Compensating the Effect of Communication Delay in Client-Server-based Shared Haptic Virtual Environments

CLEMENS SCHUWERK, XIAO XU, RAHUL CHAUDHARI and ECKEHARD STEINBACH, Technische Universität München

Shared Haptic Virtual Environments can be realized using a client-server architecture. In this architecture, each client maintains a local copy of the virtual environment (VE). A centralized physics simulation running on a server calculates the object states based on haptic device position information received from the clients. The object states are sent back to the clients to update the local copies of the VE, which are used to render interaction forces displayed to the user through a haptic device. Communication delay leads to delayed object state updates and increased force feedback rendered at the clients. In this paper, we analyze the effect of communication delay on the magnitude of the rendered forces at the clients for cooperative multi-user interactions with rigid objects. The analysis reveals guidelines on the tolerable communication delay. If this delay is exceeded, the increased force magnitude becomes haptically perceivable. We propose an adaptive force rendering scheme to compensate for this effect, which dynamically changes the stiffness used in the force rendering at the clients. Our experimental results, including a subjective user study, verify the applicability of the analysis and the proposed scheme to compensate the effect of time-varying communication delay in a multi-user SHVE.

Categories and Subject Descriptors: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities; H.5.2 [Information Interfaces and Presentation]: User Interfaces—Haptic I/O

General Terms: Algorithms

Additional Key Words and Phrases: shared haptic virtual environment, multi-user, collaboration, haptic rendering, communication delay, perceived transparency

ACM Reference Format:

1. INTRODUCTION

Humans heavily rely on kinaesthetic and tactile perception while interacting with their environment [Robles-De-La-Torre 2006]. Thus, in many areas such as teleoperation systems, teaching and training applications, assembly and surgery simulations, scientific data visualization, or entertainment, haptic technology has great potential to enhance traditional human-machine interaction [Saddik 2007]. Early studies about the ability to touch and physically interact with objects in a virtual environment (VE) have shown that haptic feedback mediated through specialized haptic inter-
Fig. 1. SHVE based on a client-server architecture with local force rendering at the clients and a centralized physics simulation at the server.

faces leads to an increased sense of immersion [Srinivasan and Basdogan 1997] and improved task performance [Basdogan et al. 2000]. It also leads to an increased sense of togetherness, if more than one user is involved in a so called Shared Haptic Virtual Environment (SHVE) [Basdogan et al. 2000].

At the same time, advances in communication technology have stimulated research of applications, where users interact with distant real or virtual environments. Teleoperation systems, where the human user controls a remote robot, attempt to achieve a high level of immersion by providing audio-visual-haptic feedback to the human. Research challenges in such systems mainly originate from the closed control loop between the remote robot and the local user as well as the communication impairments in today’s communication infrastructure like the Internet. Communication delay, for example, leads to system stability problems, which may destroy the hardware or even harm the human operator [Lawrence 1993]. Similar challenges also appear in SHVEs, but since they are virtual simulations, there is a higher flexibility in their implementation, as explained in the following.

1.1 SHVE architectures

Different communication architectures have been conceived for networked collaborative haptic applications. In a straightforward implementation, the complete VE is rendered on a centralized server (e.g. in the cloud) based on the position information received from the clients [Buttolo et al. 1997]. The rendered haptic force feedback together with a video representation of the scene is returned to the clients. Each client-server connection resembles a traditional teleoperation system, where the power variables velocity and force are transmitted over the network. Even a small delay in the communication between the server and the client leads to instabilities, as experienced in Souayed et al. [2004] and analyzed in Fotoohi et al. [2007].

In a virtual environment, parts of the application can be shifted to the user-side [Buttolo et al. 1997]. Two general architectures have emerged, based on either the client-server (CS) (e.g. [Matsumotoy et al. 2000; Hikichi et al. 2002; Ishibashi and Kaneoka 2005; Marsh et al. 2006; Lee and Kim 2009]) or the peer-to-peer (P2P) (e.g. [Kim et al. 2004; Cheong et al. 2005; Iglesias et al. 2007; Sankaranarayanan and Hannaford 2008; Huang and Lee 2013]) communication paradigms.

**Client-server architecture.** The aforementioned server-based approach is improved by maintaining a copy of the VE at the clients and executing collision detection and force rendering locally as illustrated in Figure 1 [Buttolo et al. 1997]. A globally valid object state is calculated at the server by means of a centralized physics simulation and transmitted to the clients to update the local object state databases.

Penalty-based force rendering algorithms [Ruspini et al. 1997] are commonly used at the client and the server to calculate interaction forces with the virtual objects. They implement a spring-damper virtual coupling between the virtual probe and the virtual proxy. The virtual probe is the representation of the haptic device in the VE. The virtual proxy is an additional virtual object that is restrained to stay on the object surface to hide the device penetration into the object from the user.
Compensating the Effect of Communication Delay in Client-Server-based Shared Haptic Virtual Environments

Rendering interaction forces at the client has the benefit that the transmission of force signals, which requires high packet rates and low-delay communication [Steinbach et al. 2012], is not necessary. The control loop between the haptic device and the VE is closed locally. Hence, force feedback is calculated without delay. Communication delay, however, leads to delayed object state updates at the clients and, as a consequence, to reduced operability [Matsumotoy et al. 2000] and increased forces rendered at the clients [Matsumotoy et al. 2000; Hikichi et al. 2001; Lee and Kim 2009].

Peer-to-peer architecture. The second approach, based on the peer-to-peer communication paradigm, uses a fully-distributed architecture. The object states are simulated at each peer by a local physics simulation. Peers continuously exchange their current device position or rendered interaction forces to drive the local physics simulations. The avoidance of the communication with the centralized physics simulation server is an advantage of the P2P approach. This, however, comes at the cost of consistency issues in the presence of communication delay. The local physics simulations use outdated information about the other users and become inconsistent [Buttolo et al. 1997]. Consistency maintenance and synchronization schemes are necessary, but also challenging [Sankaranarayanan and Hannaford 2008; Delaney et al. 2006a; 2006b; Glencross et al. 2007]. In Sankaranarayanan and Hannaford [2008], for example, the P2P coupling schemes do not achieve the same level of consistency as the CS-based approach.

1.2 Problem statement and considered scenario

Both architectures struggle with communication delay, but in two different ways. Depending on the type of the application, the computational complexity and the network conditions, either the CS, the P2P or even hybrid solutions may be preferred [Marsh et al. 2006; Iglesias et al. 2007].

