Telepresence Techniques for Controlling Avatar Motion in First Person Games

Henning Groenda[1], Fabian Nowak[1], Patrick Rößler[1], and Uwe D. Hanebeck[1]

Intelligent Sensor-Actuator-Systems Laboratory
Institute of Computer Science and Engineering
Universität Karlsruhe (TH)
Karlsruhe, Germany
{groenda|nowak}@ira.uka.de, {patrick.roessler|uwe.hanebeck}@ieee.org

Abstract. First person games are computer games, in which the user experiences the virtual game world from an avatar s view. This avatar is the user s alter ego in the game. In this paper, we present a telepresence interface for the rst person game Quake III Arena, which gives the user the impression of presence in the game and thus leads to identi cation with his avatar. This is achieved by tracking the user s motion and using this motion data as control input for the avatar. As the user is wearing a head-mounted display and he perceives his actions a ecting the virtual environment, he fully immerses into the target environment. Without further processing of the user s motion data, the virtual environment would be limited to the size of the user s real environment, which is not desirable. The use of Motion Compression, however, allows exploring an arbitrarily large virtual environment while the user is actually moving in an environment of limited size.

1 Introduction

Telepresence usually describes the state of presence in a remote target environment. This can 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 robot imitates the motion of the user's head and hands, which are tracked. As a result, the user only perceives the target environment, and his actions affect this environment, the user identifies with the robot, i. e., he is telepresent in the remote environment [1].

Of course, this technique can also be used in virtual reality, where the user controls an avatar instead of a robot. One of the biggest markets for virtual reality is first person games, i. e., games, where the user perceives the environment through the eyes of one of the game's characters, his avatar. Using telepresence as a user interface to this kind of games, 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 are keyboards, mice, joysticks and game pads which all lack the ability of controlling the avatar intuitively. Force feedback, a simple kind of haptic feedback, improves the gaming experience, but does not provide more intuitive avatar control.

There are several applications, that allow controlling avatars in immersive game environments. ARQuake [2] is an Augmented Reality interface to such a game. That means, in ARQuake the user perceives the real world that is augmented with computer-generated objects through a semi-transparent head-mounted display. Another approach is applied in CAVE Quake II [3], which uses the CAVE Environment [4] to give the user realistic impression of the first person game Quake II. In order order to control the avatar, the user moves a device called wand, which resembles a joystick with six degrees of freedom. In other work, the user navigates an avatar through virtual environments by means of a walking-in-place metaphor [5] or by using complex mechanical devices, e. g. an omnidirectional treadmill [6].

Our approach is to combine first person games and extended range telepresence by means of Motion Compression [7]. Motion Compression allows the user in a confined user environment to control the avatar in an arbitrarily large virtual world by natural walking. The virtual world is presented to the user by a non-transparent head-mounted display and thus, the user is unable to perceive the physical world. As the avatar is controlled by normal walking, this system creates a high degree of realism. Additionally, distances and turning angles are kept locally equal in the physical and virtual world, which makes the avatar control intuitive. As the user has both visual and proprioceptive feedback this approach results in better navigation in the virtual environment, than when using common input devices [8 10].

A basic knowledge of Motion Compression is crucial for understanding how a virtual environment of arbitrary size can be mapped to the physical environment. Hence, the next section gives a brief review of this technique. Several possible solutions for connecting a first person game to an extended range telepresence system are discussed in section 3. The best solution is developed and the final implementation is described in section 4. Section 5 gives an experimental evaluation of our approach.

2 Motion Compression

Motion Compression transforms the user's path in the user environment into the target environment. The *user environment* consists of the physical world surrounding the user, while the *target environment* is, in this application, the virtual environment of the game. Figure 1 illustrates the relations between the different environments. Motion Compression consists of three functional units described below.

