Realizing a Low-latency Virtual Reality Environment for Motor Learning

Thomas Waltemate[1] Felix Hülsmann[1,2] Thies Pfeiffer[3] Stefan Kopp[2] Mario Botsch[1]

[1]Computer Graphics Group [2]Social Cognitive Systems [3]CITEC Central Labs
Bielefeld University, Germany

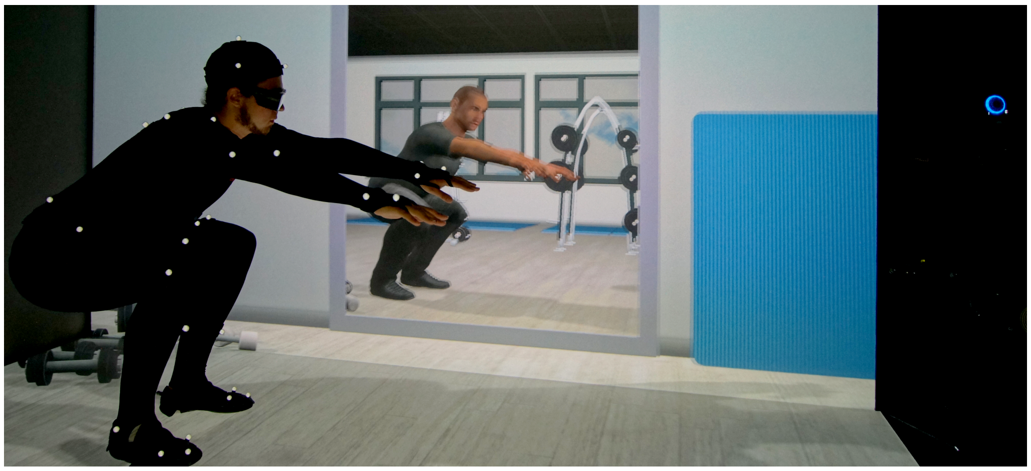
Figure 1: *Real-time feedback using a virtual mirror requires an immersive Virtual Reality environment that provides full-body motion capturing, motion analysis, and realistic character rendering at a low end-to-end latency.*

Abstract

Virtual Reality (VR) has the potential to support motor learning in ways exceeding beyond the possibilities provided by real world environments. New feedback mechanisms can be implemented that support motor learning during the performance of the trainee and afterwards as a performance review. As a consequence, VR environments excel in controlled evaluations, which has been proven in many other application scenarios.

However, in the context of motor learning of complex tasks, including full-body movements, questions regarding the main technical parameters of such a system, in particular that of the required maximum latency, have not been addressed in depth. To fill this gap, we propose a set of requirements towards VR systems for motor learning, with a special focus on motion capturing and rendering. We then assess and evaluate state-of-the-art techniques and technologies for motion capturing and rendering, in order to provide data on latencies for different setups. We focus on the end-to-end latency of the overall system, and present an evaluation of an exemplary system that has been developed to meet these requirements.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual reality I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Tracking

Keywords: Low-Latency, Virtual Reality, Motor Learning

1 Introduction

Learning of new motor tasks or improvement of already known ones is essential in many domains such as fitness training or rehabilitation. Often it is useful and much more efficient to learn motor tasks with the help of a coach or another external source of feedback, since in this way the athlete gets information about whether or not the given motor task was executed correctly and what kind of errors were made.

The extensive capabilities of Virtual Reality (VR) seem to be an ideal candidate to facilitate and boost the learning process [Rizzo and Kim 2005; Schack et al. 2014]: A VR environment can be equipped with high-precision sensors and can employ various feedback channels. Highly precise sensors gather data of the trainee, which are analyzed in real time, in order to provide directed purposeful feedback over various channels. This feedback can either be given after the movement execution or—more interestingly and more useful—during the execution. Especially in the latter case it is important to ensure that the feedback is precisely timed, so that it is presented exactly when it is relevant. As a consequence, the environment has to be highly controlled, i.e., properties like the end-to-end latency or tracking robustness either must be controlled or at least have to be taken into account. It thus seems necessary to report such basic properties of a system in every research addressing issues of motor learning in VR. This would allow researchers to compare systems and to reproduce studies more reliably.

At the time being, no general guidelines for VR environments targeted at motor learning seem to exists. Furthermore, for many systems described in literature, no sufficient information on relevant aspects such as end-to-end latency, robustness of the motion capture system, et cetera is given. Thus when building up a new VR environment, one is faced with a vast number of potential techniques and technologies, but a well-informed choice is hardly possible.

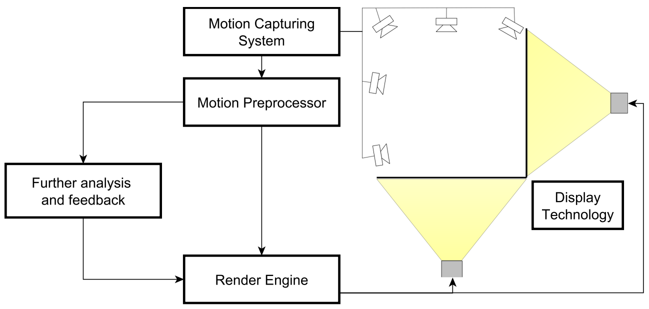

Figure 2: *A minimal architecture for a VR environment for motor learning combines a motion capturing system, motion processing (e.g., for re-targeting or motion analysis), as well as a render engine for high-fidelity character rendering.*

In this paper we aim at improving this situation by

1. providing general requirements towards VR systems for motor learning,
2. evaluating and assessing state-of-the-art techniques and technologies,
3. presenting a system built according to the aforementioned requirements,
4. providing latency measurements of the virtual environment and giving hints on how to reduce latency.

A minimal VR system for motor learning would consist of components for motion capturing, pre-processing of motion data, motion analysis, feedback generation, rendering and display technology (see Figure 2). As rendering and motion capturing are the backbones of the VR environment for motor learning, we focus on these two components in this paper. We want to stress, however, that the choice of display technology is very important, as projectors can be found with latencies in the range between 15 ms and 140 ms. But the decision for display hardware is relatively easy to make based on technical specifications.

In the following, we start by developing general requirements towards motor learning in VR applications (Section 2). After discussing related motor learning approaches in Section 3, we present in Section 4 the essentials of our low-latency VR environment, while also assessing particular state-of-the-art techniques and technologies for motion capturing and real-time rendering. In Section 5, we present an evaluation of our system and report results of a pilot user study, before concluding in Section 6.

