

Real-Time Body Tracking in Virtual Reality using a Vive Tracker

Polona Caserman · Augusto Garcia-Agundez · Robert Konrad · Stefan Göbel · Ralf Steinmetz

Received: date / Accepted: date

Abstract Due to recent improvements in virtual reality (VR) technology, the number of novel applications for entertainment, education, and rehabilitation, has increased. The primary goal of these applications is to enhance the sense of belief that the user is “present” in the virtual environment. By tracking the user’s skeleton in real-time, it is possible to synchronize the avatar’s motions with the user’s motions. Although current common devices implement body tracking to a certain degree, most approaches are limited by either high latency or insufficient accuracy. Due to the lack of positional and rotation data, the current VR applications typically do not represent the user’s motions. In this paper, we present an accurate, low-latency body tracking approach for VR-based applications using Vive Trackers. Using a HTC Vive headset and Vive Trackers we have been able to create an immersive VR experience, by animating the motions of the avatar as smoothly, rapidly and as accurately as possible. An evaluation showed our solution is capable of tracking both joint rotation and position with reasonable accuracy and a very low end-to-latency of 6.71 ± 0.80 ms. Due to this merely imperceptible delay and precise tracking, our solution can show the movements of the user in real-time in order to create deeper immersion.

Keywords

1 Introduction

Virtual Reality (VR) can be experienced wearing novel Head-Mounted Displays (HMDs). In the last few years, there has been a rapid improvement in VR technology, increasing the availability of HMDs to consumers [7][13]. The most advanced HMDs, like Oculus Rift, HTC Vive, and PlayStation VR, already have a high-definition resolution, a wide field-of-view, and a high refresh rate. Furthermore, novel VR systems are capable of positional and rotational tracking of the HMD as well as additional VR devices. Tracking systems in VR can provide new possibilities for a more comfortable, immersive experience and gameplay [16]. Oculus Rift, for example, enables tracking through an embedded infrared system [12]. HTC Vive has similar technical specifications to Oculus Rift Consumer Version 1. Both HMDs provide high-definition resolution of 2160×1200 pixels, split between each eye and can maintain a frame rate of up to 90 Hz [12] [39]. But the special feature of the HTC Vive is the Vive Tracker, which allows the developers to bring any real-world object into the virtual environment, e.g. by simply attaching it to sporting equipment like a baseball bat, a golf club or a weapon¹. The position and orientation of this device are then tracked by two “Lighthouse” station, based on infrared signals. Each “Lighthouse” stations consist of infrared LEDs, flashing at regular intervals and signaling the start of a cycle [9]. Two little motors project laser beams across the room, one spinning horizontally and the other vertically. The sensors on the VR devices then detect these lasers and can determine its position based on the order its sensors receive the laser sweeps.

P. Caserman, A. Garcia-Agundez, R. Konrad, S. Göbel and R. Steinmetz
Multimedia Communications Lab, Technische Universität Darmstadt, 64283 Darmstadt, Germany E-mail: firstname.lastname@kom.tu-darmstadt.de

¹ Vive Tracker: <https://www.vive.com/us/vive-tracker/>, last visited on April 3rd, 2018

Due to this HMD development, the number of novel and innovative games for rehabilitation, training and exercise activities has greatly increased, e.g., [3] [8] [19] [38]. The primary goal of these applications is to give the user an illusion of presence of “being there” in the VR [11] [16]. To create an immersive experience, the connection between the user and the VR, as well as between the player and the avatar, has to be established. Immersive VR can be used to induce ownership over a virtual body that substitutes the real body, as seen from the first-person view [2] [31]. The person exploring VR would then be able to look down and perceive the avatar as her/his own body. Hence, by synchronizing the body movement of the user and their avatar, a positive effect on the cognitive ability of the user as well as the feeling of agency over the avatar can be achieved [8] [31]. Especially multiplayer VR games have the requirement of synchronizing the whole body in real-time in order to create deeper immersion [20]. Recent commercial approaches use infrared VR devices to track the full-body movements in real-time². Furthermore, other developers provided Unity 3D game engine based asset packages to achieve full-body tracking, e.g. Vive IK Demo³ and Final IK⁴. However, to the best of our knowledge, there is no research on evaluating these existing commercial kinematic solutions, particularly regarding its accuracy and latency.

Tracking and representing body movements, regardless of the user orientation, can be challenging. Although common devices implement body tracking to a certain degree, most approaches are limited by either high latency or insufficient accuracy. Due to this lack of data about a user’s position and orientation in the world, the current VR games typically do not track or represent the body of the user [12].

In this paper, we implement a low-latency body tracking approach for immersive VR-based applications. By using only infrared VR Controllers, e.g. Vive Tracker it is possible to transfer full-body player’s motions onto a virtual avatar. In addition, end-to-end latency is measured. The main research contributions of our work are the following:

- We develop a latency measurement tool in order to evaluate the total delay of the proposed method. Using this tool, we want to show that the end-to-end latency of the developed system stays below 20 ms

since this satisfies the requirements of the VR experiences.

- In contrast to systems using motion capture suits, we only use a small number of sensors to avoid high initial costs as well as complex setup. We do not explicitly track each body joint. In our research, we only track the position and orientation of the end-effectors (e.g. hands). We then solve the Inverse Kinematics (IK) problem to determine the angle of other joints in order to enable full-body tracking. This satisfies the desire to reduce the amount of sensor.
- We use infrared VR Controller, which do not suffer from occlusion and high latency, such as the Kinect sensor.
- Our system is entirely based on low-cost hardware and low-level game API. We can easily access the lowest level in order to achieve maximal performance. Additionally, because our body tracking solution should be available e.g. for researchers to create immersive VR experiences, we include source code, which can be accessed on GitHub⁵.

The rest of this paper is structured as follows. Section 2 provides the related work. In Section 3, the approach of body tracking using infrared sensors and a latency measurement tool are described. An experimental evaluation of the deployed system is given in Section 4. A discussion of the results and a conclusion follow in Section 5.

2 Related Work

2.1 Full-Body Tracking

Many recent research publications add a growing base of evidence to support the use of VR and full-body tracking. Examples recently showed the benefits of having a full-body avatar in a virtual environment by demonstrating the important role of realistic looking virtual humans [27]. Furthermore, owning a virtual body and perceiving it from the first-person perspective is also important when performing reaching tasks in VR [42].

The most popular game systems, capable of motion-sensing such as Microsoft Kinect⁶ or Nintendo Wii⁷ already provided a significant evidence that exergames are entertaining and motivating [25]. However, both

² IKinema Orion: <https://ikinema.com/orion>, last visited on April 3rd, 2018

³ Vive IK Demo: https://github.com/JamesBear/vive_ik_demo, last visited on April 3rd, 2018

⁴ Final IK: <https://assetstore.unity.com/packages/tools/animation/final-ik-14290>, last visited on April 3rd, 2018

⁵ Body Tracking Demo: <https://github.com/CatCuddler/BodyTracking>, last visited on April 4th, 2018

⁶ Microsoft Kinect: <https://developer.microsoft.com/en-us/windows/kinect>, last visited on January 28th, 2018

⁷ Nintendo Wii: <https://www.nintendo.co.uk/Wii/Wii-94559.html>, last visited on January 28th, 2018

Kinect versions suffer from inconsistent tracking, jittering, and unreliable data [13]. Kinect V1 sensor is only accurate when tracking gross movements such as sit-to-stand, but is very poor for fine movements such as hand clapping, toe or finger tapping [15]. Due to the new technology, the Kinect V2 which is based on the time-of-flight principle is more accurate in detecting small movements and provides better tracking results. However, the latency of the new version still remains high, approximately at 170 ms when combined with the Oculus Rift [4]. An additional disadvantage of a single Kinect sensor is, that it can only poorly track the rotation of body parts and is incapable of tracking when the user stands sideways [13].

Although the Kinect suffers from occlusion, provides noise in skeleton tracking and has a high latency, it is the most popular device for body tracking. Due to its adequate accuracy and low cost, many researchers are using this technology to track the user's movements. Shum and Ho [37] investigated the major problems of Kinect and developed a framework for a best-matched posture from the captured motion. The proposed solution can overcome the problem of missing Degree of Freedom (DoF) due to occlusions and noises. Sra and Schmandt [38] used Kinect V2 devices to track objects and users. An Oculus Rift DK2 is used for tracking of the head rotation. Collingwoode-Williams et al. [8] used Kinect V1 to research the effect of limb and arm synchronization on body ownership in VR. In their study, the user wearing a HMD was able to see a gender-matched avatar in a virtual mirror, that moved its limbs synchronously with the user. Bolton et al. [3] developed a VR-based exergame, based on the game Paperboy, where the player is wearing a VR headset and driving a bicycle. The user movements of throwing newspapers into the neighborhood mailboxes were tracked by a Kinect camera. The arms were synchronized to create a high level of immersion. To overcome occlusion problems, other studies even use multiple Kinect devices to track a single user [10].