We argue that for smaller communication delay values, the advantage of consistency in the CS architecture outweighs the increase of responsiveness in the P2P architecture. Additionally, computationally demanding physics simulations per client are avoided. A cloud-based physics simulation server can be used instead, entailing the general advantages of cloud-based services like scalability, flexibility or availability [Armbrust et al. 2010].

Hence, we consider a client-server architecture (see Figure 1) in this paper. The virtual objects in this study are movable and rigid with a non-deformable surface. If the user pushes against an object and communication delay exists, the object will not move immediately because the object motion is simulated on the server and has to be transmitted to the client over the communication network. The penetration depth of the virtual probe into the object increases as the user further tries to move the object while waiting for the object state update from the server. In turn, the force displayed to the user is increased [Matsumotoy et al. 2000; Hikichi et al. 2001; Lee and Kim 2009].

The magnitude of the forces rendered at the clients during a cooperative interaction of two users is plotted in Figure 2 to better illustrate the effect of communication delay. The underlying SHVE implementation is described in Section 3.1. The force magnitude at the clients obviously increases with increasing round-trip-time (RTT), denoted with $R_1$ and $R_2$ in Figure 2. The higher the communication delay, the bigger the effect on the rendered force.

Fig. 2. Cooperative interaction of two users with shared objects. Magnitude of the rendered force at the client 1 and client 2 with zero communication delay (left) and in the presence of communication delay (center) for a cooperative interaction (right). The communication delay obviously leads to increased rendered forces at the clients.
The increased force magnitude at the clients leads to a loss of haptic transparency of the SHVE. Transparency in the context of teleoperation systems implies that the human can not distinguish between direct interaction with the environment and remote interaction through the teleoperation system [Raju et al. 1989]. Similarly, the client-server architecture is called haptically transparent if the user cannot perceive a difference between a fully locally simulated interaction and the interaction through the distributed system, where the physics simulation is running only on the server. A loss of transparency can have critical implications for human decision making, e.g., in haptically enhanced virtual surgical training [Hirche and Buss 2012].

1.3 Contribution and Organization

We investigate the haptic transparency of the client-server architecture in the presence of communication delay and propose a compensation scheme to achieve perceptual transparency in a multi-user application with time-varying communication delay. The presented work is an extension of our previous work in Schuwerk et al. [2014a], where we only considered the special case of a single user, constant delay and a very simple VE. The main contributions of this paper are:

1. The extension of the single-user model from Schuwerk et al. [2014a] to a multi-user model. This includes a generalized model of the physics simulation on the server, which is necessary to apply the proposed scheme to more complex VEs (Section 3).
2. An analysis of the effect of delay on the transparency in the CS architecture for multiple users (Section 4).
3. An extension of the delay compensation scheme from Schuwerk et al. [2014a] for multiple users and time-varying delay (Section 5).

Our analysis of the effect of communication delay on the rendered forces at the clients reveals guidelines on how much delay is tolerable before its effect becomes haptically perceivable. These guidelines incorporate the simulation parameters, the user interactions and the limitations of human perception. The proposed compensation scheme can be seen as a method to achieve haptic transparency if this tolerable communication delay is exceeded.

The remainder of this paper is structured as follows. We review relevant work that deals with client-server-based SHVEs in the following. Section 3 establishes a model to represent the previously introduced CS architecture. This model is used in Section 4 to analyze the transparency of the CS architecture for multi-user interactions. Based on these results, we introduce the scheme to compensate for the loss of transparency due to the communication delay in Section 5. Finally, the proposed scheme is evaluated objectively and subjectively in Section 6.

2. RELATED WORK

Early works adopting the CS architecture for SHVEs recognized a subjective performance degradation with increasing RTT. It was found that a RTT of no more than 60ms [Matsumotoy et al. 2000] or 100ms [Hikichi et al. 2001] can be allowed. In Hikichi et al. [2001], the phenomenon of objects being perceived as heavier than they actually are is attributed to the increased force feedback due to delay.

In Hikichi et al. [2002], buffers with changing size are used to compensate for out-of-order updates due to delay jitter (time-varying delay), leading to piecewise constant communication delay. However, the effect of the delay on the rendered forces and the system’s transparency is not investigated.

To compensate for the increased rendered force due to constant communication delay, Fujimoto and Ishibashi [2004] propose to reduce the spring coefficient $k$ used for the local force rendering at the clients. A simple heuristic is used to calculate a reduced stiffness. Following this idea, the algorithm proposed in Lee and Kim [2009] chooses a reduced value for $k$ based on a single-user model of the CS architecture. Therein, the server-side physics simulation is approximated with a mass-damper system, which makes it difficult to model various contacts between surfaces.

Using the same model, we propose a refined and extended version of the algorithm in Lee and Kim [2009] in our previous work [Schuwerk et al. 2014a], which considers different possible interactions of the user with the virtual
object that may or may not require a reduced stiffness. Furthermore, a subjective test shows that the proposed scheme compensates the effect of constant communication delay in the tested range of up to 150 ms.

The aforementioned works in Fujimoto and Ishibashi [2004], Lee and Kim [2009] and Schuwerk et al. [2014a] investigate only the case of constant delay during a single user interaction. Time-varying delay, again for a single user interaction, is investigated in Suzuki et al. [2014]. A subjective experiment shows that that the heuristic to calculate a reduced stiffness originally proposed in Fujimoto and Ishibashi [2004] can be adopted for time-varying delay.

In this work, we present a more detailed model of the SHVE, which is necessary to generalize previous work for various simulation parameters and multi-user interactions. It is used to identify the parameters necessary for the proposed compensation scheme. To the best of our knowledge, the work at hand is the first one to introduce an analysis of the effect of time-varying delay and a scheme to compensate for it for cooperative multi-user interactions.

The problem of communication delay in the CS architecture is approached differently in Marsh et al. [2006]. A server is dynamically selected from a set of available simulation servers based on QoS criteria. Breaks in the user interaction are exploited to eventually migrate the simulation state to a different server. This server selection reduces the communication delay for some users, but may affect some others adversely. Additionally, communication delay may not be mitigated completely. Hence, a method to compensate the effect of communication delay as proposed in this work is still necessary, as even delays of some milliseconds might be perceivable [Schuwerk et al. 2014a].

3. CLIENT-SERVER IMPLEMENTATION AND MODEL

This section first details our implementation of a client-server-based SHVE, which we consider in the remainder of this paper. Afterwards, a linear time-invariant (LTI) model is introduced in Section 3.2 that represents a two-user client-server architecture. The LTI model is used in Section 4 to analyze the effect of communication delay on the rendered forces at the clients.

3.1 Client-server implementation

Our SHVE prototype is realized based on the CS architecture described in Section 1.1 with a centralized physics simulation on the server and local force rendering at the clients. The Geomagic touch haptic device is used at each client.