2 Requirements

In this section we develop requirements necessary for an efficient motor learning system in VR. Many researchers already pointed out some of the most crucial requirements for VR applications in general: For instance, Bierbaum et al. [2000] provide an overview including general features like low latency, high frame rate, tracking robustness, but also engineering requirements such as extensibility and hardware abstraction. To our knowledge, this has not yet been done for VR systems specialized on motor learning. In the following we therefore carve out the most important requirements.

R1: Feedback on one's own motion

As a first requirement, users have to be able to verify the correct execution of a given motor task by getting feedback of whatever kind. This feedback should be as intuitive as possible, and one of the most intuitive ways is to let users observe their own motion by viewing their own body.

In real world scenarios, like fitness or dance studios, self-monitoring is usually achieved through a mirror. Thus, it seems desirable to provide mirror-like feedback in VR training environments as well [Hämäläinen 2004]. This is inter alia motivated by findings of Chau et al. [2003], who found that none of their proposed layouts of students and teachers could improve upon a standard face-to-face configuration—similar to that of a mirror—when learning Tai Chi. A *virtual* mirror, as planned in our setup, may serve multiple purposes: it may show the optimal performance, just as a teacher would, to guide the performance of the trainee; it can simply reflect the real performance of the trainee to support self-monitoring; or it could add augmentations to the real performance, e.g., emphasizing errors. Finally, it serves as a perfect base for further feedback strategies. Besides face-to-face layouts, a third person view could also improve training results, as has been recently shown by Covaci et al. [2014]. In summary it can be said that self-monitoring is an essential ingredient on the way towards meaningful visual feedback.

R2: Low latency and high frame rate

The times in which any response delay below 1 s was considered acceptable—as had been suggested by Shneiderman [1984]—are long over. Work in the area of system response time suggests that delays in the range of 80–100 ms will not be noticeable by the majority of users. In a study by Mauve et al. [2004] users did not notice network lags below 120 ms, thus, depending on the applications, tolerable latencies might be even higher than 80–100 ms. Gutwin [2002] showed that in a simple coordination task a delay of 200 ms already significantly increased the error rates. However, the examples used in such studies are primarily targeting human-machine interaction or manipulation of objects and do not address the issue of self-perception and self-monitoring.

Research on the effects of latency in the context of virtual environments has been primarily focused in research on distributed virtual environments. In this context, Roberts et al. [1995] define the time required to present the user's actions back to the user as local latency. In a study on collaborative virtual environments, Park et al. [1999] show that with increased latencies, humans adopt a move-and-wait strategy, waiting several seconds to let their views synchronize, before continuing performing their tasks. They showed that in such setups jitter had a larger impact on collaborative performance than latency. The development of similar strategies has to be avoided in our target scenario, as it would hamper with the natural flow of movements.

Regarding display latencies, it has been shown that trained users are able to detect a latency of perspective adaptation of about 15 ms in a HMD-based study [Mania et al. 2004]. In CAVE- or Powerwall-based VR systems, latency is less critical, as the projection screens remain stationary. However, a highly responsive system is important in terms of task performance and presence in VR environments in general [Meehan et al. 2003]. In particular for HMDs a high frame rate is also important. We thus want to separate the requirements regarding latency induced by the display technology (HMD or projection-based) from the requirements regarding a low-latent update of visual feedback, e.g., of a figure animated via motion capturing. In our description we are focusing on the latter.

For *feedback on one's own motion* (R1) in a virtual mirror, latency-induced effects could be reduced since humans can use motor prediction to adapt to delayed sensory feedback [Keetels and Vroomen 2012; Rohde and Ernst 2012; Heron et al. 2009]. Still, having more complex feedback strategies and an augmented virtual mirror in mind, low latency will become even more important for precise presentation of feedback (e.g., an avatar pointing at erroneous parts of the user's body during movement execution).

However, no fixed rules concerning the maximum allowed level of latency in VR motor learning applications exist. Literature suggests values of 150 ms for controlling characters in computer games, since higher latencies are already directly noticeable for untrained users and affect players in several ways [Jörg et al. 2012]. Meehan et al. [2003] showed that decreasing the latency from 90 ms to 50 ms already affects presence in virtual environments. Mackenzie and Ware [1993] used Fitt's tapping task to investigate the influence of latency on performance: They found that the performance of participants is reduced when being exposed to a latency of 75 ms or higher. According to Ware and Balakrishnan [1994] even a latency of 70 ms already affects performance in a VR reaching task. In a non-VR tapping task Jota et al. [2013] found that performance improves only little using latencies below 25 ms. Even for latencies below 50 ms, only a very slight improvement was measured. Improvements in latencies below 40 ms were not even noticed by most untrained participants.

In conclusion, it seems to be desirable to reach the lowest possible latency. An optimal corridor of latencies for visual feedback appears to be between 40 ms and 70 ms, depending on the specific application. But since there is, to our knowledge, no study proving guidelines for latency in immersive full-body motor learning, these values can only be inferred from related systems.

R3: Minimal level of disturbance

To guarantee a natural and intuitive training, the user should be able to move freely, at least regarding the movements that are relevant for the motions to be trained. Thus the hardware attached to the user has to be as unobtrusive as possible, since otherwise the user would not be able to use her full range of motion. For instance, the use of long and stiff wires as well as heavy components should be prevented if possible: Participants should perform the motor actions as they would in a real training scenario. Besides issues of naturalness of movements, obtrusive hardware could also make the optimal perception of the virtual environment more difficult [Witmer and Singer 1998]. Therefore motion capturing system and VR environment have to be chosen to offer a reasonable compromise between tracking precision, immersion, and obtrusiveness.

R4: Robust tracking

Many typical sports exercises include movements during which parts of the body are occluded for outside-in tracking systems. The motion capture system has to be as robust as possible against such kind of occlusions, where single or multiple markers might get lost. If the tracking is not robust enough, it might require a re-calibration of the human that is to be tracked. Thus, the training has to be interrupted and cannot be continued until the re-calibration is performed. The training is severely affected by such a re-calibration procedure to re-align tracking: If this happens, the naturalness of the application, as for instance demanded by Witmer and Singer [1998], would be significantly reduced.

From all requirements, low latency (R2) is most crucial. We therefore argue that the latencies of a VR environment for motor learning should be reported whenever presenting results of studies conducted in such a setup. This is important to exclude high latency as a potential side effect in the conducted experiments. More generally, when the exact specifications of an environment are known, the results of future experiments become better comparable as well as more reproducible. That is why we lay a special focus on latency measurement in this paper.