Unfortunately, the Kinect sensor in combination with VR is inaccurate and will eventually show a false avatar posture [41]. To provide a more accurate tracking, other recent studies have attempted to use a suit-based motion capture technology. These body-tracking suits have attached infrared LED markers which can be then detected by a high-speed camera. Peck et al. [31] developed a VR experience, whereby the motions were tracked by the OptiTrack⁸ system with 12 cameras. The movements could be reconstructed at 100 Hz and synchronized with the virtual avatar. The users wearing

the HMD could see their virtual body from the first-person perspective as well as a reflection in a virtual mirror. Likewise, Banakou et al. [2] used 34 cameras to track user's motions. Chan et al. [6] proposed a dance system using a similar optical motion capture system. The user, wearing the motion capture suit, can learn new dance movements by imitating the motions demonstrated by a virtual teacher and listening to the feedback. Since suit-based tracking technology is capable of real-time full-body tracking of multiple users, some authors developed VR multiplayer applications or games, e.g. creating a physical condition control for athletes and dancers [22]. In contrast to a single infrared camera, such as Kinect, a wearable motion capture suit is capable of a very accurate body tracking. However, it is very expensive and complicated to use. Using a suit with LED markers for tracking requires a setup area and multiple high-speed cameras. Due to the high initial cost and complex setup, such a motion capture system is in general not applicable for home-based usage.

For tracking full-body movements also Inertial Measurement Units (IMUs) can be attached to the user's body. Different commercial tracking systems, such as PrioVR⁹, Perception Neuron¹⁰ or Xsense¹¹ are based on IMU. Perception Neuron furthermore utilizes a special data glove with a Vive Tracker in order to track hand position as well as individual fingers. Tsai et al. [43] developed an own wearable sensor to determine the skeleton posture in real-time. Moreover, measurement values of the integrated sensors of a HMD can be used to recognize steps [5]. Applying this step detector, the researchers were able to synchronize the feet of the user while walking on a treadmill. The user can then look down and see her/his virtual body from the first-person perspective as she or he would in the real world. In another work, an IMU is attached to a bicycle to detect the steering and breaking information [28]. The player movements have been detected while the player was sitting on the bicycle with the feet on the pedals and hands on the handlebar.

2.2 Inverse Kinematics

Recent studies have attempted to use IK to determine a set of appropriate joint configurations based upon the desired end-effector position. IK approaches based on Jacobian are originally used in robotics in order to

⁸ OptiTrack system: <http://www.optitrack.com>, last visited on January 17th, 2018

⁹ PrioVR: <https://yostlabs.com/priovr/>, last visited on July 31st, 2018

¹⁰ Perception Neuron: <https://neuronmocap.com>, last visited on July 31st, 2018

¹¹ Xsense: <https://www.xsens.com/>, last visited on 31st July, 2018

control (industrial) manipulators and were already presented in the 80's [30] [29]. The IK problem, to provide a solution that satisfies the positional and orientational constraints of each specific joint, has been well studied. Kenwright [24] presented a realistic and robust method for solving nonlinear IK problems with angular limits using the Gauss-Seidel iterative method. The proposed method merely requires a small number of iterations and needs only a few milliseconds to compute the solution. Aristidou and Lasenby [1] proposed a novel heuristic method, combining forward and backward IK. Other recent studies improved IK solutions using a multivariate Gaussian distribution model, which precisely specifies the joint constraints of a kinematic skeleton by integrating bio-mechanical properties and physical capacity of a human [18]. Additionally, IK systems based on a probabilistic model of learned human poses were presented [17]. However, such a system can only produce the most likely pose satisfying the constraints. Other researchers try to improve tracking quality by taking advantage of neural networks to reconstruct the motions, such as walking, jogging, jumping, crouching and turning [20].

2.3 Latency

To improve the feeling of presence in the VR, merely tracking user movements in order to synchronize the movements with those in VR is not sufficient [8] [19]. Similarly, the total delay from the time a motion occurs, to the time the results of that motion are displayed, should be well considered. A high frame rate and low HMD latency must be ensured in order to create an immersive VR experience [12]. On the one hand, a high latency of the HMD can contribute to cybersickness symptoms of disorientation, headache, nausea, and dizziness [7] [40]. On the other hand, a big delay between a physical movement and an output image can decrease the user's sense of immersion [12] [14]. Especially in VR, the end-to-end latency should not be higher than 20 ms [32]. Kasahara et al. [22] also showed similar results. The researchers found that a high latency (> 30 ms) will break the sense of agency and body ownership. Therefore, when developing VR experiences, it is important to keep the end-to-end latency as low as possible.

Previous works have shown that one or more synchronized cameras can be used to measure the latency in an immersive virtual environment [14] [33]. By filming the tracked real object and the associated output of the virtual environment, the delay can be determined using image processing techniques.

3 Approach

To develop a reliable real-time body tracking system, that can be used in an immersive VR experience, the HTC Vive HMD and the Vive Tracker are used to track the movements of the user. By using two "Lighthouse" position-sensing base stations and the VR devices with a large number of infrared sensors, such a system suffers much less from occlusion than a single Kinect device. With this technology, we can develop a reliable full-body tracking system which is able to provide accurate user posture, regardless of the user orientation. Thereby, the technical requirements of using only a small number of sensors and avoiding of high costs as well as complex setup (see Section 1) are satisfied. With an accurate real-time body tracking solution and an efficient IK solver, the virtual character can be synchronized with the user. The person wearing a HMD is then able to view the virtual body from the first-person perspective.

3.1 Development of The Body Tracking System

In this section, the approach of the real-time body tracking system to determine user movements will be described. Because this system should be used to synchronize the virtual avatar with the body movements of the user, an articulated character model with a skeleton must be created. Then, by obtaining the positional and rotational data of the Vive Trackers that are strapped to hands and feet, the full-body motions of the user can be continuously tracked. Through the efficient implementation of the iterative method for solving the IK problem, a set of appropriate joint configurations in an articulated model based upon a desirable end-effector position can be determined in only a few iterations. Finally, the skeleton is animated according to the calculated positions and orientations of the bones. In the following, a detailed description of these individual steps will be given. The flow chart for the body-tracking system is presented in Figure 1.

3.1.1 Character Model

An articulated character with a skeleton was modeled with the MakeHuman¹² open source tool. A skeleton of a small number of bones was consciously chosen in order to easily define joint constraints to solve the IK problem and to animate the user's motions, which will

¹² MakeHuman: <http://www.makehuman.org>, last visited on February 3rd, 2018

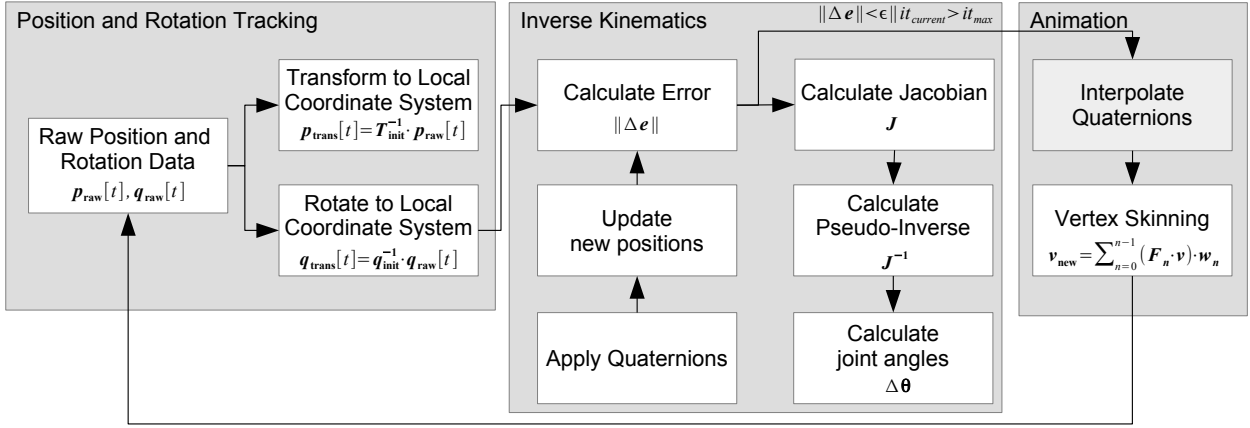


Fig. 1 Flow chart for the body-tracking system. After obtaining the position and rotation of the Vive Controller, we solve the IK problem in order to determine the appropriate joint configurations. In the last step, we animate the character, according to the skeleton.

be described later. To facilitate the transfer of the character model to the Kore¹³ framework, the Open Game Engine Exchange format (OpenGEX¹⁴) is used. Both, the OpenGEX format and the Kore framework are open source projects. OpenGEX exports skinned meshes (vertex data, skeleton, bind-pose transforms, bone influence weighting data) in a human-readable text-based file. Kore is a low-level game library and hardware abstraction framework, which is implemented in the C++ programming language. It provides the necessary functionality to develop games and multimedia applications with high performance.