Physics simulation. The application is built on top of the Chai3D haptic library [Conti et al. 2005], which uses the OpenDynamicsEngine (ODE) [Smith 2006] to simulate rigid body dynamics. The physics simulation integrates the equations of motions over time based on all forces acting on the object. These forces can be separated into forces due to gravity, contacts with other objects (e.g. frictional contacts) and forces due to the interaction of the users with an object. ODE offers either a linearized coulomb friction or a viscous friction model to implement frictional contacts between two objects.

Rigid objects with a non-deformable surface are characterized in the physics simulation by their shape (a polygon mesh) and the object’s mass $m_o$. The result of the physics simulation is the object state $S$ containing the position $x_o$, velocity $\dot{x}_o$, rotation $r_o$ and rotational velocity $\dot{r}_o$ ($S = \{x_o, \dot{x}_o, r_o, \dot{r}_o\}$ where $x_o, \dot{x}_o$ are vectors and $r_o, \dot{r}_o$ are quaternions).

ODE applies a general damping of the calculated velocities for stability reasons. Such a non physical damping is required to ensure simulation stability due to the lack of accuracy of iterative solvers [Kaufman et al. 2008].

Force rendering. The penalty-based proxy algorithm [Ruspini et al. 1997] is used to render user interaction forces with the objects at the clients and at the server. The proxy algorithm adopts a virtual representation of the haptic device (denoted as virtual proxy) that is constrained to always stay on the object’s polygon surface, while the haptic device (denoted as virtual probe) penetrates the polygon surface. A spring-damper virtual coupling is modeled between the proxy and the probe to calculate interaction forces applied on the object and forwarded to the physics simulation. Following Newton’s Third Law, the negative of the interaction force is displayed to the user at the client. The proxy algorithm
requires collision detection, which is implemented using a hierarchical tree of axis-aligned bounding boxes [Jiménez et al. 2001].

The damping element in the virtual coupling is used to ensure a stable (passive) force rendering at the clients for high stiffness values [Adams and Hannaford 1999]. Alternatively, time-domain passivity control can be adopted [Hannafor and Ryu 2002]. Note that the communication delay and the proposed scheme to compensate for the effect of communication delay (Section 5) may lead to a slightly active behavior. However, the haptic device damping and the impedance of the human arm also contribute to system stability by dissipating energy [Gil et al. 2004]. Additional stability ensuring control schemes are not necessary in our prototype using the Geomagic Touch haptic device due to the high rendering rate at the clients, the device damping and the relatively low stiffness that the device can display mechanically. A passivity-based control scheme, e.g. similar to [Hannafor and Ryu 2002], can be used to ensure stability for large stiffness values.

\textbf{Client-server communication.} Since this work focuses on the effect of communication delay only, no traffic control schemes (e.g. investigated in Ishibashi et al. [2002] and Schuwerk et al. [2014b]) are implemented for the client-server communication. Thus, state updates are transmitted bidirectionally at the full rate of the haptic loop (1 kHz).

3.2 Client-server LTI model

![Diagram](Fig. 3. Modeling a client-server-based SHVE. Left: LTI model representing the multi-user client-server architecture. \( G_{c,j} \) and \( G_{s,i} \) denote the force rendering algorithms at the clients and the server for client \( i \), respectively. The centralized physics simulation is denoted by \( G_o \). Right: mechanical model of the point mass on the server. Both users apply a force in the same direction, leading to an object movement in this direction. The damping in the physics simulation (\(-b_0x_0\)) and the external (friction) force (\(-f_0\)) act against the movement.)

In the following, we derive the LTI model of the CS architecture shown in Figure 3. Our intention is to establish a mathematical representation of the implemented SHVE prototype that allows for an analysis of the effect of communication delay on the forces rendered at the clients.

We first focus on a 1-degree-of-freedom (DoF) LTI system to simplify the analysis and subsequently show how the obtained results are generalized to the aforementioned 3-DoF implementation. The presented model is an extension of the single-user model used in Lee and Kim [2009] and Schuwerk et al. [2014a] to a two-user model with a generalized model of the physics simulation on the server.

In the SHVE model in Figure 3, \( \dot{X}_{d,i}(s) \) and \( X_{d,i}(s) \) denote the velocity and position of the user’s virtual probe at the client \( i \) in the Laplace domain. \( X_{c,i}(s) \) is the same, but delayed position at the server.

\( G_{c,i}(s) \) represents the spring-damper-based force rendering algorithm at client \( i \). It is described by the stiffness and damping virtual coupling \( G_{c,i}(s) = k_{c,i} + b_{c,i}s \) \((k_{c,i} > 0, b_{c,i} \geq 0)\). At the clients, the locally calculated interaction forces, \( F_{d,i}(s) \), are displayed to the human via the haptic device.

Similar to other transparency analyses (e.g. in Hirche and Buss [2012]), we do not model the haptic device, i.e., we assume an ideal haptic device. This allows us to investigate the effect of delay on the rendered forces, which is independent of the device dynamics.
At the server, \( G_{1,s}(s) = k_{s,1} + b_{s,1} s \) denote the spring-damper force rendering algorithms to calculate the interaction forces of the clients \((k_{s,1} > 0, b_{s,1} \geq 0)\).

The mechanical model of the shared object is shown in Figure 3. The physics simulation is represented by \( G_o(s) \) in Figure 3 and modeled as a mass-damper: \( G_o(s) = \frac{1}{m_o s^2 + b_o s} \). Here, \( m_o \) is the object’s point mass and \( b_o \) the aforementioned non-physical damping of velocities in the simulation \((b_o > 0)\). Compared to Lee and Kim [2009] and Schuwerk et al. [2014a], the dependent input force \( F_c(s) \) is added to the server model to better incorporate possible contact scenarios between surfaces and objects. \( F_c(s) \) corresponds to \( f_c \) in the mechanical model in Figure 3. \( F_c(s) \) summarizes the external forces acting against the object movement, e.g., due to frictional contacts between surfaces. For the contact scenario shown in Figure 2, for example, \( F_c(s) \) corresponds to the Laplace transform of the friction force caused by the objects’ weight while they are pushed over the ground surface.

Note that Lee and Kim [2009] and Schuwerk et al. [2014a] model the physics simulation only with the mass-damper system \( G_o(s) \), which makes it difficult to model various contacts between surfaces. The parameter \( b_o \) has to be determined based on the current user interaction and its simulation parameters (e.g., the friction coefficient). For this reason, the parameter \( b_o \) is predetermined in Schuwerk et al. [2014a] by measuring the input-output behavior of the SHVE system. In contrast, the proposed model using \( F_c \) does not require such measurements. Hence, the proposed model supports the generalization towards various interaction scenarios.