3 Related Approaches

This section gives a short overview of state-of-the-art approaches to motor learning systems in VR with respect to requirements developed above. In the following, these are referenced as R1–R4.

Smeddinck et al. [2014] present a training system that covers a large range of human movements. The system aims at improving motor performance for Parkinson's disease patients. Participants can monitor their own motion visualized through a coarsely rendered skeleton. Furthermore, the movement of the instructor can also be monitored, depending on experimental condition. The authors evaluate the effect of different abstractions of instruction presentations on motor performance. A Microsoft Kinect camera was used for motion capturing. The authors fulfill R1 (feedback on one's own motion), but did not provide any information on system latency (R2). However, given the latencies of the Kinect sensor, they can be expected to be well above 100 ms. Requirement R3 can be considered fulfilled as no hardware has to be attached to the user for Kinect-based motion tracking. The overall tracking robustness can be assumed to be sufficient for the task of rehabilitation for Parkinson's disease patients and the employed set of simple movements. However, using a Kinect camera might not be fast and robust enough for more complex motions (R4).

A yoga training game with a special focus on visually impaired people is presented by Rector et al. [2013]. They focus on spoken feedback to help trainees to reach a desired yoga posture. To get information about the performed movement, they also employ a Kinect camera. As the system targets visually impaired people, requirement R1, which demands for feedback on one's own motion can be seen as fulfilled via the provided spoken feedback. Indeed, the authors do not give any information on the system's latency (R2), which might be important to counter-steer over- and under-shooting movements caused by a high latency. For example, the system could state "Lean forward" based on a delayed measurement, although the user already exceeded the desired angle. Yet, yoga movements are typically rather slow, such that a high latency might only slightly influence the given task. Requirement R3, which requires a minimal level of disturbance, is fulfilled due to the marker-less Kinect tracking. Concerning the robustness of the tracking for the desired type of motion (R4), no information is given. It can be assumed that the authors chose postures that are easy to track with the Kinect camera and do not require too many changes in user orientation or self-occlusions of body-parts.

A highly specialized training system for rowing in VR is presented by Sigrist et al. [2014]. The user is placed in a modified boat, surrounded by projection walls. An extended version of the rowing blade is visualized and superimposed by the optimal blade position. Furthermore, the authors employ auditory feedback, which consists of a sonified oar blade and a sound which is played when the blade enters the virtual water. Haptic feedback is applied via resistance torques against the user's movement as soon as the user's blade moves away from the target position. Virtual self-monitoring (R1) is only possible via observing the virtual oar blade. Concerning latency and frame rate (R2), no information on the overall latency is given. Only the update rate of the projectors (> 30 Hz), movement sonification (30 Hz), and the frequency of the haptic device (1000 Hz) is described. The Unity engine is used to render the virtual ocean and the motion of the oar blade. Requirement R3 is satisfied as no additional hardware except from headphones has to be attached to the user and he/she is located inside a real boat. The tracking can be assumed to be sufficiently robust (R4), since the tracking task is not very complex.

Covaci et al. [2014] present a training system that aims at high-precision tasks such as the basketball free throw. The system is located in a CAVE environment, hence the ball has to be attached to a special construction to prevent the walls from damage. The ball and the user are tracked by a Vicon MX motion capture system. Directly after throwing the ball, the system calculates the trajectory of the ball and visualizes the throw. The users can monitor their own motion (R1) either in first- or in third-person perspective. The third-person perspective can also be overlaid with the correct trajectory of the ball. The system's shutter glasses run at 30 Hz per eye, the motion capture system has a frequency of 120 Hz. Information on the system's latency is not stated (R2). In a user study, the authors showed that the overall latency did not disturb the users. Requirement R3 (minimal disturbance) was evaluated via questionnaires: The interaction was stated as natural by participants, such that R3 can be considered fulfilled. The tracking is described as being robust (R4) and the calculation of the ball trajectory leads to correct results in 87.5 % of 500 trials.

To summarize, many different approaches towards VR motor learning exist. However, information on end-to-end latency is only rarely given. Hence results are difficult to compare, e.g., concerning the achieved levels of performance, and it is difficult to exactly replicate experiments. Furthermore, some systems use sensors unable to provide a robust tracking for a broad set of possible motor actions. To the best of our knowledge, no approach published until now aims at providing a general, highly controlled, efficient training environment that satisfies the above mentioned requirements and provides information on end-to-end latency. This work tries to fill this gap via description, discussion, and evaluation of state-of-the-art techniques, leading to an exemplary realization of a system that satisfies the stated requirements. Furthermore, we provide information on the system's end-to-end latency, which enables replication, comparison, and assessment of future experiments to be performed in this particular VR system.

4 Realization of Low-Latency Environment

This section describes our hardware setup, provides an assessment of state-of-the-art techniques for building a low-latency VR environment for motor learning, and finally presents our design choices and developments for this particular task. Figure 2 depicts the architecture of our system. It consists of three major parts: (i) display technology, (ii) render engine and (iii) motion capturing system / motion preprocessing.

To display our virtual world, we decided to use a CAVE environment. This ensures a *minimal level of disturbance* (R3), since the equipment attached to the user is limited to a pair of tracked 3D glasses. These glasses are usually much lighter and smaller than a full-sized HMD and there are no cables attached to the user. Moreover, the user is still able to see her own physical body and thus gets *feedback on her own motion* (R1) without any additional equipment. The still slightly narrow field of view of available HMDs impedes self monitoring by looking at one's own (virtually rendered) body: The user has to make larger head movements, especially when looking down along one's own body, which then may interfere with the training goals. In particular training situations, in which head and neck orientation and/or movements are essential, the additional weight imposed by the HMD also influences the trainee's posture compared to the optimal natural posture.

Our two-sided CAVE (L-Shape, 3 m $\times$ 2.3 m for each side) has a resolution of 2100 $\times$ 1600 pixels per side. Each side is driven by two projectors with INFITEC filters to enable passive stereoscopic vision by utilizing wavelength division. Both walls (floor and front) use back-projections. The four projectors are driven by a single computer (2 Intel Xeon CPU E5-2609 @2.4 GHz, 16 GB Ram, 2 Nvidia Quadro K5000 GPUs).