The *path prediction* unit predicts the path the user wants the avatar to take in the virtual environment. This path is called *target path* and is required by the next unit. The path is predicted based on head motion and information about

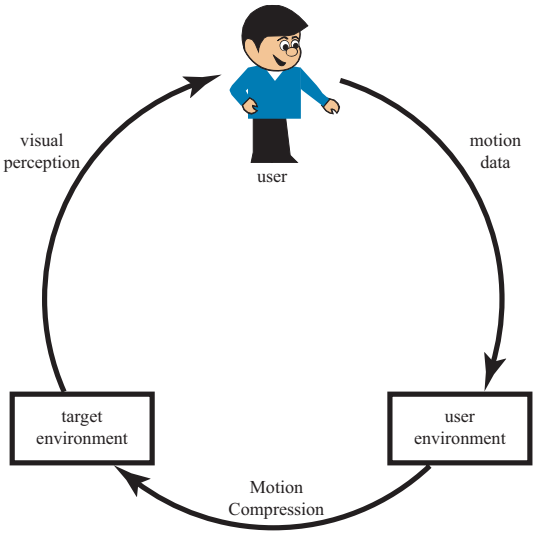

Fig. 1. Overview of the di erent Motion Compression environments.

the virtual environment. A basic approach only uses the avatar's current gaze direction for prediction. The path from the current position to a position in a fixed distance in gaze direction is assumed as the target path. This approach can be used for any virtual or remote environment.

The *path transformation* unit maps the target path onto a path fitting into 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 virtual environment locally equal. Thus, only the curvature of the path is changed. A possible target path and its transformation are illustrated in figure 2. To give the user a realistic impression of controlling the avatar, the resulting curvature deviation is kept at a minimum. In [7], it is shown that users felt comfortable walking in a virtual environment even with a relatively large curvature deviation.

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 for *user guidance*. When walking, the avatar's gaze direction is rotated slightly. The user compensates for this rotation and thus follows the transformed path in the user environment, while the avatar moves on the desired target path.

3 Possible Implementation Approaches

There are three possible approaches for employing Motion Compression to a first person game. These approaches are discussed below.

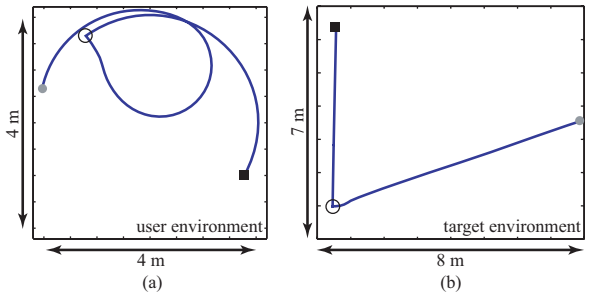

Fig. 2. The left picture shows the transformed target path in the user environment. The right picture shows the corresponding target path in the target environment.

3.1 System Driver

A system driver is a piece of software responsible for connecting hardware devices to the operating system, which in return provides applications access to these devices. Game environments usually only accept input from keyboards, joysticks, game pads, and mice. A driver could be implemented which sends simulated key presses or mouse movements to the game depending on the user's locomotion.

First person games are usually controlled by more than one input device at once. Thus, it has to be taken care that the simulated device is equipped with a su cient number of input keys and analogue inputs for controlling the avatar in all necessary ways.

A long period of manual calibration is required in order to find out how long a key press has to be simulated because no position and orientation information from the game environment can be obtained. This calibration procedure needs to be done only once per game, and any game accepting the driver's simulated input device can be supported. Unfortunately, small deviations between intended and executed movements cannot be prevented because manual calibration is error prone. Furthermore, this solution is confined to a specific operating system.

3.2 Communication Software

The idea of communication software like LIRC [11] is sending signals by use of Inter-Process-Communication to other applications. These signals are system messages which contain information about mouse movements, key presses and releases, and so on. This approach is less involved in the operating system than the driver approach and thus only depends on the operating system family. However, using a technique similar to the system driver, this approach requires a similar calibration effort.

3.3 Modification of the Game

The companies developing first person games often provide parts of the game's source code, additional tools, and documentation to the fan community. Hence, the community is able to create modifications which alter game play, design new weapons or power-ups, and create new enemies. Modifying the game's source code is also the approach chosen for ARQuake. Changes were made in order to display the correct part of the virtual map. A detailed description can be found in [12]. Of course, the modification of an arbitrary game cannot be used for controlling an avatar in any other game but it apparently runs on any platform the game runs on.