A skeleton is defined as a tree structure of bone nodes, where each of these nodes is described by a 4×4 transformation matrix. The transformation matrix describes the bind-pose of a bone node, thus, the default pose of the character mesh before any bone transformation is applied. When the animation is applied, this matrix is used to calculate the new position and orientation. However, in order to prevent unnatural-looking poses while animating the character, we have to define a DoF for each joint. We have to restrict the possible rotations, i.e. rotations around the x , y , and z -axis. The generated skeleton gives us 56 DoF in total, as it can be seen in Table 1.

The constraints are defined by creating an axis vector $\mathbf{a} \in \mathbb{R}^{3 \times 1}$ for each joint and setting the angular limits for each of the axes. To prevent such an abnormal pose, like it can be seen in Figure 2, a constraint for the knee has to be specified, i.e. $\mathbf{a}_{knee} = [1 \ 0 \ 0]^T$ with angular limits $\min_{knee} = 0$ and $\max_{knee} = 2$. Hence, the knee can rotate only around the x -axis and the an-

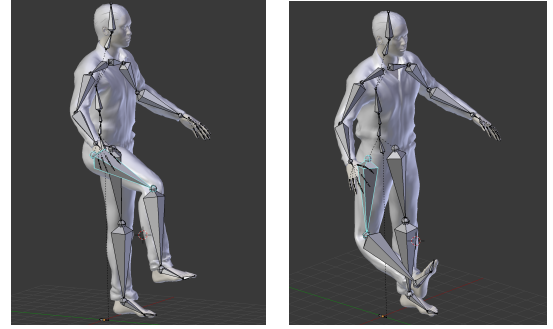


Fig. 2 Character foot reached the desired end position in both variants. However, only the left body posture is natural. On the right, an unrealistic pose is shown.

gle can be only in the range of 0 and 2 radians. When a joint rotates around multiple axes, consequently angular limits for each axis have to be specified.

3.1.2 Position and Rotation Tracking

The position and rotation tracking is the core task of the full body-tracking system to represent the movements of the user in the VR. For tracking Vive Trackers

Table 1 DoF of the articulated character model. The number in the brackets indicates how many of these joints exist in the skeleton.

Joint	DoF	Joint	DoF
Head (1)	3	Foot (2)	3
Neck (1)	3	Ball (2)	3
Pelvis (1)	3	Clavicle (2)	2
Spine (3)	3	Upper arm (2)	3
Thigh (2)	3	Lower arm (2)	2
Calf (2)	1	Hand (2)	2
Total:		56	

¹³ Kore: <https://github.com/Kode/Kore>, last visited on April 3rd, 2018

¹⁴ OpenGEX: <http://opengex.org>, last visited on February 21st, 2018

are strapped to hands and feet. The sensors can accurately track the yaw, pitch and roll movements as well as the spatial position.

The coordinate system of the avatar is attached on the floor and is a right-handed coordinate system, where the x -axis points to the left, the y -axis points backward and the z -axis points upwards. To locate the character so that the user wearing HMD can look down and see her/his virtual body, the character has to be transformed, rotated and scaled. First, the character is scaled by a $\mathbf{S}_{init} \in \mathbb{R}^{4 \times 4}$ matrix so that the eye height of the character corresponds to the height of the HMD. The sensor measurements of the HMD are provided in a head-fixed coordinate system, where the x -axis points to the right, the y -axis points upwards and the z -axis points backward. To calculate the scale factor, we can divide the current height of the user (y position of the HMD, $p_{y,hmd}$) by the character height (z position of the character head bone, $p_{z,head}$), i.e. $s = p_{y,hmd}/p_{z,head}$.

In addition, we have to rotate the character so that its orientation coincides with the orientation of the user. Let $\mathbf{q}_{init} \in \mathbb{R}^{4 \times 1}$ be the initial quaternion that rotates the character so that the virtual body looks in the same direction as the user. Quaternions are used because they are very simple, efficient and do not suffer from Gimbal lock [36]. However, because the local transformation of the character is calculated by applying the scale, rotation and lastly translation matrix, we have to convert the quaternion to a matrix. To include quaternion calculations in a regular, matrix-based transformation pipeline, we can represent the quaternion $q = [x \ y \ z \ w]^T$ as a matrix [36]:

$$\mathbf{R}_{init} = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2zw & 2xz + 2yw \\ 2xy + 2zw & 1 - 2x^2 - 2z^2 & 2yz - 2xw \\ 2xz - 2yw & 2yz + 2xw & 1 - 2x^2 - 2y^2 \end{bmatrix} \quad (1)$$

Finally, we translate the character to the position of the HMD, represented by a vector $\mathbf{p}_{hmd} \in \mathbb{R}^{3 \times 1}$. Let $\mathbf{T}_{init} \in \mathbb{R}^{4 \times 4}$ be the initial transformation matrix and is described as:

$$\mathbf{T}_{init} = \begin{bmatrix} 1 & 0 & 0 & p_{x,hmd} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & p_{z,hmd} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

Combining all three matrices, the coordinate system of the character is placed so that the user can look down and see her/his virtual body. Multiplying the raw positional vector $\mathbf{p}_{raw}[t] \in \mathbb{R}^{4 \times 1}$ of the Vive Tracker at time step t with the inverse transformation matrix $\mathbf{T}^\dagger$

will transform the sensor measurements to the character local coordinate system:

$$\mathbf{T}^\dagger = (\mathbf{T}_{init} \cdot \mathbf{R}_{init} \cdot \mathbf{S}_{init})^{-1}, \quad \mathbf{T}^\dagger \in \mathbb{R}^{4 \times 4} \quad (3)$$

$$\mathbf{p}_{trans}[t] = \mathbf{T}^\dagger \cdot \mathbf{p}_{raw}[t], \quad \mathbf{p}_{trans}[t] \in \mathbb{R}^{4 \times 1} \quad (4)$$

Similar, the orientation of the Vive Tracker $\mathbf{q}_{raw}[t] \in \mathbb{R}^{4 \times 1}$, has to be transformed as:

$$\mathbf{q}_{trans}[t] = \mathbf{q}_{init}^{-1} \cdot \mathbf{q}_{raw}[t], \quad \mathbf{q}_{trans}[t] \in \mathbb{R}^{4 \times 1} \quad (5)$$

3.1.3 Implementation of Inverse Kinematics with reduced Jacobian Matrix

To solve the IK problem we use the transformed positional vector and the quaternion (rotation) computed in the previous step. With the known desired position and orientation of the end-effector (e.g., a hand or a foot), the angle of each predecessor joint (e.g., an elbow or a knee) can be computed using an iterative, numerical method. An overview of the algorithm is as follows:

1. Calculate error between desired and actual position as well as rotation
2. Check for convergence
3. Calculate Jacobian
4. Calculate Pseudo-Inverse
5. Calculate joint angles for each bone joint
6. Apply quaternions to the transformation matrix
7. Update new positions

In each iteration, in the first step, an error between the desired and actual position $\Delta \mathbf{e}_{pos}$ as well as the desired and current rotation $\Delta \mathbf{e}_{rot}$ of the end-effector has to be calculated. Subsequently the error $\Delta \mathbf{e} = [\Delta \mathbf{e}_{pos}; \Delta \mathbf{e}_{rot}]$ is normalized. When checking for convergence in the second step, the error is compared with the maximum error threshold, i.e. $\|\Delta \mathbf{e}\| < \epsilon$. When the end-effector is close enough to the desired location or when there is no significant change between current and desired rotation, the algorithm will terminate. Because the end-effector may not be able to reach the desired position, we have to specify the maximal iteration number. This can happen when the desired position is out of range and therefore too far away to be reached. Otherwise, the Jacobian matrix $\mathbf{J}$ is calculated in the third step. The position and rotation values of the required axes can be obtained from a combined transformation matrix:

$${}^0\mathbf{C}_j = \begin{bmatrix} {}^0\mathbf{a}_{xj} & {}^0\mathbf{a}_{yj} & {}^0\mathbf{a}_{zj} & {}^0\mathbf{p}_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{4 \times 4}, \quad (6)$$

where ${}^0\mathbf{a}_j \in \mathbb{R}^{3 \times 1}$ represents the global rotation around the x , y and z -axis and ${}^0\mathbf{p}_j \in \mathbb{R}^{3 \times 1}$ represents the global position of the j^{th} joint with respect to the origin.