Our virtual world consists of the following components: a virtual fitness room with a virtual mirror mounted on the front wall. The user is placed in front of this mirror and her motions are mapped onto a generic avatar visible in the mirror. This effectively generates a virtual reflection of the user's motions, which further enhances the fulfillment of the requirement for *feedback on one's own motion* (R1). The user's motions are captured by an optical motion tracking system mounted at the top and the sides of the CAVE. Motion data is streamed into the Motion Preprocessor, which prepares the data for its use in the render engine and additional software packages for further analysis and feedback generation. The render engine then visualizes the scene while adapting the camera perspective(s) according to the user's head position/orientation and animates the virtual character in the mirror using the full-body tracking data.

In order to evaluate and compare the overall end-to-end latency of different rendering and tracking approaches, we adopted and extended a well-established latency measurement approach [Liang et al. 1991; Steed 2008; Friston and Steed 2014]: Typically, a pendulum is placed inside the tracking area, and the tracking data is visualized on a display behind the pendulum. A high speed camera records both the swinging real pendulum and the virtual pendulum on the screen. Afterwards, the recording is analyzed by hand, and the time-offset between the real and virtual pendulum is the end-to-end latency of the overall system. The following individual system latencies add up to the total latency: tracking latency, network latency, rendering latency, and display latency. The simple and periodic movement of a pendulum allows the application of automatic evaluation techniques [Friston and Steed 2014]. However, in our case, we are not only interested in the end-to-end latency of a single marker tracked by the system, but in the latency induced by (more complex) full-body motion capture. Hence we replace the pendulum by a human standing in the center of the CAVE, who is fully tracked and instructed to move one arm up and down. The tracked motions are mapped onto the virtual character, which is rendered on the front screen (see Figure 3). The scene is again recorded by a high-speed camera (170 Hz), and the video is analyzed by hand (see supplementary video). To reduce errors due to manual labelling, we average latency results over 30 trials.

In the following we first discuss our rendering solution, before presenting the full-body motion capturing approach.

4.1 Real-Time Rendering

Stereoscopic visualization in a CAVE requires to render two images (left/right eye) for each projection wall (floor and front in our case). Thus, the rendering framework must be capable of rendering multiple views per frame while still keeping up to the stated requirements: We satisfy requirement R1 (*feedback on one's own motion*) by visualizing the movements of the participant through a virtual character in a virtual mirror. Requirement R2 (*low latency and high frame rate*) then mainly depends on the chosen hardware and software solution for real-time multi-view rendering.

We first assess and evaluate several rendering techniques and check how well they fit our requirements, before presenting the rendering approach chosen for our low-latency VR system.

4.1.1 Evaluation of Existing Approaches

In terms of hardware, there are basically two solutions for stereoscopic rendering in a CAVE environment: The first is a render cluster, typically using one render node per projection wall or per view. The second possibility is to use a single computer with multiple GPUs (multi-pipe rendering). We implemented both approaches

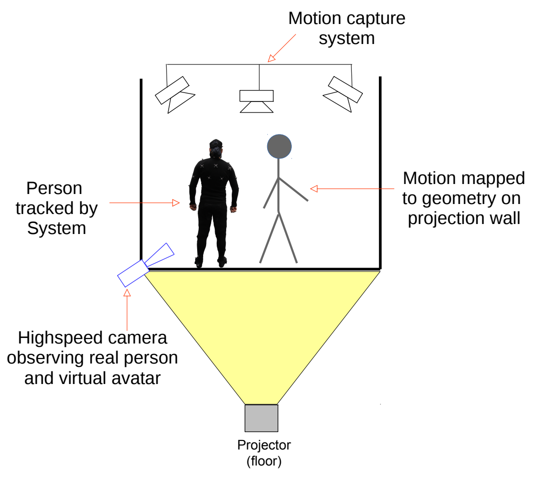
Motion capture system
Person tracked by System
Motion mapped to geometry on projection wall
Highspeed camera observing real person and virtual avatar
Projector (floor)

Figure 3: *Overview on latency measurement: The person inside the CAVE is equipped with a motion capture suit. Motion is directly mapped on the virtual character. A high-speed camera records both: real person and virtual character. Furthermore, it adds a time-stamp to the video. Later on, the number of milliseconds between the real person reaching a turning point and the virtual character reaching the turning point is determined.*

and compared them with respect to frame rate and latency. To this end, we set up two test cases:

1. Distributed rendering on a minimalist render cluster of two nodes with one GPU each, where each machine is responsible for driving one projection wall.

2. Multi-pipe rendering on a single computer, which is equipped with two GPUs for driving the two screens.

The cluster nodes and the single computer had identical system specifications (2 Intel Xeon 2.4 GHz, 16 GB Ram), with either one (cluster) or two (multi-pipe) Nvidia Quadro K5000 GPUs. The cluster nodes were connected by a fast 10 Gigabit Ethernet network.

We analyzed how much latency the network communication of even our minimalist render cluster introduces by employing the full-body latency measurement described in the previous section. To this end, we use an OptiTrack Prime13W system (240 Hz) for motion tracking (see Section 4.2) and map the motion onto a minimalist stick figure. Rendering is done using InstantReality[1], which is based on the distributed rendering framework OpenSG[2].

The scene was rendered at about 260 fps for the cluster setup and 280 fps for the multi-pipe setup. Over a range of 30 full-body latency measurements we determined a mean latency of 50 ms for the render cluster and 41 ms for the multi-pipe setup, with standard deviations of 12 ms and 10 ms, respectively. These results demonstrate that even a minimal cluster consisting of only two render nodes can already lead to an increased latency. This suggests that in terms of latency a multi-pipe framework on a single machine should be preferred over a render cluster, at least when the number of projection screens/views allows for a multi-pipe approach (easily for up to four walls). Additional reasons for the single-machine solution are easy maintenance, easier implementation, and less expensive hardware setup.

[1]`http://www.instantreality.org`
[2]`http://www.opensg.org`

In terms of software frameworks, stereoscopic multi-view rendering is a mature and well-established topic, and consequently numerous software solutions are available for this task. Full-featured VR frameworks, like for instance InstantReality, which we employed for the cluster-vs-multi-pipe benchmark, seem to be the canonical first choice. However, being targeted at fast and easy prototyping of VR applications, the primary goal of InstantReality is not high-performance rendering. While it supports character animation, it does not exploit GPU-acceleration for the involved skinning computations. As a consequence, when tested on our high-quality "mirror character" of 135k triangles, performance dropped down to about 5 fps even for a single window. Since fast character animation is crucial to ensure *low latency and high frame rate* (R2), InstantReality could not be employed.