The calculated object position on the server, \( X_o(s) \), is transmitted back to the clients over the communication network, denoted by the delay elements \( G_T(j) \) with delay \( T_j \) \((j = 1, \ldots, 4)\). The round-trip-times are denoted by \( R_1 = T_1 + T_2 \) and \( R_2 = T_3 + T_4 \) for client 1 and client 2, respectively.

We assume \( b_{s,j} = b_{s,i} = 0 \) in the following derivations to simplify the calculations. The transparency analysis in Section 4 is based on the steady state, in which the penetration depth of the haptic device into the virtual object does not change. As a result, the damping elements have no impact on the rendered forces in the steady state and can be set to zero for our analysis without loss of generality.

The delay elements are approximated by a first-order Padé series for low-frequencies \( G_T(j) = e^{-sT_j} \approx \frac{1 - \frac{1}{2} sT_j}{1 + \frac{1}{2} sT_j} \). The first-order Padé approximation limits the validity of the transfer functions derived in this section to input frequencies of \( \omega < \frac{1}{2T_j} \) [Hirche and Buss 2012]. Again, this limitation does not impair our analysis as we are interested only in the steady state \((s = j\omega = 0)\).

We first consider the interaction of user 1 with a movable object, while client 2 is not interacting with the same object. In this case, \( F_{1,2}(s) \) in Figure 3 is zero and we have to consider only the black part in the LTI model. The transfer function in (1) can be derived from Figure 3 for client 1. The derivation for client 2 is identical due to the symmetry of the model. The superscript 1 denotes that only client 1 is considered.

\[
F_{c,1}(s) = \frac{s^2 k_{o,1} m_o + k_{s,1} b_{s,1} + k_{o,1} k_{s,1} + k_{o,1} R_1 + b_{s,1} R_1}{s^3 (m_o k_{o,1} + k_{s,1} + k_{o,1} b_{s,1}) + s^2 (k_{o,1} + k_{s,1} + k_{o,1} b_{s,1}) + k_{o,1} b_{s,1}} X_o(s) + \frac{-k_{o,1} R_1}{s^3 (m_o k_{o,1} + k_{s,1} + k_{o,1} b_{s,1}) + s^2 (k_{o,1} + k_{s,1} + k_{o,1} b_{s,1}) + k_{o,1} b_{s,1}} F_c(s)
\]

If the second user is also interacting with the same object, we can calculate the following transfer function that represents the effect of this interaction on the force rendered for client 1. Note that only the force applied by client 2 on the server, \( F_{2,2}(s) \), may influence the object movement. For client 1, hence, only \( F_{2,2}(s) \) has to be considered, which can be seen as an additional input similar to \( F_c(s) \).

\[
F_{c,2}(s) = -\frac{-k_{o,1} R_1}{s^3 (m_o k_{o,1} + k_{s,1} + k_{o,1} b_{s,1}) + s^2 (k_{o,1} + k_{s,1} + k_{o,1} b_{s,1}) + k_{o,1} b_{s,1}} F_{c,2}(s)
\]

Due to the linearity of the model, we can sum \( F_{c,1}(s) \) from (1) and \( F_{c,2}(s) \) from (2) to get the force rendered at client 1 for a cooperative two-user interaction with a shared object: \( F_{c,1}(s) = F_{c,1}(s) + F_{c,2}(s) \).
3.3 Parameters of the LTI model

Most parameters that describe the LTI model in Figure 3 are already known. The force rendering parameters $k_{c,i}$, $b_{c,i}$, $k_{i}$, and $b_{i}$ are explicitly set in the haptic application. The delay between the clients and the server can be measured, for example using time stamps. Hence, $T_1$, $T_2$ and also $R_1 = T_1 + T_2$ are assumed to be known.

The mass $m_o$ is set as a parameter for the physics simulation. As previously mentioned, the physics simulation applies a general damping of velocities for stability reasons. Each velocity is multiplied with a defined factor $0 < c < 1$ after each integration time step $\Delta t$. The kinetic energy lost due to this general damping can be calculated as:

$$\Delta E = \frac{1}{2} m_o \left[ \dot{x}_o(t) \right]^2 - \left( \dot{x}_o(t) e \right)^2 = \frac{1}{2} m_o \dot{x}_o(t)^2 (1 - c^2)$$

The viscous damper $b_o$ in $G_o(s)$ is employed to model this energy dissipation. The object velocity is constant during a time step $t \to t + \Delta t$. A viscous damper dissipates the energy $W_{b_o} = b_o \dot{x}_o(t)^2 \Delta t$ during this period. With (3) and $W_{b_o}$, the parameter $b_o$ can be calculated as:

$$b_o = \frac{m_o (1 - c^2)}{2 \Delta t} > 0$$

In summary, all parameters required to model the virtual environment are known or can be calculated. $\dot{X}_{d,1}(s)$ are the user inputs. $F_e$ is calculated within the physics simulation on the server.

4. TRANSPARENCY ANALYSIS

In this section, the transparency of the CS architecture is discussed based on the presented client-server LTI model. We investigate the transparency in the steady state ($s = j\omega = 0$) for client 1 in the following. This practice is also used in the analysis of teleoperation systems (e.g. in Niemeyer and Slotine [2004]). The analysis for client 2 is identical. The steady state implies that for a constant input velocity of client 1, the object and client 2 also move with the same velocity. Otherwise the spring’s elongation would change, leading to a change in the rendered force and, hence, to an acceleration/deceleration of the object.

Using the Routh-Hurwitz stability criterion, it can be shown that the transfer functions in (1) and (2) are asymptotically stable for $b_o > 0$, independent of the delay elements and, hence, a steady state exists. Note that this does not imply stability of the complete haptic application, as the presented model does not include the haptic device and the human user.

4.1 Ideal transparency

Ideal transparency in the context of teleoperation systems requires the position and force signals on the local and remote side to be identical [Yokokohji and Yoshikawa 1994]. Similarly, the positions and forces on the clients and the server need to be identical for ideal transparency.