The Jacobian $\mathbf{J}$ is defined by the partial derivatives of the joint angles and the difference between the current position and the desired position of the end-effector. It can be determined by computing the cross product of the joint angle and the change in end-effector location:

$${}^0\mathbf{J}_{n,j} = \frac{\partial {}^0\mathbf{p}_n}{\partial \theta_j} = \begin{bmatrix} {}^0\mathbf{a}_j \times ({}^0\mathbf{p}_n - {}^0\mathbf{p}_j) \\ {}^0\mathbf{a}_j \end{bmatrix}, \quad {}^0\mathbf{J}_{n,j} \in \mathbb{R}^{6 \times 1} \quad (7)$$

where ${}^0\mathbf{p}_n \in \mathbb{R}^{3 \times 1}$ represents the current position of the end-effector, ${}^0\mathbf{a}_j \in \mathbb{R}^{3 \times 1}$ the rotation axis and ${}^0\mathbf{p}_j \in \mathbb{R}^{3 \times 1}$ the position vector of the j^{th} joint. In order to minimize the computational effort, we calculate for each end-effector only three partial derivatives. In other words, e.g. for a hand to reach the final position, we only determine the position and orientation of the three predecessor joints. The Jacobian $\mathbf{J}$ is then build as:

$${}^0\mathbf{J}_3 = [{}^0\mathbf{J}_{3,1} \quad {}^0\mathbf{J}_{3,2} \quad {}^0\mathbf{J}_{3,3}] \in \mathbb{R}^{6 \times 3}, \quad (8)$$

where the 0^{th} joint specifies the root node and the 3^{th} joint the end-effector. Thus, to manipulate the hand, we rotate the three predecessor joints, i.e. lower arm, upper arm, and clavicle.

In the fourth step, the pseudo-inverse of the Jacobian $\mathbf{J}$ has to be computed. Due to our adjustment of the Jacobian matrix, it will always have the same dimensionality:

$$\mathbf{J}_{\text{left}}^{-1} = \underbrace{(\mathbf{J}^T \mathbf{J})^{-1}}_{3 \times 3} \mathbf{J}^T. \quad (9)$$

The calculation of the left pseudo-inverse will lead to the determination of a smaller inverse matrix (3×3) and is therefore advantageous.

In the fifth step, the joint angles are calculated by multiplying the inverse Jacobian with the difference between desired and current position as well as rotation of the end-effector, as it can be seen in Equation 10:

$$\Delta \theta = \mathbf{J}^{-1} \cdot \Delta \mathbf{e}, \quad \Delta \theta \in \mathbb{R}^{3 \times 1} \quad (10)$$

In the sixth step, we can apply the new rotation to the joints:

$$\theta^{(k+1)} = \theta^{(k)} + \Delta \theta \quad (11)$$

Before the quaternion can be applied to the transformation matrix, the angular limits have to be ensured as described in Section 3.1.1. Otherwise, the character hand or foot will reach the desired position, however, the individual joints within the kinematic chain can

cause unnatural movements. The joint rotations are enforced through clamping between a lower bound (LB) and an upper bound (UB):

$$\theta^{(k+1)} = \begin{cases} \text{LB} & \text{if } \theta^{(k)} + \mathbf{J}^{-1} \Delta \mathbf{e} < \text{LB} \\ \text{UB} & \text{if } \theta^{(k)} + \mathbf{J}^{-1} \Delta \mathbf{e} > \text{UB} \\ \theta^{(k)} + \mathbf{J}^{-1} \Delta \mathbf{e} & \text{otherwise} \end{cases} \quad (12)$$

To apply the rotation to a joint, the quaternion is first represented as a matrix (see Equation 1). The local transformation matrix is then computed by multiplying the bind transformation matrix with the new rotation matrix.

Finally, in the last iteration step, the new rotation of each joint in the skeleton is calculated by updating the combined transformation matrix:

$${}^0\mathbf{C}_i = {}^0\mathbf{P}_i \cdot \mathbf{L}_i, \quad \mathbf{C}_i \in \mathbb{R}^{3 \times 3} \quad (13)$$

where ${}^0\mathbf{C}_i$ is the combined transformation matrix, $\mathbf{L}_i$ is the local transformation matrix of the i^{th} bone and ${}^0\mathbf{P}_i$ is the combined transformation matrix of the i^{th} bone's parent. Thus, a bone's combined transformation matrix is determined by first applying its local transformation and then by applying the local transform of its parent. If the maximum number of iterations is not yet reached, we go back to the first step. Otherwise, the algorithm terminates.

3.1.4 Animation

To animate the avatar, the calculated new orientations has to be applied to the bone joints. While solving the IK, the quaternion of each joint was updated, depending on the desired position and orientation of the end-effector. However, the difference between the new joint orientation, compared to the joint orientation from the previous frame, is eventually large. In this case, we have to interpolate between the quaternions by applying the SLERP method. Then, to calculate the new skinned vertex position, we first have to calculate a final transformation ${}^0\mathbf{F}_j$ by multiplying the combined transform ${}^0\mathbf{C}_j$ with the inverse transform matrix.

By iterating over all vertices, we calculate the new position of every vertex with respect to the bone rotation. Because each vertex and, above all, vertices near the joint can be influenced by several bones, the new position of the vertex is determined by a weighted average of the influential bone transformations. Therefore, for each vertex, the new position vector is determined by multiplying the final transform matrix of the bone influencing this vertex with the current position vector

and the bone weight. The final vertex position $\mathbf{v}_{new}$ is computed as proposed by Kavan [23]:

$$\mathbf{v}_{new} = \sum_{n=0}^{n-1} ({}^0\mathbf{F}_n \cdot \mathbf{v}) \cdot w_n, \quad \mathbf{v}_{new} \in \mathbb{R}^{3 \times 1} \quad (14)$$

where $\mathbf{v}$ is the current vertex position, $\mathbf{F}_n$ is the final transform of the n^{th} bone, that influences vertex $\mathbf{v}$ and w_n is the weight of the n^{th} bone.

Finally, the vertex buffer is updated with the new calculated vertex positions and normals. The vertex shader then uses this buffer to draw indexed vertices. In the current implementation, the vertex skinning calculation is done on a Central Processing Unit (CPU), that runs the sequential code as fast as possible. However, especially in the context of VR games, the execution speed has to be considered. Therefore, these computations could also be carried out by a Graphical Processing Unit (GPU), that can do hundreds of calculations in parallel. These would eventually speed-up the calculations of the animation system.

The currently developed prototype can track multiple end-effectors in order to animate the virtual body. In Figure 3 the actual posture of the user and the corresponding virtual reality image is shown. In this case, the Vive Tracker was attached to the foot in order to synchronize the virtual leg with the user's motions.

3.2 Development of a Latency Measurement Tool

In the second part, a latency measurement tool was implemented to measure the total delay of the implemented body tracking system. The total delay of a VR-based tracking system represents the time at which the motion occurs, to the time the tracking system detects this motion and the results are displayed on the HMD. Using this latency measurement tool, we expect to show

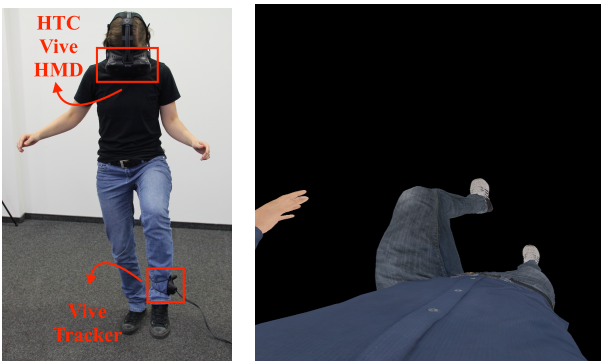


Fig. 3 The user's actual posture (left) and a first-person perspective of the user looking down at the virtual body (right).

that the total delay of the developed method stays below 20 ms. This would satisfy the requirement for a real-time VR experience [32] [22].