Fast character animation and high-quality rendering are (besides many other features) provided by game engines like Unity[3] or Unreal[4]. While these game engines have not been designed for multi-view rendering, there are extensions to use them in a CAVE. For instance, the commercial software MiddleVR[5] allows the use of the Unity engine for CAVE rendering by taking care of the data distribution and synchronization for distributed cluster rendering. Although being designed mainly as a cluster solution, MiddleVR also allows to set up a local cluster on a single computer, thereby effectively providing a multi-pipe rendering solution. This configuration will probably add less latency compared to a multi-machine rendering cluster. However, with Unity and MiddleVR being closed source, the data storage and data flow cannot be precisely controlled, which might be important for latency reduction. CaveUDK [Lugrin et al. 2012] is a middle-ware for using the open-source Unreal 3 engine in a CAVE and is based on a distributed rendering approach. They report rather high end-to-end latencies of 82 ms for pressing a button to trigger an event and 136 ms for user navigation, which rules out this rendering solution.

While several other high-level and low-level software packages exist, such as OpenScenegraph[6] or Equalizer [Eilemann et al. 2009], respectively, we eventually decided to develop our own slim rendering framework, since this gives us full control to exploit low-level hardware acceleration and parallelization. The architecture and features of our rendering engine are presented in the next section.

4.1.2 Realization

In order to minimize latency, we developed a single-computer multi-pipe approach for rendering the scene in the CAVE. High performance rendering requires to offload all expensive computations to the available GPUs. This is even more important for a multi-pipe approach, because only this way it is ensured that the rendering performance scales properly with the number of GPUs. In addition, data transfer costs have to be reduced to a minimum, which is important for the implementation of character animation.

We minimize computational cost by animating the virtual character using the very efficient and simple linear blend skinning [Jacobson et al. 2014], where vertices $\mathbf{x}_i$ are transformed using a weighted linear blending of the joints' transformation matrices $\mathbf{T}_j$:

$$\mathbf{x}_i' = \left(\sum_{j=0}^{n} w_{i,j} \mathbf{T}_j \right) \mathbf{x}_i, \qquad (1)$$

where the weights $w_{i,j}$ determine the influence of joint/bone j onto vertex i. Vertex normals $\mathbf{n}_i$ are transformed using a similar equa-

[3]`http://www.unity3d.com`
[4]`https://www.unrealengine.com`
[5]`http://www.middlevr.com`
[6]`http://www.openscenegraph.org`

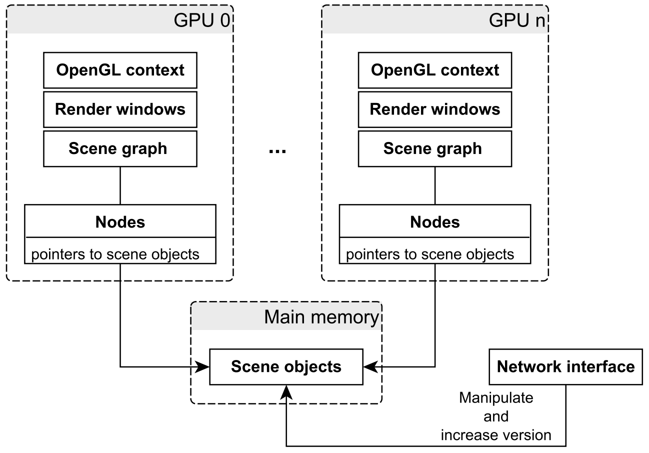

Figure 4: *Data distribution in our render engine: Each GPU has its own OpenGL context and scene graph. The scene graph nodes represent an object in the GPU memory and carry a pointer to their corresponding object in main memory. The nodes are kept up-to-date using a versioning approach.*

tion. Since this computation can be performed independently for each vertex, it can easily be mapped to the GPU. In this case, the rest-pose vertices $\mathbf{x}_i$ and normals $\mathbf{n}_i$, as well as the skinning weights $w_{i,j}$ remain constant in GPU memory, such that only the (rather small number of) joint transformations $\mathbf{T}_j$ have to be uploaded to GPU memory in each frame. In contrast, a CPU implementation (as done, e.g., in InstantReality) uses fewer computation cores and has to upload all new per-vertex data ($\mathbf{x}'_i$, $\mathbf{n}'_i$) to GPU memory. A performance comparison on our high-quality character (135k triangles) showed the GPU implementation to be about 50 times faster (2980 fps vs. 57 fps, single window, Intel Xeon E5-1620 3.6 GHz, Nvidia GeForce GTX 980).

The virtual mirror is implemented by first rendering the scene, including the animated character, from the mirrored perspective of the user. The content of the resulting framebuffer is then mapped as a texture onto the mirror geometry in the scene (see Figure 1 and Figure 6). To animate the character, we stream the pre-processed motion data from the Motion Preprocessor to the render engine using a network interface (compare Figure 2 and Figure 4). The data is received asynchronously and is then directly used to update the pose of the character. This updates the transformation matrices in CPU memory, and a simple version counter approach is used to keep the data on the GPUs up-to-date.

To ensure that the rendering scales well to the available GPUs, we employed the `WGL_GPU_affinity` extension to ensure that the OpenGL commands are sent only to the correct GPU. The OpenGL context is shared for all views associated to one GPU, such that data is stored only once on each GPU. In terms of lighting we use the simple Phong lighting model, and we apply shadow mapping to the character and other objects in the scene (see Figure 1 and Figure 6).

The resulting render engine provides all necessary features for our VR motor learning environment, while maintaining a slim software design and flexibility.

4.2 Motion Capture

Full-body motion capture is necessary to provide real-time augmented feedback on motor performance. In the following, we give an overview of state-of-the-art motion tracking approaches and assess them with respect to the aforementioned requirements.

4.2.1 Evaluation of Existing Approaches

To track full-body motion, the distinction between *outside-in* and *inside-out* approaches is important. For outside-in approaches, markers are attached to the human body and the actual capturing devices are placed at fixed positions outside the tracking area. The inside-out approach works the other way around, for instance by attaching inertial trackers to the user. Although the outside-in approach has to deal with occluded markers, it has the important advantage that no sensitive and/or heavy devices have to be attached to the user. Furthermore, outside-in approaches do not suffer from drift due to time-integration of sensor data, and they provide the exact location of the user. Since we want to avoid attaching disturbing hardware on the user (R3) we only take outside-in approaches into account in the following.