A modification also offers direct manipulation of the position and orientation of the avatar, rendering calibration unnecessary. This feature can be used for exactly placing the avatar on a specific position with a specific view angle and thus, any deviation between target and virtual environment is prevented. We chose this approach for implementation because of the capabilities for precise manipulation of the avatar's position and orientation.

4 Implementation

The first person game Quake III Arena [13] was chosen to be modified because it is widespread, up-to-date, and renders fast and effectively. The resulting modification of Quake III Arena is called *MCQuake*.

MCQuake is connected to the hardware/software framework for telepresent game-play, which is presented in [14]. This framework provides two modules. The first of these modules, the user interface, is responsible for tracking the user's position and orientation as well as presenting the camera images to the user. The second module, MC Server, is a CORBA server that provides an implementation of the Motion Compression algorithm. This implementation calculates the target position, which is used as the position of the avatar in *MCQuake*, which implements the third module and thus completes the framework. The data flow between the user, MC Server, and *MCQuake* is shown in figure 3.

When *MCQuake* is started a data connection to MC Server is established. During the game, this connection is used to continuously fetch the target position.The connection is maintained until the game is quit. The target position fetched from the Motion Compression implementation can be used in two different ways for moving the avatar.

The first possibility is to set the position of the avatar exactly to the user's target position. In this case, the collision detection of the game engine is bypassed as the avatar is not moved to the new position, but is directly set there in the virtual environment. Since MC Server has no information about obstacles in the target environment, the avatar can walk through them. However, this behavior is not desirable in most cases.

Therefore, an alternative way of moving the avatar was implemented, that uses the game's collision detection. This is achieved by calculating a motion

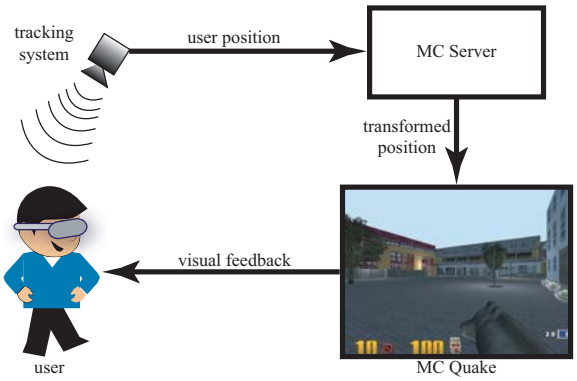

Fig. 3. Data flow between *MCQuake*, the user, the Motion Compression implementation, and the tracking module.

vector as the difference between the avatar's current position and the commanded position. This vector is then handed to the game engine, which now checks for collisions.

In both ways, 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, objects, and the floors in the virtual environment. This height is unrestricted allowing the avatar to fall, climb stairs, and walk on slopes. Since common input devices for first person games control avatar movement by commanding only two-dimensional moving directions, the game engine handles the avatar's absolute height 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. Of course, changes of the absolute height cannot be simulated in the user environment. Thus, using these manipulation methods in *MCQuake* allows to move the avatar by normal walking in the user environment without restricting the virtual environment to only one fixed height.

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. In the user environment, the tracking unit also provides the user's view height relative to the physical floor, but as the user's view height is not restricted to exactly two values, direct mapping is not possible. Nevertheless, crouched movement can be supported by defining a threshold. As long as the user's view height is below this threshold, the avatar crouches.

Given these mappings, the user is now able to control the avatar in an intuitive way through 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 handled as a sequence of position updates.

5 Experimental Evaluation

In the experimental setup, the user interface includes a non-transparent head-mounted display with a resolution of 1280×1024 Pixels per eye and a field of view of 60°. This high quality display ensures a realistic impression of the game environment. The position and orientation of the head-mounted display is tracked with an acoustic tracking system. Fig. 5 shows the hardware setup of the user interface in the user environment.

