., Wells, M., Furness III, T. A., and Hunt, E. (1998). The Effects of the Interface on Navigation in Virtual Environments. In *Proceedings of Human Factors and Ergonomics Society 1998 Annual Meeting*, volume 5, pages 1496–1505.

Piekarski, W. and Thomas, B. (2002). ARQuake: The Outdoor Augmented Reality Gaming System. *ACM Communications*, 45(1):36–38.

Rajlich, P. (2001). CAVE Quake II. `http://brighton.ncsa.uiuc.edu/~prajlich/caveQuake`.

Rößler, P., Beutler, F., Hanebeck, U. D., and Nitzsche, N. (2005). Motion Compression Applied to Guidance of a Mobile Teleoperator. In *Proceedings of the IEEE Intl. Conference on Intelligent Robots and Systems (IROS'05)*, Edmonton, AB, Canada.

Slater, M., Usoh, M., and Steed, A. (1994). Steps and Ladders in Virtual Reality. In *ACM Proceedings of VRST '94 - Virtual Reality Software and Technology*, pages 45–54.

Tarr, M. J. and Warren, W. H. (2002). Virtual Reality in Behavioral Neuroscience and Beyond. *Nature Neuroscience Supplement*, 5:1089–1092.

The MensaQuake Project (2002). MensaQuake. `http://mensaquake.sourceforge.net`.