For these systems, the next distinction is between *marker-based* and *marker-less* approaches. Many commercially available systems exist for both approaches, such as the marker-based systems Vicon and OptiTrack, or the marker-less systems Microsoft Kinect and Organic Motion. The advantage of marker-less systems seems obvious: No hardware has to be attached to the participants, thereby reducing setup time significantly and minimizing user disturbance. However, marker-less systems often depend more on lighting conditions and a good view of the participant. We decided to focus on marker-based systems, since they provide more robust and more accurate tracking results in a CAVE environment. Nevertheless, we also analyzed the marker-less Kinect sensor, since this device can also operate in rather dark environments and is used in many related approaches towards motor performance training (e.g., [Smeddinck et al. 2014; Rector et al. 2013]).

For the marker-based systems, one can use *active* or *passive* markers. Passive markers simply reflect the infrared light emitted by the tracking cameras. The tracking system captures a set of markers, which then have to be consistently labeled. As soon as markers get lost and re-appear later on, the labeling step can produce errors. Active markers, as used in systems like PhaseSpace[7] avoid the labeling problem by emitting light at a unique frequency. The disadvantage of active markers is that they require more service and are more prone to get damaged during experiments. Furthermore, the marker suits are more difficult to clean. Additionally, active motion capture suits are often less comfortable to wear than suits for passive markers (R3).

Thus we decided to focus on outside-in tracking systems based on passive markers. We analyzed and compared the Vicon T20 system and the OptiTrack systems Flex 100 and Prime 13W. These systems require a motion capture suit with attached markers, or having the markers attached directly to the human skin. As motion capture suit any tightly fitting sports clothing can be used as long as it does not contain reflective materials. Thus these systems satisfy requirement R3. We evaluate the end-to-end latency and update rate (R2) of the different tracking systems using the latency measurement approach described in Section 4. In order to focus on the tracking latency, we only rendered a simple stick figure (at about 280 fps). Table 1 summarizes the resulting latencies for Vicon T20, OptiTrack Prime 13W, OptiTrack Flex 100, and Microsoft Kinect.

Concerning the robustness of the tracking (R4), the marker-based systems meet our demands: For most basic movements and exercises (e.g., squats, walking around, jumping), the user is tracked without the need for re-calibration or returning to the T-Pose during a session. In contrast, the tracking robustness for the Kinect camera was worse: Here, many kinds of exercises, e.g. squats, cannot be tracked reliably due to occluded body parts.

[7] `http://www.phasespace.com`

System	Price / Camera	Camera Resolution	Max. Frame-rate	Used Frame-rate	Latency	Std. Dev.
Vicon T20	20,000 EUR	1600 × 1280	500 Hz	100 Hz	54.9 ms	13.18 ms
				240 Hz	44.7 ms	10.6 ms
				500 Hz	38 ms	8.4 ms
OptiTrack Prime 13W	2,500 EUR	1280 × 1024	240 Hz	100 Hz	59.7 ms	12.3 ms
				240 Hz	41 ms	9.9 ms
OptiTrack Flex 100	600 EUR	640 × 480	100 Hz	100 Hz	65.5 ms	21 ms
Microsoft Kinect 2	150 EUR	512 × 424	30 Hz	30 Hz	98.8 ms	19.17 ms

Table 1: *Comparison of end-to-end latencies of the different motion capturing systems (averaged over 30 measurements), also listing price per camera, camera resolution, as well as the maximum and the employed frame rates.*

Render Quality	Fps	Latency	Std. Dev.
Stick figure	690	36 ms	9 ms
Low resolution	114	54 ms	9 ms
Low resolution + Shadows	88	60 ms	10 ms
High resolution	86	62 ms	12 ms
High resolution + Shadows	62	81 ms	14 ms

Table 2: *Latency and performance values for different rendering qualities (mean value of 30 tries and standard deviation). Rendering a minimalist stick figure without a virtual environment, or rendering the full gym scene and the virtual mirror, but using either a low-resolution (20k triangles) or high-resolution (135k triangles) virtual character, with optional shadow mapping.*

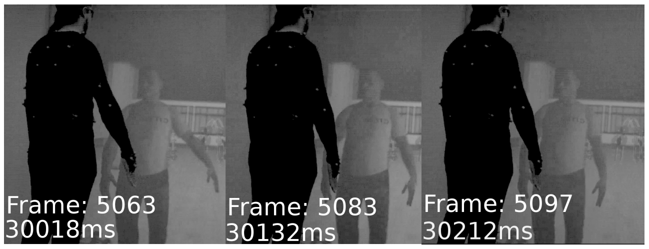

Figure 5: *Example frames from one of the latency test videos (highest quality character with shadows). The left image shows the user's arm approaching the lowest point. The image in the middle shows the turning point of the real arm, the picture on the right shows the turning point of the virtual arm, while the real arm already moves upwards.*

4.2.2 Realization

Based on the benchmark results shown in Table 1, we decided to use an OptiTrack Prime 13W system with 10 cameras. This marker-based solution is a good compromise as long as there is no marker-less option of similar performance and robustness. The Microsoft Kinect was excluded due to its high latency (R2) and problems in dealing with occluded body parts (R4). The Vicon cameras' advantage in terms of temporal and spatial resolution did not justify the much higher price for our field of application. We decided to use the Prime 13W system instead of the Flex 100 cameras because of the wider field of view (82° vs. 58°) and the higher temporal and spatial resolution.

The cameras are arranged in a way that allows an almost failure-free tracking. Participants are equipped with a marker suit with 44 markers for accurate skeleton tracking. This layout is based on the 41-marker layout specified by OptiTrack, which extends their basic 37-marker layout by two additional markers on each foot. These increase the robustness of foot tracking and allow us to get information on metatarsal rotation. We further extend this layout by three markers on the back in order to capture the bending of the spine in more detail. With the final setup, we are able to obtain transformations for 23 joints. The system provides joint positions and rotations, which are used to animate the virtual mirror character, for feature extraction (e.g., calculation of movement direction, speed, acceleration), and for motion analysis.

5 Benchmark

To evaluate the influence of rendering options on latency, we evaluated different quality levels to find the best trade-off between quality and performance. The same measurement procedure as described in Section 4 was used for 30 trials, and we report mean latency values and standard deviations. The virtual scene used for the tests consists of a virtual fitness studio (about 100 k triangles) including the virtual mirror (see Figure 6).