The force at the client 1 can be calculated as the sum of the forces from (1) and (2): $F_{e,1}(s) = F_{1,1}(s) + F_{2,1}(s)$. The steady state force response in the time-domain ($f_{c,1}$) for a unit-step input ($\dot{X}_{d,1}(s) = \dot{x}_{d,1} \frac{1}{s}$) can be calculated with the final value theorem as:

$$f_{c,1} = \lim_{t \to \infty} f_{c,1}(t) = \lim_{s \to 0} s F_{c,1}(s) = k_{c,1} \left( k_{s,1} R_1 + b_o \right) \dot{x}_{d,1} + k_{s,1} f_e - k_{c,1} \dot{x}_{s,2}$$

Here, $f_e$ denotes the amplitude of the steady state external force, $\dot{x}_{d,1}$ the amplitude of the steady state device velocity of client 1 and $f_{s,2}$ the amplitude of the steady state force of client 2 applied on the object on the server. We can rewrite (5) as follows, because $\dot{x}_{d,1} = \dot{x}_o$ in the steady state:

$$f_{c,1} = \frac{k_{c,1}}{k_{s,1}} \left( R_1 k_{s,1} \dot{x}_{d,1} + b_o \dot{x}_o + f_e - f_{s,2} \right)$$

ACM Transactions on Applied Perception, Vol. 0, No. 0, Article 0, Publication date: January 2015.
The steady state force rendered on the server for client 1 is known from the LTI model or the mechanical model in Figure 3:

\[ f_{s,1} = b_0\dot{x}_o + f_e - f_{s,2} \]  

(7)

Ideally, the server force is also displayed to the user at the client. For zero communication delay \( (R_1 = 0\text{ms}) \), the client force (6) equals the desired server force (7) only if the stiffness coefficients \( k_{c,i} \) and \( k_{s,i} \) are set to be equal. We denote this as the original stiffness setting. With this setting, the rendered steady state force at the client increases with increasing delay:

\[ f_{c,1} = R_1 k_{c,1}\dot{x}_{d,1} + b_0\dot{x}_o + f_e - f_{s,2} \]  

(8)

Note that only the first term in (8) depends on the RTT of client 1. If the original stiffness setting is used, the client interacting with the virtual object perceives an increased force magnitude in the steady state of:

\[ |\Delta f_{c,i}| = k_{c,i} R_1 |\dot{x}_{d,i}| \]  

(9)

This result is intuitive: \( |\Delta f_{c,i}| \) is caused by the increased penetration depth \( (R_i |\dot{x}_{d,i}|) \) multiplied with the object stiffness.

In summary, the requirement for ideal transparency of positions and forces being the same on the server and on the client can obviously only be fulfilled for zero communication delay \( (R_i = 0\text{ms}) \) in the CS architecture.

In the presence of delay, the object positions, as well as the haptic device positions at the client and the server will differ. For the CS architecture, the difference in positions is an accepted trade-off for a globally consistent object state calculated at the server.

The delayed object state updates also lead to increased force feedback displayed to the user (see (8), (9)). This gives the user the impression of increased weight of an object, as observed in Hikichi et al. [2001], if the object is, for example, pushed over a surface with non-zero friction as in Figure 2.

The human user might recognize the delayed object state updates visually and haptically. It has been observed that the more precise modality often dominates during the integration of various human sensory modalities [Ernst and Bulthoff 2004]. While the user might not visually recognize the delayed object acceleration for low communication delay, she/he will very likely still notice the increased feedback forces.

4.2 Perceived transparency

It is known from psychophysics that human perception has limitations in terms of detection and discrimination thresholds. The concept of perceived transparency introduced in Hirche and Buss [2012] relaxes the condition for ideal transparency defined in Yokokohji and Yoshikawa [1994]. The system is haptically perceived to be transparent, if the increased force rendered at the client (6) is not distinguishable from the ideally displayed force (7). The perceivable difference in force magnitude is commonly denoted by the Weber-fraction \( p \). It is known to be between 7%–10% of the original force magnitude for static stimuli [Jones 2000]. Research in the area of haptic compression has shown that the Weber-fraction increases with the velocity of dynamic hand/arm movements [Zadeh et al. 2007; Kammerl et al. 2010].

4.2.1 Perceivable communication delay. The perceivable communication delay is the RTT for which the effect of delay becomes perceivable. Hence, the perceivable communication delay denotes a guideline for the tolerable RTT in the CS architecture. We describe the loss of perceived transparency with the increased force magnitude at the client, because communication delay has a direct effect on the rendered forces as shown in Fig. 2.

With the Weber-fraction \( p \), (7) and \( |\Delta f_{c,i}| \) from (9) we get for client 1 in the two-user scenario:

\[ k_{c,1} R_1 |\dot{x}_{d,1}| = p |b_0\dot{x}_o + f_e - f_{s,2}| \]  

(10)
Rearranged to \( R_1 \), we get the maximum allowable RTT \( R_{1}^{\text{max}} \) for the client 1:

\[
R_{1}^{\text{max}} = p \frac{[b_o x_o + f_e - f_{s,2}]}{|\dot{x}_{d,1}|k_{c,1}}
\]  \tag{11}

If the RTT exceeds \( R_{1}^{\text{max}} \) and the user controls the haptic device around a constant operating point \( \dot{x}_{d,1} \), which equals \( \dot{x}_o \) in the steady state, the rendered force at the client exceeds the zero delay reference by \( p\% \) in the steady state. Assuming that the Weber-fraction \( p \) correctly predicts the perceivable difference in force magnitude during the current user interaction, \( R_{1}^{\text{max}} \) also predicts the perceivable communication delay.

Note that this can be easily generalized for client \( i \) with \( N \) clients manipulating the same object by replacing the force applied on the object by client 2 with the sum of all user forces acting on the object except of the force of user \( i \) \( (f_s = \sum_{j \neq i}^N f_{s,j}) \):

\[
R_{i}^{\text{max}} = p \frac{[b_o x_o + f_e - f_{s,i}]}{|\dot{x}_{d,i}|k_{c,i}}
\]  \tag{12}

Note that (11) and (12) reduce to the formulas derived in Lee and Kim [2009] and Schuwerk et al. [2014a] if only a single user is considered and the external force \( f_e \) is calculated as a viscous friction force \( (f_e = b \dot{x}_o) \), i.e., a viscous friction model between surfaces is assumed for the physics simulation.