The easiest way to measure the latency is to record the Vive Tracker and the output screen at the same time with a high-speed camera. When the Tracker moves in a specific pattern, a video can be analyzed frame by frame in order to identify distinctive motions and to calculate the delay between events. As proposed by Steed [40], the Tracker can be bind to a string in order to swing it. Applying this approach, the features can be identified by extracting the horizontal positions of both objects and detecting the local minima and maxima. By calculating the deviation between the local minimum or maximum of the Tracker and the local minimum or maximum of the virtual object, the frame difference between them can be calculated and finally, the total delay can be determined. Using this approach of identifying the frames at which the Tracker changes the direction and calculating the difference between them, the measurements include only the end-to-end latency of the developed application. However, we cannot make any statement about the accuracy.

To measure the end-to-end latency, the steps described below should be followed:

1. Load the video and select two bounding boxes, that contain the Vive Tracker and the corresponding virtual object
2. Extract the horizontal position of both bounding boxes in each frame
3. Apply the Gaussian kernel to smooth the data and normalize the samples
4. Identify distinctive motions by detecting local peaks
5. Calculate the frame difference and determine the latency

Once a capture of the targets in motion has been taken, an algorithm tracks their locations throughout the video. For this purpose, we use a KCF tracking algorithm, which is already implemented by OpenCV¹⁵ and is able to track multiple objects simultaneously. After the KCF tracker is initialized, a video can be loaded and two initial bounding boxes, one for the marker and one for the VR object, are defined.

To detect distinctive motions, the horizontal positions of both bounding boxes are smoothed and normalized. By applying the Gaussian kernel at different scales, we reduce the noise and ensure smooth data. Then the peaks, thus, the minima and maxima of both curves have to be identified. When a peak is detected,

¹⁵ KCF Tracker: http://docs.opencv.org/trunk/d2/df/classcv_1_1TrackerKCF.html, last visited on February 17th, 2017

the frame number is saved in an array. This results in two equally long arrays, one containing frame numbers of the peaks for the Vive Tracker and the other containing frame numbers for the virtual object.

Once all frames have been processed and the features were extracted, the time difference between the two events $t_{\text{motion}} - t_{\text{display}}$ is calculated. The latency is calculated as follows:

$$t = \frac{1}{N} \sum_{i=0}^N (X_i - \tilde{X}_i) \cdot \frac{1000}{\text{FPS}} \text{ms}, \quad (15)$$

where N is the number of peaks identified in a video, X_i is the frame number of the i^{th} peak for the Vive Tracker, and $\tilde{X}_i$ is the frame number of the i^{th} peak for the virtual object. The total delay is then determined by multiplying the mean difference between two events with the time, the camera needs to capture a new image. The flow chart of the latency measurement tool, including the entire calculation, is presented in Figure 4.

4 Results

The main objective of this research is to evaluate the end-to-end latency of the proposed method as well as the accuracy and reliability of motions tracking with a Vive Tracker. To evaluate the performance of the tracking system, we determined the total delay with the developed latency measurement tool. We analyzed which limitations and errors can occur and how we can

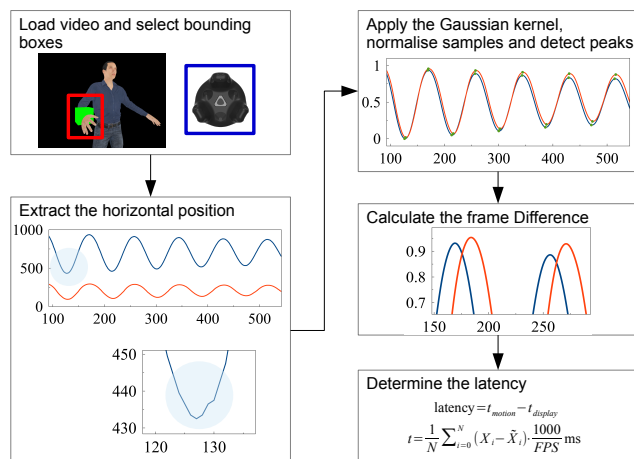


Fig. 4 Flow chart of the latency measurement tool. The blue curve corresponds to the horizontal positions of the Vive Tracker and the red curve corresponds to the horizontal positions of the virtual object. One can easily recognize, that the red curve is shifted to the right. This horizontal difference of both curves indicates the latency of the distinctive motions.



Fig. 5 Strapped Vive Tracker during the evaluation (left). To track the leg movements, an additional Tracker must be strapped to the ankle.

minimize them. In addition, we evaluated if the person exploring VR can perceive the avatar movements as her/his own.

For the evaluation, an HTC Vive was connected to a computer running Microsoft Windows 10 to enable full-body tracking of the user. The computer has an 3.30 GHz Intel CoreTM i7-5820K processor with 16 GB RAM and a NVIDIA GeForceTM GTX 980 graphics card. It has sufficient processing power and it fulfills the minimal requirements for the HTC Vive¹⁶.

4.1 Evaluation of the Body Tracking System

The evaluation of the body tracking system was conducted in the TU Darmstadt, Germany. A total of 13 subjects participated, 1 female and 12 males with an average age of 27 years. First, the participants were asked to fill out the pre-study questionnaire, which included personal questions about gender, age, game consumption habits, VR experience and body tracking. Afterward, the participants tried out the simulation. They strapped the Vive Tracker to the left and right wrist, as it can be seen in Figure 5. The bands were able to fix the Tracker strongly enough so that it could not slip away and could remain in place for the time of the evaluation. For the immersive experience, the subjects were wearing an HTC Vive HMD to view the virtual character from the first-person perspective. The simulation was then run for approximately 5 Minutes. Finally, the participants were again asked to fill out the post-study questionnaire.

The results of the pre-study questionnaire (see Figure 6) showed, that almost the two-fifths (38.5%) play video games more than seven hours per week. However, almost one-third (30.8%) never play video games. Furthermore, the majority (84.6%) had already prior experience with HMDs, e.g., Oculus Rift or HTC Vive. On

¹⁶ Minimum requirements: <https://www.vive.com/us/ready/>, last visited on February 5th, 2018

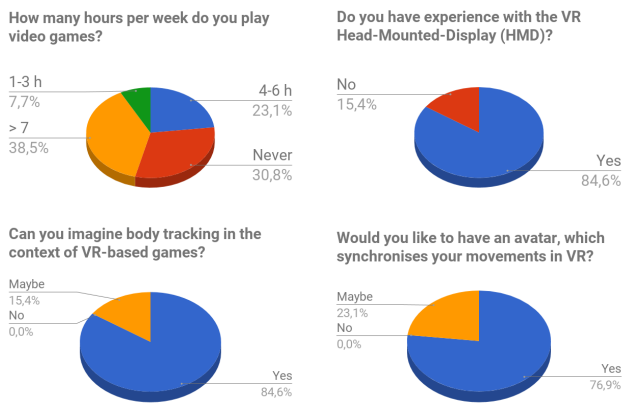


Fig. 6 The results of the pre-study questionnaire.

the one hand, some subjects reported, that they already suffered from cybersickness, such as dizziness, nausea or a headache while wearing a HMD for a longer time. On the other hand, some of them also explicitly stated, that they never feel side effects of any kind. The majority, however, criticised an insufficiently low resolution of the currently available HMDs. Most of the participants (84.6%) can imagine body tracking in the context of VR-based games. Three-fourth (76.9%) would like to have an avatar, which synchronizes the movements and one-fourth (23.1%) is not sure if they want an avatar. However, the results of the post-study questionnaire reveal that almost all participants (92.3%) would like to have body tracking in VR games. Body tracking has been proposed in various application scenarios, e.g., goalkeeper, ego-shooter games and other first-person games, where the player can interact with virtual objects. The subjects furthermore suggested body tracking in medicine, where the user can remotely control the surgery with her/his own movements. VR in a combination with the body tracking could be used to train the users to complete a certain task correctly. Further scenarios could include virtual video calls or meetings.

Figure 7 shows the mean response of selected questions from the post-study questionnaire, along with their associated standard deviations. The results show that the majority in general likes the VR as well as the idea of body tracking in the VR¹⁷. Furthermore, it was found that the subjects could feel as if they were “present” in the virtual environment and could also identify themselves with the avatar¹⁸. The subjects liked it, that they could see the full-body avatar and not only the arms, as

¹⁷ Question: “I find the VR in general exciting”, five-level Likert scale, $N = 13$, $AVR = 4.92$, $SD = \pm 0.27$, question: “I like the idea of body tracking in VR”, five-level Likert scale, $N = 13$, $AVR = 4.92$, $SD = \pm 0.27$

¹⁸ Question: “I felt like I was a part of the VR”, five-level Likert scale, $N = 13$, $AVR = 4.3$, $SD = \pm 0.48$, question: “I

in the most current first-person games. Some subjects stated they liked that the avatar reflects the movements of the arms, however, they missed the tracking of the legs. Since the current implementation can handle multiple end-effectors, we would only need additional Vive Tracker in order to track the hands and legs simultaneously.