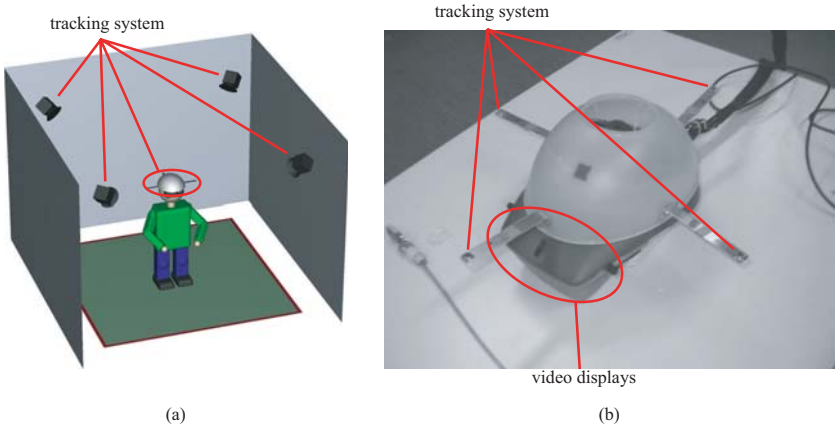

Fig. 4. Schematic view of the user environment equipped with four emitters for the acoustic tracking system (a). The head-mounted display with four receivers attached to it (b).

In order to properly test if the users experienced the virtual environment like the real world, an environment well-known to the users was chosen. Hence the map from MensaQuake [15] was used. This map is a realistic model of the cafeteria of the University of Karlsruhe. An impression of this map is given in figure 5(a). Both, the Quake engine and MC Server, run on a multimedia Linux-PC, which allows a frame rate of approximately 80 images/s. The limiting factor of the setup is currently the tracking system, which gives 15 updates per minute. Development of the tracking system, however, aims at a better accuracy and a higher update rate [16] and was already enhanced with a gyroscope cube for better orientation estimation [1].

Ten users without any experience concerning telepresence control by Motion Compression were precisely observed when exploring the virtual cafeteria and asked about their impressions. Figure 5(b) shows a user playing *MCQuake*.

(a) (b)

Fig. 5. An impression of the student cafeteria in Quake [15] (a). A user playing *MCQuake* (b).

In the current setup, the head-mounted display is supplied with video data by a strong and inflexible cable, which is, of course, invisible to the user. A second person is needed in order to take care of the cable and thus prevent the user from getting caught by it. This problem, however, can be addressed by building a wearable computer system for rendering game graphics, that communicates wirelessly with MC Server.

Some users also stated to feel a little bit uncomfortable by the sudden change between normal and crouched view height when passing the threshold responsible for crouching. This is due to users expecting the virtual environment to reflect their own view height and thus proves a high degree of feeling present in the virtual environment.

Despite the drawbacks stated above, the users's impression of being present in the virtual environment did not decrease. It was further observed that after a few cautious steps all users were able to navigate intuitively and identified with the avatar, as was confirmed by the users afterwards.

In order to obtain a more quantitative analysis of performance, a second experiment was conducted. This experiment compares the time a user needs to navigate his avatar along a specified path in the virtual cafeteria with *MCQuake* and with 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 adaption, 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. 6.

Table 1 gives a comparison of the times of completion 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

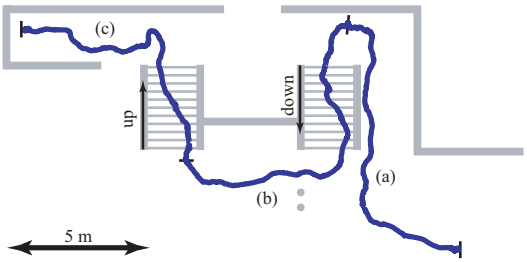

Fig. 6. User path from the time of completion experiment in the virtual cafeteria.

Table 1. Average time for a speci ed path from three runs.

	(a)	(b)	(c)	total
standard 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**

in standard Quake, even when the running mode is turned off. A comparison with the real cafeteria shows, that *MCQuake* provides a more realistic gaming experience than common game control. For example, turning 360 degrees in half a second is no longer possible.