The results are listed in Table 2. For our high-resolution character (135 k triangles), we observed a latency of 81 ms at 62 fps when using shadow mapping. Without shadows, a latency of 62 ms at 86 fps was measured. Using the low-resolution character (20 k triangles) reduces the latency to 60 ms at 88 fps (with shadows) and 54 ms at 114 fps (without shadows). As a baseline test, we also rendered a simple stick figure without the surrounding fitness studio, which resulted in a latency of 36 ms at 690 fps. We conducted an additional test which consists of a single marker attached to a pendulum instead of a tracked human. The pendulum was visualized as a box inside the CAVE. Here, we observed a latency of 32 ms (SD=9 ms).

Our end-to-end latency consists of the individual latencies of the cameras (approx. 4 ms according to manufacturer), of the tracking software, the motion preprocessing (approx. 2 ms), the network communication (approx. 1 ms), as well as rendering, synchronization, and display hardware (approx. 19 ms according to manufacturer). Figure 5 shows exemplary frames of the recording filmed by the high-speed camera, showing the experiment using the high resolution character and real-time shadows.

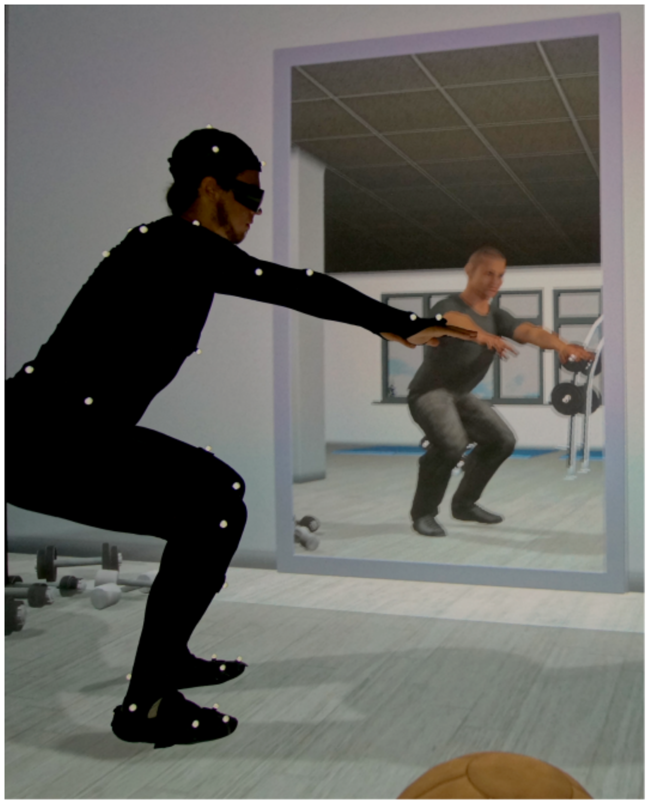

Figure 6: *The visual quality achieved in the final system, including artificial shadows for the trainee.*

In a pilot study we examined whether users have the feeling of being able to control the virtual character and whether the system induces simulator sickness (e.g., due to latency effects). For this study we rendered the high-resolution character including shadow mapping. 23 participants (15 female; age M=26.17, SD=8.94) interacted for 5–6 minutes with our system: They had to perform squats while getting simple textual feedback after having performed a squat. The degree of perceived control was measured using a 7-point Likert scale ranging from 0 (no control) to 6 (highest level of control). The results were quite satisfying (M=5, SD=1). Furthermore, no increase of simulator sickness, using the Simulator Sickness questionnaire by Kennedy et al. [Kennedy et al. 1993], was detected due to the experiment. For this experiment, we also evaluated the robustness of our system in terms of full-body motion capture: Only one single time, too many markers were occluded which required a re-calibration of the participant. In all other trials, temporary loss of markers was compensated by the system.

Figure 6 shows a photograph of a user reflected in the virtual mirror inside our environment. Please also see the accompanying video, which shows a user interacting with the system using the low-resolution character. The second part gives a glance on the latency measurement procedure recorded using a high resolution character with shadows.

6 Conclusion and Future Work

In this paper we developed and motivated requirements for VR motor learning. We examined state-of-the-art techniques and technologies for motion capturing and rendering with respect to these requirements, and propose a low-latency environment based on the most promising components or approaches. In terms of rendering, a single-PC multi-pipe approach was shown to achieve a lower latency than even a minimal render cluster using two nodes. Our slim custom-designed render engine maps all expensive computations to the GPUs and parallelizes well. For full-body motion capture, we decided to use the marker-based outside-in OptiTrack system. The resulting system provides a virtual environment with a mirror and a high quality character, and it can serve as a solid base for further developments and experiments in VR motor learning.

Using the 20k-triangle character, our system meets the stated requirements: The user is able to monitor his own motion in the virtual mirror (R1). The overall latency of the system is at around 60 ms, which is comparable or better then related systems (R2). The graphics engine runs at 88 fps, feeding four channels with 2100×1600 pixels each, which is sufficient to perceive smooth images. Requirement R3 is also satisfied as users only have to wear passive stereo glasses and tight clothing with attached markers. Of course marker-less motion tracking would be the ideal solution, but to our experience the available solutions are not fast or robust enough in a CAVE environment. Requirement R4 can also be considered as satisfied, as shown in the accompanying video.

This paper gives guidelines on how to develop a VR environment usable for motor learning experiments. Inherent variables of our system as well as possible alternative approaches have been evaluated and compared. This information should support reproducibility and increase comparability of experiments.

Our proposed system lacks portability, since the display technology as well as the motion tracking system are fixed installations. A portable system could be achieved by using components like a commodity depth sensor (e.g., Kinect) or inertial trackers for motion tracking and a HMD for visualization. However, any configuration of that sort will have the problems presented in this paper. Still, developing a portable system for motor learning that can be used at home or in a small clinic is an interesting challenge. It is then to be evaluated how the more obtrusive display hardware (HMDs) influences participants' performance of motor actions and their ability of motor learning.

In future work we will also conduct experiments to quantify the effect of reduced/increased/disturbed visual quality, latency, and tracking robustness on user experience and task performance in the field of motor learning. In particular, the influence of various levels of latency will be analyzed in order to identify an upper bound on latency that still allows for effective motor learning. Further steps will go into the direction of development and evaluation of different feedback strategies by augmenting the virtual mirror. Currently, we are using the same generic male character for all users. An important next step is to automatically create virtual avatars for the users by body scanning, which would then result in a more realistic virtual mirror and achieve a stronger identification with the avatar.

Acknowledgements

This research was supported by the Cluster of Excellence Cognitive Interaction Technology CITEC (EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG).

References

BIERBAUM, A. D. 2000. *VR Juggler: A Virtual Platform for Virtual Reality Application Development*. PhD thesis, Iowa State University.