The results of the body tracking show that the movements of the avatar corresponded to the real movements of the user and that tracking provided accurate positions¹⁹. However, the orientation of the elbow did not always correspond to the reality. Some subjects reported, that the elbow was sometimes twisted or that they could perceive some inaccuracies in arm tracking. To evaluate the accuracy, the subjects were asked to perform various movements, including small and large movements at different speeds. Although the most subject stated that the tracking was quite accurate, sometimes the position of the virtual hands was different from the position of the real hands. This was especially noticeable when touching the own arms or hands. Subjects reported, that while the fingers were touching in the real world, the virtual fingers were too far away from each other or they were overlapping. In order to improve the accuracy, the length of the upper and lower arm could be considered. Moreover, a collision detection should be incorporated in order to prevent that the body limbs are overlapping.

The results of the tracking itself suggest, that the subjects only perceived a low latency²⁰. A low latency was identified for very fast movements by only one participant, who plays the games more than seven hours a week. As it will be described in the next section, the total delay of the tracking remains very low, at 6.71 ± 0.80 ms. Some subjects reported that the tracking had some jitter problems²¹. This sometimes happened, when the Vive Tracker was not able to detect enough laser sweeps from the base stations. When holding an arm in a steady position, no noise or other tracking errors could be identified. Three subjects, however, reported, that the arm was for a short moment locked at some position. Then, after the subjects stretched the arm again, the arm “jumped” to the right position.

could identify myself with the avatar”, five-level Likert scale, $N = 13$, $AVR = 4.07$, $SD = \pm 0.49$

¹⁹ Question: “The movements in the VR have corresponded to the real movements”, five-level Likert scale, $N = 13$, $AVR = 4.23$, $SD = \pm 0.59$, question: “The tracking was accurate”, five-level Likert scale, $N = 13$, $AVR = 4.07$, $SD = \pm 0.64$

²⁰ Question: “The movements of the avatar were delayed”, five-level Likert scale, $N = 13$, $AVR = 1.23$, $SD = \pm 0.43$

²¹ Question: “The tracking had some jitter problems”, five-level Likert scale, $N = 13$, $AVR = 2.37$, $SD = \pm 1.25$

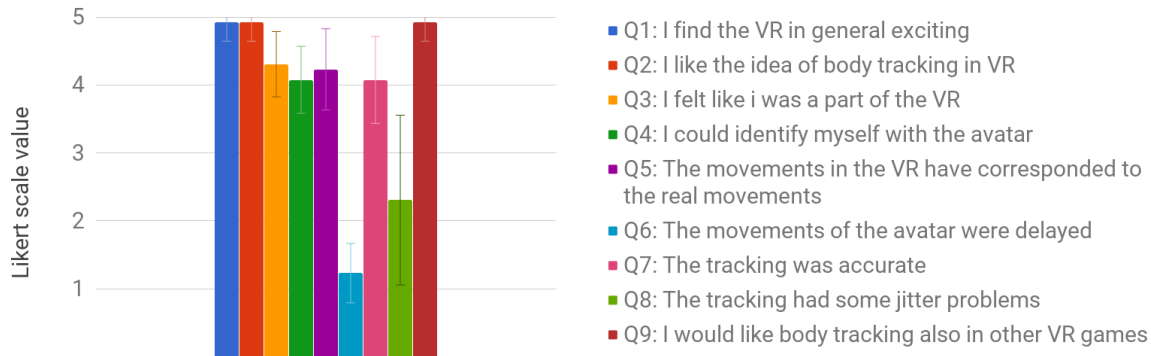


Fig. 7 Rating the simulation, where 1 stands for *totally incorrect*, 2 for *kind of incorrect*, 3 for *not sure*, 4 for *kind of correct* and 5 for *totally correct*. The bar shows the mean responses to questions from the post-study questionnaire and the error bars indicate the standard deviations.

Finally, the results show that almost all subjects would like to have body tracking also in other VR games²². For a more immersive experience, finger recognition should be included, e.g., using a Leap Motion²³ device or special gloves such as Hi5 VR Gloves²⁴, VRgluv²⁵, HaptX²⁶, VRtouch²⁷), which can detect the motion of each individual finger. Some subjects stated that they liked to have nothing to hold in their hands. However, the Vive Tracker that was fixed to a hand, is big and is actually developed to be attached to sporting equipment. In the future work, we could also create a smaller and lighter Tracker, e.g. using only a few infrared sensors in a bracelet that can be attached to the hand as well as ankle.

The total number of 13 participants is too low for any statistical conclusions. In addition to the problem of too few participants, some of them were friends or colleagues. Although the study participants were asked to answer the questions honestly, one cannot rule out that the feedback still was more beneficial. Therefore, an evaluation with more subjects should be carried out in a future work.

4.2 Evaluation of Latency Measurements

For the second part of the evaluation, an estimation using a latency measuring tool based on an automatic frame counting method using a video camera was made. The Vive Tracker and the output of the virtual environment were captured by a single phone camera (iPhone SE) at 240 FPS and 1280×720 pixels resolution. The virtual environment was rendered on a gaming monitor with a 144 Hz refresh rate and G-Sync support. Thanks to G-Sync, the frame rate of the output device can be adapted which allows us to maintain the frame rate at the highest possible value of the VR system. Thus, it was possible to ensure a frame rate of 90 FPS.

In the first step, the latency of the Vive Tracker itself was determined. Thus, the latency of the Tracker as provided by the Vive system, without further processing (thus, without IK or other calculations), was measured. The measurements have shown a latency of 6.07 ± 1.36 ms. Since the developed system cannot obtain a better latency than the one provided by the Vive system, we want to get as close to the value as possible.

In the second step, the latency of the developed body tracking system using a Vive Tracker is measured. That is the total time between making a movement, sensing it by the Vive system, solving the IK problem and displaying the motion. Table 2 shows the mean and standard deviation of the measured latency due to the different maximum number of iterations. For each trial (5, 10, 30, 50 and 100 maximum number of iterations), 9 measurements were provided and the average (AVG), as well as the standard deviation (STD), were calculated. As one can see, the body tracking system with 5 maximum number of iterations shows the worst results with an average delay of almost 200 ms. With a higher iteration number (10, 30, 50 and 100), the de-

²² Question: "I would like body tracking also in other VR games", five-level Likert scale, $N = 13$, $AVR = 4.92$, $SD = \pm 0.27$

²³ Leap Motion: <https://www.leapmotion.com>, last visited on January 19th, 2018

²⁴ Hi5 VR Glove: <https://hi5vrglove.com>, last visited on January 19th, 2018

²⁵ VRgluv: <https://vrgluv.com>, last visited on January 19th, 2018

²⁶ HaptX: <https://haptx.com>, last visited on January 19th, 2018

²⁷ VRtouch: https://www.gotouchvr.com/order_vrtouch/, last visited on January 19th, 2018

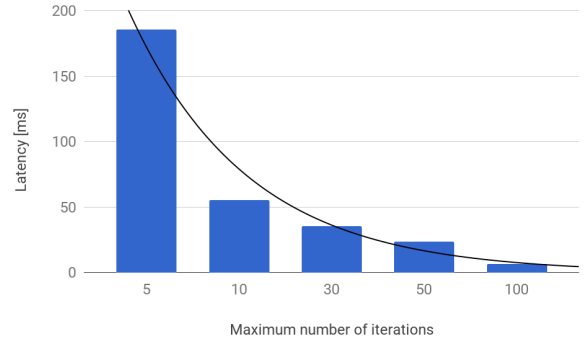
Table 2 Measured latency due to the different maximal steps of the IK solver.

Max Iterations	Latency
5	185.59 ± 1.30 ms
10	55.24 ± 1.62 ms
30	33.33 ± 0.52 ms
50	23.78 ± 0.30 ms
100	6.71 ± 0.80 ms

lay will greatly decrease. From these results, we can assume, that with a very small maximum number of iterations, the end-effector will move towards the desired position, but the joint angle will change over time only very slowly. In this case, the IK solver will not be able to provide an appropriate set of joint configurations in order to reach the desired position as smoothly, rapidly and as accurately as possible. The results of the latency evaluation suggest, that the maximum number of iterations is very important for the performance of the IK solver.