### 4.2.2 Perceivable communication delay change

Similarly, we can also derive the perceivable communication delay change \( \Delta R_{i}^{\text{max}} \) from \( R_i \) to \( R_i \pm \Delta R_{i}^{\text{max}} \). From (8) and (9) with \( \Delta R_i \) for client 1 we get:

\[
\pm \Delta R_{1} |\dot{x}_{d,1}|k_{c,1} = \pm p|R_{1} k_{c,1} \dot{x}_{d,1} + b_o x_o + f_e - f_{s,2}|
\]  \tag{13}

Hence, the perceivable delay change is:

\[
\pm \Delta R_{1}^{\text{max}} = \pm p\frac{|R_{1} k_{c,1} \dot{x}_{d,1} + b_o x_o + f_e - f_{s,2}|}{|\dot{x}_{d,1}|k_{c,1}}
\]  \tag{14}

And in the generalized form:

\[
\pm \Delta R_{i}^{\text{max}} = \pm p\frac{|R_{i} k_{c,i} \dot{x}_{d,i} + b_o x_o + f_e - f_{s,i}|}{|\dot{x}_{d,i}|k_{c,i}}
\]  \tag{15}

In summary, the above formulas for the perceivable communication delay and the perceivable communication delay change are based on the original stiffness setting \( k_{c,i} = k_{s,i} \) and on a Weber-factor \( p \) describing the human force magnitude discrimination threshold. They constitute guidelines on how much delay is acceptable in the client-server based SHVE architecture before the effect of the delay becomes haptically perceivable. They incorporate the simulation parameters, limitations of human perception, and the current user interactions.

The reason for the increased force magnitude at the client is the RTT in the communication with the centralized server. The penetration depth of the virtual probe into the object increases proportionally to the device velocity during the round-trip-time. The fraction in (12) and (15) decreases accordingly, which suggests that also the perceivable communication delay decreases. At the same time, however, it is also known that \( p \) increases with increasing velocity during dynamic arm movements [Zadeh et al. 2007; Kammerl et al. 2010]. This suggests that the perceivable communication delay increases again. Hence, we assume that the perceivable communication delay does not decrease exactly inversely proportional to the device velocity. Rigorous subjective tests are necessary to find detection thresholds for various delay and velocity combinations. This is an interesting direction for future work.

Furthermore, if the delay varies drastically while the user is in contact with an object, other artifacts might become haptically and visually perceivable leading to a loss of perceived transparency as discussed Section 6.4.
5. COMPENSATING THE EFFECT OF DELAY

To compensate for the increased force feedback at the client due to communication delay, Fujimoto and Ishibashi [2004], Lee and Kim [2009] and Suzuki et al. [2014] propose to reduce the stiffness \( k_{c,i} \) used in the local force rendering at the client. We refined the approach from Lee and Kim [2009] for a single user and a simple contact scenario in the presence of constant communication delay in our previous work [Schuwerk et al. 2014a] to consider various types of user interactions that may or may not require a stiffness reduction. In the following, we extend this to the generalized multi-user scenario discussed in Section 3 for constant and time-varying delay. We also show how the proposed scheme is generalized to three DoFs and implemented in our SHVE.

5.1 Stiffness adaptation to compensate for the effect of delay

To reduce the effect of delay on the steady state force rendered at the client, we reduce the stiffness \( k_{c,i} \) at the client \( i \) to a value denoted as \( k'_{c,i} \). Our goal is that the steady state force magnitude in the presence of communication delay matches (using a reduced stiffness \( k'_{c,i} \)) the steady state force for the zero delay case (using the original stiffness \( k_{c,i} \)).

For client 1, \( k'_{c,1} \leq k_{s,1} = k_{c,1} \). From (6) with \( k_{c,1} \) as \( k'_{c,1} \) and (7) we get:

\[
\frac{k'_{c,1}}{k_{s,1}} |R_1 k_{c,1} \hat{x}_{d,1} + b_\omega \dot{x}_\omega + f_e - f_{s,2}| = |b_\omega \dot{x}_\omega + f_e - f_{s,2}|
\]

Rearranged, we get the reduced stiffness \( k'_{c,1} \) for client 1 (\( k_{s,1} \) is replaced with the original client stiffness \( k_{c,1} \)):

\[
k'_{c,1} = k_{c,1} \frac{|b_\omega \dot{x}_\omega + f_e - f_{s,2}|}{|b_\omega \dot{x}_\omega + f_e - f_{s,2} + R_1 k_{c,1} \hat{x}_{d,1}|}
\]

For \( R_1 = 0 \)ms the fraction on the right hand side of (17) is 1 and the stiffness \( k'_{c,1} \) to be used at the client 1 is the original stiffness value: \( k'_{c,1} = k_{c,1} \). With increasing delay, the stiffness \( k'_{c,1} \) decreases. In the general multi-user form, one can rewrite (17) with \( f_s = \sum_{j \neq 1}^N f_{s,j} \) as:

\[
k'_{c,i} = k_{c,i} \frac{|b_\omega \dot{x}_\omega + f_e - f_i|}{|b_\omega \dot{x}_\omega + f_e - f_i + R_i k_{c,i} \hat{x}_{d,i}|}
\]

In summary, if the communication delay exceeds the perceivable communication delay from (12), the stiffness used in the proxy algorithm at the client should be set according to (18). If the communication delay changes by more than \( \Delta R_i^{max} \) from (15) while the user is in contact with an object, the stiffness should be updated again.

Note that abrupt stiffness updates introduce perceivable artifacts and jumps in the rendered force feedback. We use an exponential moving average filter on the calculated stiffness value to smoothly converge to the desired value as explained in Section 5.3.

5.2 Extension from 1-DoF to 3-DoF

So far, we considered only a one-dimensional interaction, where the user movement, the object movement and the considered forces are all aligned. The terms in the steady state force at the client in (8) describe the resistive force the user feels in the presence of communication delay while interacting with the object in this one-dimensional example. For 3-DoF, the forces acting on the object on the server (\( \mathbf{f}_o \)) and the forces applied by the other clients (\( \mathbf{f}_{o,j}, j \neq i \)) are obviously not necessarily in the same direction as the movement of client \( i \).

The direction of the increased force, however, is inherently aligned to the client’s movement. The penetration depth increases along the direction of the device velocity until the object state at the client is updated, which leads to the increased force rendered at the client. We project the force vectors \( b_\omega \mathbf{x}_\omega, \mathbf{f}_c \) and \( \mathbf{f}_i = \sum_{j \neq i}^N \mathbf{f}_{o,j} \) on the direction of the movement of user \( i \) to extend the 1-DoF equations to 3-DoF.
The resistive force against the direction of the movement for client $i$ can be written as follows, where $\cdot$ denotes the dot product between two vectors and $\| \| \|$ the vector norm:

$$f_{r,i} = -\frac{(b_i\hat{x}_o + f_e - f_s) \cdot \hat{x}_{d,i}}{\|\hat{x}_{d,i}\|} = f_{r,i} \cdot \hat{x}_{d,i}$$  \hspace{1cm} (19)

The amplitude of the resistive force is denoted by $f_{r,i}$. We can rewrite (17) for the case of the resistive force acting against the user movement, i.e., the dot product in (19) being positive ($f_{r,i}$ being negative), as:

$$k_{r,i} = k_{c,i} \frac{-f_{r,i}}{f_{r,i} + \beta R_i k_{c,i} \|\hat{x}_{d,i}\|}$$  \hspace{1cm} (20)

With $f_{r,i}$ being negative, the fraction in (20) is always positive and smaller or equal to one.