CHUA, P. T., CRIVELLA, R., DALY, B., HU, N., SCHAAF, R., VENTURA, D., CAMILL, T., HODGINS, J., AND PAUSCH, R. 2003. Training for physical tasks in virtual environments: Tai Chi. In *Proc. of IEEE Virtual Reality*, 87–94.

COVACI, A., OLIVIER, A.-H., AND MULTON, F. 2014. Third person view and guidance for more natural motor behaviour in immersive basketball playing. In *Proc. of ACM Symposium on Virtual Reality Software and Technology*, 55–64.

EILEMANN, S., MAKHINYA, M., AND PAJAROLA, R. 2009. Equalizer: A scalable parallel rendering framework. *IEEE Transactions on Visualization and Computer Graphics 15*, 3, 436–452.

FRISTON, S., AND STEED, A. 2014. Measuring latency in virtual environments. *IEEE Transactions on Visualization and Computer Graphics 20*, 4, 616–625.

GUTWIN, C. 2002. The effects of network delays on group work in real-time groupware. In *Proc. of European Conference on Computer Supported Cooperative Work*, 299–318.

HÄMÄLÄINEN, P. 2004. Interactive video mirrors for sports training. In *Proc. of the third Nordic conference on Human-computer interaction*, ACM, 199–202.

HERON, J., HANSON, J. V., AND WHITAKER, D. 2009. Effect before cause: supramodal recalibration of sensorimotor timing. *PLoS ONE 4*, 11.

JACOBSON, A., DENG, Z., KAVAN, L., AND LEWIS, J. 2014. Skinning: Real-time shape deformation. In *ACM SIGGRAPH Course Notes*.

JÖRG, S., NORMOYLE, A., AND SAFONOVA, A. 2012. How responsiveness affects players' perception in digital games. In *Proc. of ACM Symposium on Applied Perception*, 33–38.

JOTA, R., NG, A., DIETZ, P., AND WIGDOR, D. 2013. How fast is fast enough? a study of the effects of latency in direct-touch pointing tasks. In *Proc. of ACM SIGCHI Conference on Human Factors in Computing Systems*, 2291–2300.

KEETELS, M., AND VROOMEN, J. 2012. Exposure to delayed visual feedback of the hand changes motor-sensory synchrony perception. *Experimental brain research 219*, 4, 431–440.

KENNEDY, R. S., LANE, N. E., BERBAUM, K. S., AND LILIENTHAL, M. G. 1993. Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. *The international journal of aviation psychology 3*, 3, 203–220.

LIANG, J., SHAW, C., AND GREEN, M. 1991. On temporal-spatial realism in the virtual reality environment. In *Proc. of ACM symposium on User interface software and technology*, 19–25.

LUGRIN, J.-L., CHARLES, F., CAVAZZA, M., LE RENARD, M., FREEMAN, J., AND LESSITER, J. 2012. CaveUDK: a VR game engine middleware. In *Proc. of ACM symposium on Virtual reality software and technology*, 137–144.

MACKENZIE, I. S., AND WARE, C. 1993. Lag as a determinant of human performance in interactive systems. In *Proc. of the ACM INTERACT'93 and CHI'93 conference on Human factors in computing systems*, 488–493.

MANIA, K., ADELSTEIN, B. D., ELLIS, S. R., AND HILL, M. I. 2004. Perceptual sensitivity to head tracking latency in virtual environments with varying degrees of scene complexity. In *Proc. of ACM Symposium on Applied perception in graphics and visualization*, 39–47.

MAUVE, M., VOGEL, J., HILT, V., AND EFFELSBERG, W. 2004. Local-lag and timewarp: providing consistency for replicated continuous applications. *IEEE Transactions on Multimedia 6*, 1, 47–57.

MEEHAN, M., RAZZAQUE, S., WHITTON, M. C., AND BROOKS JR, F. P. 2003. Effect of latency on presence in stressful virtual environments. In *Proc. of IEEE Virtual Reality*, 141–148.

PARK, K. S., AND KENYON, R. V. 1999. Effects of network characteristics on human performance in a collaborative virtual environment. In *Proc. of IEEE Virtual Reality*, 104–111.

RECTOR, K., BENNETT, C. L., AND KIENTZ, J. A. 2013. Eyes-free yoga: an exergame using depth cameras for blind & low vision exercise. In *Proc. of International ACM SIGACCESS Conference on Computers and Accessibility*, 12:1–12:8.

RIZZO, A., AND KIM, G. 2005. A SWOT analysis of the field of virtual reality rehabilitation and therapy. *Presence 14*, 2, 119–146.

ROBERTS, D. J., SHARKEY, P. M., AND SANDOZ, P. 1995. A real-time, predictive architecture for distributed virtual reality. In *Proc. of ACM SIGGRAPH Workshop on Simulation & Interaction in Virtual Environments*.

ROHDE, M., AND ERNST, M. O. 2012. To lead and to lag–forward and backward recalibration of perceived visuo-motor simultaneity. *Frontiers in psychology 3*.

SCHACK, T., BERTOLLO, M., KOESTER, D., MAYCOCK, J., AND ESSIG, K. 2014. *Technological advancements in sport psychology*. Routledge Companion to Sport and Exercise Psychology: Global perspectives and fundamental concepts. Routledge, 953–965.

SHNEIDERMAN, B. 1984. Response time and display rate in human performance with computers. *ACM Computing Surveys 16*, 3, 265–285.

SIGRIST, R., RAUTER, G., MARCHAL-CRESPO, L., RIENER, R., AND WOLF, P. 2014. Sonification and haptic feedback in addition to visual feedback enhances complex motor task learning. *Experimental brain research 233*, 1–17.

SMEDDINCK, J. D., VOGES, J., HERRLICH, M., AND MALAKA, R. 2014. Comparing modalities for kinesiatric exercise instruction. In *ACM CHI'14 Extended Abstracts on Human Factors in Computing Systems*, 2377–2382.

STEED, A. 2008. A simple method for estimating the latency of interactive, real-time graphics simulations. In *Proc. of ACM symposium on Virtual reality software and technology*, 123–129.

WARE, C., AND BALAKRISHNAN, R. 1994. Reaching for objects in VR displays: lag and frame rate. *ACM Transactions on Computer-Human Interaction 1*, 4, 331–356.

WITMER, B. G., AND SINGER, M. J. 1998. Measuring presence in virtual environments: A presence questionnaire. *Presence: Teleoperators and virtual environments 7*, 3, 225–240.