As presented in Figure 8, with a higher iteration number, the latency will exponentially fall. One would normally expect increasing computational costs in terms of time. As already mentioned before, with fast movements, a very small number of iterations will indeed move the end-effector towards the desired position. However, it will always stay too far away from this desired position. Thus, for fast movements, we obtain better results with a higher number of iterations. If the speed of the movement is very slow, even a lower number of iterations is enough for the convergence because the position in the current frame is almost the same as in the previous frame. As described in Section 3.1.3, the IK solver converges if the end-effector is close enough to the desired location or when there is no significant change between current and desired rotation. However, on average after 95 iterations there is no significant change between desired and current position as well as orientation. If the algorithm would terminate at much higher iteration number, without checking if the end-effector is close enough to the desired position, the latency would increase. Due to many calculations (e.g. calculating an inverse of a non-square matrix), it would not be possible to complete the computations before the next frame needs to be rendered. Thus, the frame rate would drop rapidly, which would decline the performance.

The end-to-end latency of 6.71 ms shows that the implemented solution can reconstruct the motions in real-time. Since the result stays well below 20 ms, it meets the requirements for real-time VR experiences [32] [22]. Compared to the results based on the publication by Jiang et al. [20] with a total latency of 7 ms, our solution provides a slightly better result. In this work, two

**Fig. 8** Visualisation of the measured latency. With the higher iteration number, the latency will exponentially fall. However, with a higher maximum number of iteration, the frame rate will drop since we would not be able to complete all computations before the next frame needs to be rendered.

Vive Controllers were used. Similarly, Seele et al. [35] also used two Vive Controller. However, only the upper body was reconstructed and no latency was measured. The Table 3 summarizes the end-to-end latencies of the related work. All these publications tracked full-body movements and visualized an avatar. As it can be seen, we could achieve similar results or even a much lower latency.

Compared to the latency of the Vive system (without IK solution) with 6.07 ms, we can still improve our method. We expected a total delay below 11.11 ms since this would satisfy the refresh rate of the HTC Vive HMD, which is 90 Hz. Thus, the latency measurements fulfill our expectations. However, because the tracked and the corresponding VR objects were captured with a camera at 240 FPS, a latency below 4.16 ms cannot be detected at all. In the future work, an even better camera, which is capable of recording at a high-speed, could be used in order to measure the latency even more accurately.

The overall evaluation results suggest that the algorithm can be further optimized. On the one hand, the current performance of the implementation can be improved, so that a smaller number of iterations would be

Table 3 Latency results of the related work

Author	Latency
Latoschik, Lugin et al. [26]	73 ms
Desai et al. [10]	< 50 ms
Kasahara et al. [22]	≈ 70 ms
Latoschik, Roth et al. [27]	< 150 ms
Jiang et al. [20]	7 ms
Schmidt et al. [34]	< 100 ms
Thomas et al. [42]	39 ms
Johnson et al. [21]	> 300 ms

needed to obtain the best solution. Therefore, we must first evaluate the performance of computing the Jacobian inverse. In the current implementation, a pseudo-inverse method is used to approximate the inverse of the Jacobian matrix. By applying a more computationally efficient approach to calculate the inverse, e.g. damped least squares, we could reduce the computational cost, complex matrix calculations, and singularity problems. On the other hand, we could minimize the user's experience of latency by predicting their movements. More specifically, we could analyze the posture of the user and their movements in order to predict the actions in the virtual environment to further reduce the latency.

5 Conclusion

In this paper, a novel body tracking system using IK approach with reduced Jacobian Matrix was developed. Such a real-time solution can be used for immersive VR-based games. By strapping only a small number of Vive Tracker to the player, the full-body motions of the player can be transformed into a virtual avatar. With the tracked motions even the gestures can be recognized in order to create multiplayer VR experiences. The evaluation with the latency measurement tool showed a very low delay of only 6.71 ± 0.80 ms. Thus, the results show that the proposed method is satisfied with the technical requirement of the HTC Vive HMD and fulfill our expectations. Furthermore, compared to the related work, our latency evaluation show similar or even better results. Our system can provide an appropriate set of joint configuration in order to reach the desired position as smoothly, rapidly and as accurately as possible in real-time. The evaluation with the participants revealed that the position and orientation of the arms were accurately tracked. Because the movements of the virtual body corresponded to the real movements of the users, the user could feel like they were a part of the VR and could identify themselves with the avatar. The evaluation with the subjects also validated that only a low end-to-end latency could be perceived by a minority.

Future research will focus on making the body tracking even more robust and reliable. The effectiveness of the iterative method to solve the IK should be improved. Therefore, we should reduce the computational cost by minimizing the maximum number of iterations that are needed to calculate the appropriate orientation of bones. Especially, because there is no objective evaluation on the accuracy in this research (the accuracy was only evaluated with the subjects), in future work a tool should be developed in order to measure how accurate the developed body tracking system is.

Since in the current evaluation the participants could see the full-body avatar, but only the hands were animated, also feet should be animated in the future work. By attaching an additional Vive Tracker to the back or hip as well as feet, the user should be able to see an animated avatar while walking, dancing or jumping. To further improve the immersion in the VR experience, the steps of the user could be identified in order to create stepping sound. Furthermore, we could do a comparative study, comparing different presence approaches, e.g. full-body animated avatar vs. only animated hands as well as even showing only the Vive Controller or Tracker.

In addition, collision detection should be considered, e.g. to interact with the environment. Collision detection is also important in the detection of body movements since we do not want the body limbs to intersect. When the user tries to touch the virtual body, the collision detection should prevent that the hands go through the body. Another important aspect is the appearance of the avatar. Therefore, a tool to personalize the avatar body, e.g., based on muscles, clothes and skin color should be integrated into the pipeline, to create an even more immersive VR experience.

References

1. Aristidou, A., Lasenby, J.: FABRIK: A Fast, Iterative Solver for the Inverse Kinematics Problem. *Graphical Models* **73**(5), 243–260 (2011)
2. Banakou, D., Groten, R., Slater, M.: Illusory Ownership of a Virtual Child Body Causes Overestimation of Object Sizes and Implicit Attitude Changes. *Proceedings of the National Academy of Sciences* **110**(31), 12846–12851 (2013)
3. Bolton, J., Lambert, M., Lirette, D., Unsworth, B.: PaperDude: A Virtual Reality Cycling Exergame. In: *CHI '14 Extended Abstracts on Human Factors in Computing Systems, CHI EA '14*, pp. 475–478. ACM, New York, NY, USA (2014)
4. Botev, J., Rothkugel, S.: High-Precision Gestural Input for Immersive Large-Scale Distributed Virtual Environments. In: *Proceedings of the 9th Workshop on Massively Multiuser Virtual Environments, MMVE'17*, pp. 7–11. ACM, New York, NY, USA (2017)
5. Caserman, P., Krabbe, P., Wojtusich, J., von Stryk, O.: Real-Time Step Detection Using the Integrated Sensors of a Head-Mounted Display. In: *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 3510–3515 (2016)
6. Chan, J.C.P., Leung, H., Tang, J.K.T., Komura, T.: A Virtual Reality Dance Training System Using Motion Capture Technology. *IEEE Transactions on Learning Technologies* **4**(2), 187–195 (2011)
7. Choi, S.W., Seo, M.W., Lee, S.L., Park, J.H., Oh, E.Y., Baek, J.S., Kang, S.J.: Head Position Model-based Latency Measurement System for Virtual Reality Head Mounted Display. *SID Symposium Digest of Technical Papers* **47**(1), 1381–1384 (2016)