We additionally introduce the blending factor $\beta$ in (20) as proposed in Schuwerk et al. [2014a]. The reduced stiffness is used to compensate the effect of increased resistive force while moving an object in the presence of communication delay. However, if the movement of the object is constrained by another surface (e.g. the ground), the stiffness adaptation leads to a new artifact. An object softer than the intended rigid object is displayed to the user, if he/she presses on the top of an object. A new stiffness $k'_{r,i}$ is chosen adaptively at the client not only based on the delay, but also depending on the triangle in contact with the virtual proxy and based on local motion constraints. The blending factor $\beta$ with $0 \leq \beta \leq 1$ is calculated as $\beta = \frac{1}{2} \arccos(|c \cdot n|)$. The unit vectors $n_i$ and $c$ denote the normal vector of the contacted triangle and the constraining surface. The vector $n_i$ is determined by the proxy algorithm at the client at runtime. Motion constraints need to be determined additionally.

We can also rewrite (12) and (15) for 3-DoF:

$$R^\text{max}_i = p \frac{-f_{r,i}}{\|\hat{x}_{d,i}\|k_{c,i}}$$  \hspace{1cm} (21)

$$\pm \Delta R^\text{max}_i = \pm p \frac{R_i k_{c,i} \|\hat{x}_{d,i}\| - f_{r,i}}{\|\hat{x}_{d,i}\|k_{c,i}}$$  \hspace{1cm} (22)

5.3 Implementation

In the following, we explain how the formulas above are used at client $i$ to calculate a reduced stiffness for the local force rendering algorithm.

Equation (20) includes $f_{r,i}$ which describes the amplitude of the resistive force the client $i$ will feel while pushing the object with the constant velocity $\hat{x}_{d,i}$. The value of $f_{r,i}$ depends on the contact forces rendered in the physics simulation and the forces applied by other users on the server. Hence, the knowledge to calculate the reduced stiffness using (20) is not directly available at the client.

We include the forces acting on the objects in the object state $S$ (see Section 3.1) transmitted from the server to the clients: $S = \{\hat{x}_o, \hat{x}_e, r_e, f_e, f_o\}$. Here, $f_e$ represents forces in the physics simulation due to gravity, frictional contacts and contacts with other objects (i.e. $f_e$ corresponds to $F_e(s)$ in Figure 3) and $f_o = \{f_{o,1}, ..., f_{o,n}\}$ corresponds to the forces on the object applied by the users at the server.

This information, together with the measured RTT at the client ($R_i$), the original stiffness of the object $k_{c,i}$, the locally rendered force in the last iteration $f_{e,i}$ and the device velocity $\hat{x}_{d,i}$ are used in the algorithm below to calculate a reduced stiffness at the client for the case of a coulomb friction contact of the touched object with one or more surface(s).

There is no dynamic friction rendered in the physics simulation if the object is in contact with another surface, but currently not moving ($\|\hat{x}_o\| = 0.0$). The proposed stiffness reduction in (20), however, is based on the resistive force during a constant movement of the object. Hence, at the initial contact, $f_e$ is not known directly. In this case, we estimate the expected resistive force as the normal force multiplied with the friction coefficient $\mu$ (line 7). The normal force is included in the received $f_e$, while the friction coefficient is set in the physics simulation and, thus, is known.
The current force magnitude is smaller than the static friction force. We return 0 to use the original stiffness if force is not acting against the user’s movement.

The resistive force is the static friction force.

Standard case: calculate the resistive force

Special case: the object is currently not moving!

Algorithm 1: Algorithm to calculate the stiffness at the client

```
Algorithm calculateObjectStiffnessAtClient(R, k_{ij}, \mathbf{x}_{d,i}, \mathbf{f}_{c,i}, S)
1: f_r ← calculateResistiveForceAtClient(\mathbf{x}_{d,i}, \mathbf{f}_{c,i}, S)
2: if f_r < 0 then k_i[t] ← k_{r,i} /* k_{r,i} calculated with (20) */
3: else k_i[t] ← k_{i,j} /* Use original stiffness if force is not acting against the user’s movement */
4: \bar{k}_i[t] ← \bar{\lambda} \cdot k_i[t] + (1 - \bar{\lambda}) \cdot \bar{k}_i[t - 1] /* Apply an exponential moving average filter */
5: return \bar{k}_i[t]

Procedure calculateResistiveForceAtClient(\mathbf{x}_{d,i}, \mathbf{f}_{c,i}, S)
/* Special case: the object is currently not moving! */
6: if ||\mathbf{x}_o|| === 0.0 then
7: \hspace{1em} f_r ← -\mu ||\mathbf{f}_i|| /* The resistive force is the static friction force */
8: /* The current force magnitude is smaller than static friction force. We return 0 to keep the original stiffness: */
9: if ||\mathbf{f}_i|| < ||\mathbf{f}_r|| then return 0
10: else f_r ← f_{r,i} /* Standard case: calculate the resistive force f_{r,i} with (19) */
11: return f_r
```

Note that the used physics simulation, ODE, does not distinguish between static and dynamic friction. It uses the same friction coefficient to implement static and dynamic friction in the virtual environment. If the object is not moving and the magnitude of the locally rendered force \( f_{r,i} \) is smaller than the estimated friction force (i.e. the static friction force is not exceeded) we use the original stiffness value \( k_{i,j} \) to calculate the interaction force (line 8).

If the object is moving, i.e., the client has received a non-zero object velocity \( ||\mathbf{x}_o|| \), the resistive force is directly calculated with (19) (line 9). The calculated resistive force \( f_r \) now represents the resisting force due to the frictional contact in the physics simulation and the desired stiffness \( k_{r,i} \) is calculated in line 2 with (20).

The desired stiffness \( k_{r,i} \) might, however, change drastically with time-varying delay and device velocity. This leads to perceivable artifacts and jumps in the rendered force feedback. To mitigate this effect, we use an exponential moving average filter on the calculated stiffness value to smoothly converge to the desired value (line 4). A similar filter is also used by Suzuki et al. [2014].

The stronger the low-pass characteristic of this filter, defined by the parameter \( \lambda \), the slower the client adapts to the new stiffness value. While slow stiffness updates are desirable to avoid sudden force jumps, they also lead to a mismatch between the desired stiffness \( k_{r,i} \) and the displayed stiffness. Hence, a trade-off between perceptually convincing and fast enough updates needs to be found. In our implementation, we found \( \lambda = 0.01 \) to be a reasonable value.