6 Conclusions and Future Work

Telepresence techniques were designed for controlling robots remotely. Since the remote environment can be easily replaced by a virtual environment, telepresence techniques can also be used to control an avatar in a first person game. Possible implementation approaches for connecting a telepresence system using the Motion Compression algorithm to the game Quake III Arena were evaluated. The resulting possibilities were system driver, communication software, and modification of the game environment. The approach of implementing a modification was chosen due to its exact positioning capabilities and the widest range of possibilities for controlling the avatar. Two different ways for moving the avatar were implemented since they offer orthogonal advantages such as exact positioning and in-game collision detection.

In an experimental evaluation, Motion Compression as input for a first person game proved to be very intuitive and succeeded in the user feeling present in the virtual environment leading to a high degree of realism.

These results are consistent with the results from a quantitative experiment. This experiment shows that the system presented in this work, results in completion times very similar to real world scenarios.

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

The authors expect systems like this to become common in gaming halls in the next couple of years. As soon as the hardware is affordable, people will even start installing these systems in their homes.

References

1. Rößler, P., Beutler, F., Hanebeck, U.D., Nitzsche, N.: 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 (2005)
2. Piekarski, W., Thomas, B.: ARQuake: The Outdoor Augmented Reality Gaming System. ACM Communications **45** (2002) 36 38
3. Rajlich, P.: CAVE Quake II. `http://brighton.ncsa.uiuc.edu/~prajlich/caveQuake` (2001)
4. Cruz-Neira, C., Sandin, D.J., DeFanti, T.A.: 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), Anaheim, CA, USA (1993) 135 142
5. Slater, M., Usoh, M., Steed, A.: Steps and Ladders in Virtual Reality. In: ACM Proceedings of VRST 94 - Virtual Reality Software and Technology. (1994) 45 54
6. Iwata, H.: The Torus Treadmill: Realizing Locomotion in VEs. IEEE Computer Graphics and Applications **19** (1999) 30 35
7. Nitzsche, N., Hanebeck, U.D., Schmidt, G.: Motion Compression for Telepresent Walking in Large Target Environments. Presence **13** (2004) 44 60
8. Darken, R.P., Allard, T., Achille, L.B.: Spatial Orientation and Way nding in Large-Scale Virtual Spaces: An Introduction. Presence **7** (1998) 101 107
9. Peterson, B., Wells, M., Furness III, T.A., Hunt, E.: The E ects of the Interface on Navigation in Virtual Environments. In: Proceedings of Human Factors and Ergonomics Society 1998 Annual Meeting. Volume 5. (1998) 1496 1505
10. Tarr, M.J., Warren, W.H.: Virtual Reality in Behavioral Neuroscience and Beyond. Nature Neuroscience Supplement **5** (2002) 1089 1092
11. Bartelmus, K.S.C.: LIRC Linux Infrared Remote Control. `http://www.lirc.org` (1999)
12. Thomas, B., Close, B., Donoghue, J., Squires, J., De Bondi, P., Morris, M., Piekarski, W.: ARQuake: An Outdoor/Indoor Augmented Reality First Person Application. In: Proceedings of 4th Intl. Symposium on Wearable Computers, Atlanta, GA, USA (2000) 139 146
13. id Software: Quake III Arena. `http://www.idsoftware.com/games/quake/quake3-arena` (2001)
14. Rößler, P., Beutler, F., Hanebeck, U.D., Nitzsche, N.: A Framework for Telepresent Game-Play in Large Virtual Environments. In: 2nd Intl. Conference on Informatics in Control, Automation and Robotics (ICINCO 2005), Barcelona, Spain (2005)
15. The MensaQuake Project: MensaQuake. `http://mensaquake.sourceforge.net` (2002)
16. Beutler, F., Hanebeck, U.D.: A New Nonlinear Filtering Technique for Source Localization. In: The 3rd IEEE Conference on Sensors (IEEE Sensors 2004), Vienna, Austria (2004) 413 416