8. Collingwoode-Williams, T., Gillies, M., McCall, C., Pan, X.: The Effect of Lip and Arm Synchronization on Embodiment: A Pilot Study. In: 2017 IEEE Virtual Reality (VR), pp. 253–254. IEEE (2017)
9. Dempsey, P.: The teardown: HTC Vive VR headset. *Engineering Technology* **11**(7-8), 80–81 (2016)
10. Desai, K., Raghuraman, S., Jin, R., Prabhakaran, B.: QoE Studies on Interactive 3D Tele-Immersion. In: 2017 IEEE International Symposium on Multimedia (ISM), pp. 130–137 (2017)
11. Desai, P.R., Desai, P.N., Ajmera, K.D., Mehta, K.: A Review Paper on Oculus Rift-A Virtual Reality Headset. *International Journal of Engineering Trends and Technology (IJETT)* **13**(4) (2014)
12. Farahani, N., Post, R., Duboy, J., Ahmed, I., Kolowitz, B.J., Krinchai, T., Monaco, S.E., Fine, J.L., Hartman, D.J., Pantanowitz, L.: Exploring Virtual Reality Technology and the Oculus Rift for the Examination of Digital Pathology Slides. *Journal of pathology informatics* **7** (2016)
13. Friðriksson, F.A., Kristjánsson, H.S., Sigurðsson, D.A., Thue, D., Vilhjálmsson, H.H.: Become Your Avatar: Fast Skeletal Reconstruction From Sparse Data for Fully-Tracked VR. In: Proceedings of the 26th International Conference on Artificial Reality and Telexistence and the 21st Eurographics Symposium on Virtual Environments: Posters and Demos, pp. 19–20 (2016)
14. Friston, S., Steed, A.: Measuring Latency in Virtual Environments. *IEEE Transactions on Visualization and Computer Graphics* **20**(4), 616–625 (2014)
15. Galna, B., Barry, G., Jackson, D., Mhiripiri, D., Olivier, P., Rochester, L.: Accuracy of the Microsoft Kinect Sensor for Measuring Movement in People with Parkinson's Disease. *Gait & Posture* **39**(4), 1062–1068 (2014)
16. Goradia, I., Doshi, J., Kurup, L.: A Review Paper on Oculus Rift & Project Morpheus. *International Journal of Current Engineering and Technology* **4**(5), 3196–3200 (2014)
17. Grochow, K., Martin, S.L., Hertzmann, A., Popović, Z.: Style-based Inverse Kinematics. *ACM Trans. Graph.* **23**(3), 522–531 (2004)
18. Huang, J., Wang, Q., Fratarcangeli, M., Yan, K., Pelachaud, C.: Multi-Variate Gaussian-based Inverse Kinematics. *Computer Graphics Forum* **36**(8), 418–428 (2017)
19. Jain, D., Sra, M., Guo, J., Marques, R., Wu, R., Chiu, J., Schmandt, C.: Immersive Terrestrial Scuba Diving Using Virtual Reality. In: Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems, pp. 1563–1569. ACM, New York, USA (2016)
20. Jiang, F., Yang, X., Feng, L.: Real-time Full-body Motion Reconstruction and Recognition for Off-the-shelf VR Devices. In: Proceedings of the 15th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry - Volume 1, VRCAI '16, pp. 309–318. ACM (2016)
21. Johnson, M., Humer, I., Zimmerman, B., Shallow, J., Tahai, L., Pietroszek, K.: Low-Cost Latency Compensation in Motion Tracking for Smartphone-based Head Mounted Display. In: Proceedings of the International Working Conference on Advanced Visual Interfaces, AVI '16, pp. 316–317. ACM, New York, NY, USA (2016)
22. Kasahara, S., Konno, K., Owaki, R., Nishi, T., Takeshita, A., Ito, T., Kasuga, S., Ushiba, J.: Malleable Embodiment: Changing Sense of Embodiment by Spatial-Temporal Deformation of Virtual Human Body. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, CHI '17, pp. 6438–6448. ACM, New York, NY, USA (2017)
23. Kavan, L., Sloan, P.P., O'Sullivan, C.: Fast and Efficient Skinning of Animated Meshes. *Computer Graphics Forum* **29**(2), 327–336 (2010)
24. Kenwright, B.: Real-Time Character Inverse Kinematics Using the Gauss-Seidel Iterative Approximation Method. In: International Conference on Creative Content Technologies, vol. 4, pp. 63–68 (2012)
25. Lange, B., Rizzo, S., Chang, C.Y., Suma, E.A., Bolas, M.: Markerless Full Body Tracking: Depth-Sensing Technology within Virtual Environments. In: Interservice/industry training, simulation, and education conference (I/ITSEC) (2011)
26. Latoschik, M.E., Lugrin, J.L., Habel, M., Roth, D., Seufert, C., Grafe, S.: Breaking Bad Behavior: Immersive Training of Class Room Management. In: Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology, VRST '16, pp. 317–318. ACM, New York, NY, USA (2016)
27. Latoschik, M.E., Roth, D., Gall, D., Achenbach, J., Waltemate, T., Botsch, M.: The Effect of Avatar Realism in Immersive Social Virtual Realities. In: Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology, VRST '17, pp. 39:1–39:10. ACM, New York, NY, USA (2017)
28. Melo, M., Rocha, T., Barbosa, L., Bessa, M.: The Impact of Body Position on the Usability of Multisensory Virtual Environments: Case Study of a Virtual Bicycle. In: Proceedings of the 7th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion, DSAI 2016, pp. 20–24. ACM, New York, NY, USA (2016)
29. Nakamura, Y., Hanafusa, H.: Inverse Kinematic Solutions With Singularity Robustness for Robot Manipulator Control. *Journal of Dynamic Systems, Measurement, and Control* **108**(3), 163–171 (1986)
30. Orin, D.E., Schrader, W.W.: Efficient Computation of the Jacobian for Robot Manipulators. *The International Journal of Robotics Research* **3**(4), 66–75 (1984)
31. Peck, T.C., Seinfeld, S., Aglioti, S.M., Slater, M.: Putting Yourself in the Skin of a Black Avatar Reduces Implicit Racial Bias. *Consciousness and Cognition* **22**(3), 779 – 787 (2013)
32. Raaen, K., Kjellmo, I.: Measuring Latency in Virtual Reality Systems. In: K. Chorianopoulos, M. Divitini, J. Baalsrud Hauge, L. Jaccheri, R. Malaka (eds.) *Entertainment Computing - ICEC 2015*, pp. 457–462. Springer International Publishing, Cham (2015)
33. Roberts, D., Duckworth, T., Moore, C., Wolff, R., O'Hare, J.: Comparing the End to End Latency of an Immersive Collaborative Environment and a Video Conference. In: Proceedings of the 2009 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, DS-RT '09, pp. 89–94. IEEE Computer Society, Washington, DC, USA (2009)
34. Schmidt, D., Kovacs, R., Mehta, V., Umapathi, U., Köhler, S., Cheng, L.P., Baudisch, P.: Level-Ups: Motorized Stilts That Simulate Stair Steps in Virtual Reality. In: Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems, CHI EA '15, pp. 359–362. ACM, New York, NY, USA (2015)
35. Seele, S., Misztal, S., Buhler, H., Herpers, R., Schild, J.: Here's Looking At You Anyway!: How Important is Realistic Gaze Behavior in Co-located Social Virtual Reality

- Games? In: Proceedings of the Annual Symposium on Computer-Human Interaction in Play, CHI PLAY '17, pp. 531–540. ACM, New York, NY, USA (2017)
36. Shoemake, K.: Animating Rotation with Quaternion Curves. In: Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '85, pp. 245–254. ACM, New York, NY, USA (1985)
37. Shum, H., Ho, E.S.: Real-time Physical Modelling of Character Movements with Microsoft Kinect. In: Proceedings of the 18th ACM Symposium on Virtual Reality Software and Technology, VRST '12, pp. 17–24. ACM (2012)
38. Sra, M., Schmandt, C.: MetaSpace II: Object and Full-Body Tracking for Interaction and Navigation in Social VR. CoRR **abs/1512.02922** (2015)
39. Staff, D.T.: Spec Comparison: Does the Rift's Touch Update Make It a True Vive Competitor? (2016). URL <https://www.digitaltrends.com/virtual-reality/oculus-rift-vs-htc-vive/>. Last visited on 4.5.2017
40. Steed, A.: A Simple Method for Estimating the Latency of Interactive, Real-time Graphics Simulations. In: Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology, VRST '08, pp. 123–129. ACM, New York, NY, USA (2008)
41. Tao, G., Archambault, P.S., Levin, M.F.: Evaluation of Kinect Skeletal Tracking in a Virtual Reality Rehabilitation System for Upper Limb Hemiparesis. In: 2013 International Conference on Virtual Rehabilitation (ICVR), pp. 164–165 (2013)
42. Thomas, J.S., France, C.R., Leitkam, S.T., Applegate, M.E., Pidcoe, P.E., Walkowski, S.: Effects of Real-World Versus Virtual Environments on Joint Excursions in Full-Body Reaching Tasks. IEEE Journal of Translational Engineering in Health and Medicine **4**, 1–8 (2016)
43. Tsai, T.C., Chen, C.Y., Su, G.J.: U-Art: Your Art and Ubiquitous Art. In: Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers, UbiComp/ISWC'15 Adjunct, pp. 1295–1302. ACM, New York, NY, USA (2015)