Note that we calculate a desired stiffness and update the displayed stiffness using the moving average filter in our implementation even if the delay change is below \( \Delta R_{\text{max}} \) to avoid large stiffness updates and to use a stiffness value that follows the desired stiffness value as close as possible.

The proposed stiffness adaptation scheme sometimes needs to increase the stiffness while the user is in contact with the object, e.g. due to communication delay changes. This artificially creates energy in the local force rendering, leading to a non-passive behavior and potential instability. The aforementioned filter leads to slow stiffness updates so that instabilities are avoided. Alternatively, a stability ensuring stiffness update method based on the concept of passivity as proposed in Xu et al. [2015] can be used.

6. EVALUATION AND DISCUSSION

This section presents objective and subjective evaluation results of the proposed scheme when used to compensate for the effect of communication delay in the client-server architecture.

We implemented the Jenga game shown in Figure 4 for this purpose. We chose this application because it necessitates precise interaction with the objects, constitutes different contact scenarios between objects, and is suitable for...
single point of contact haptic devices. The Geomagic touch haptic device is used at each client. The mass of each block is set to $m_o = 0.3$ kg. The coulomb friction coefficient in the physics simulation is set to $\mu = 0.45$, damping in the physics simulation is set to $c = 0.995$ and the original stiffness at the clients and the server is set to $k_{c,i} = k_{s,i} = 700$ N/m.

Both the client and the server run on the same machine and the network between them is simulated by software. This gives us precise control over the runtime conditions for the evaluation.

6.1 Simulated user inputs with constant communication delay

We simulate a user input with constant velocity in our SHVE setup to verify the introduced two-user client-server LTI model, the derived tolerable delay boundaries and the proposed delay compensation scheme.

The recorded forces for four constant delay values are plotted in Figure 5. The increasing steady-state force at the client for increasing RTT can be clearly observed for both clients. The Weber-fraction of $p = 10\%$, together with the reference force ($R_1 = R_2 = 0$ ms) defines the so-called area of perceived transparency. If the rendered force falls within this area, the user theoretically does not perceive the difference to the reference force. Note that we use a Weber-fraction of 10\%, which lies in the range of static force discrimination thresholds as discussed in Section 4. The calculated perceivable communication delay can be seen as conservative, because the Weber-fraction increases with increasing hand/arm velocity [Zadeh et al. 2007; Kammerl et al. 2010]. The estimated perceivable communication delay, calculated with (21) and $p = 10\%$, is $R_{1, \text{max}} = R_{2, \text{max}} = 18$ ms, which well predicts an increased steady state force magnitude of 10 percent. This can be clearly observed in Figure 5a, where the steady state force for $R_1 = 18$ ms falls on the border of the area of perceived transparency. For client 2 and $R_2 = 25$ ms in Figure 5b the steady state force is slightly above the area of perceived transparency.

The proposed stiffness adaptation approach is verified with the same simulated input. The rendered force values at the clients are plotted in Figure 5c and Figure 5d. The steady state force perfectly matches the zero-delay reference during the first time period for the constant RTT until the delay increases at around $t = 2.8$ s. The predicted perceivable
delay change with (22) is between 20 ms and 28 ms for the tested round-trip times. Hence, the delay increase of 20 ms should not lead to an increased force magnitude that exceeds the area of transparency. The simulation results for $t > 2.8$ s confirm this theoretical results.

6.2 Human user inputs with time-varying communication delay

The previously presented simulation results show that the proposed scheme perfectly compensates the effect of delay on the steady state forces rendered at the clients. The human-in-the-loop, however, has not been considered so far. Additionally, time-varying delay has been considered only as a single delay change in the previously presented simulations. In the following, we present force signals recorded during different real user interactions to validate the proposed algorithm in the presence of constant and time-varying delay.

With the human-in-the-loop it is not possible to repeat exactly the same interaction several times. Hence, a sample-by-sample comparison of the displayed force is not feasible. In this experiment, we tried to be as consistent during the interaction as possible to allow for a qualitative comparison.

Time-varying communication delay is simulated as a piecewise constant function. Both the forward and the backward channel consist of a deterministic delay (e.g. propagation delay). On top of this, additional stochastic delay (e.g. additional queuing delay) is simulated. The choice of a piecewise constant function is motivated by the fact that packet-reordering in Internet UDP traffic is rare [Tinta et al. 2009]. Buffers and playout-scheduling can always be used to rearrange out-of-order packets to some extent [Hikichi et al. 2002; Liang et al. 2003], leading to a piecewise constant playout delay. The stochastic delay has a gamma distribution, which describes common end-to-end delay in Internet communication [Bovy et al. 2002].

Cooperative pushing (see (a) in Figure 4). The recorded signals in Figure 6 show that the proposed compensation scheme successfully compensates for the effect of constant communication delay on the force magnitude for the exemplary interaction already shown in Figure 2. The profile of the force magnitude during the interaction with zero communication delay ((1) in Figure 6) is very similar to the one in the presence of delay and compensation enabled ((3) in Figure 6). Without compensation, the magnitude of the rendered forces is clearly increased for both users ((2) in Figure 6).

Pushing and pressing (see (b) in Figure 4). The signals in Figure 7 are recorded while client 1 pushes a single block. During the movement, client 2 presses on the top surface of the block, which leads to an increased normal force and, hence, an increased friction force simulated in the physics simulation. The increased friction force is also perceived by client 1. As the contact forces are included in the object state updates transmitted to the clients (see Section 5), the client 1 can adapt accordingly. Again, the force signals recorded with simulated delay and compensation enabled ((3) in Figure 7) are very similar to the reference ((1) in Figure 7). The communication delay has no effect on client 2. He presses on the top surface and, hence, the stiffness is not reduced due to the motion constraint (see Section 5).

Pushing from opposite sides (see (c) in Figure 4). The signals in Figure 8 are recorded while client 1 pushes a single block. After some time, client 2 presses from the opposite side. The object comes to rest as both forces are counteracting and have equal magnitudes. In this special case of the object not moving, the communication delay has no effect on the force magnitudes as one can observe from Figure 8. Note that one assumption of the derivations in Section 4 and Section 5 was that the object is moving in a steady state. The proposed scheme, however, also inherently captures such a scenario (see (3) in Figure 8).

6.3 Subjective evaluation

The delay compensation scheme is based on a steady state analysis. Hence, it is not expected that the force rendered at the clients always perfectly matches the force without communication delay. Furthermore, the subjects interact with an object with reduced stiffness if the proposed delay compensation is enabled. We perform a subjective test to assess if the proposed scheme can achieve perceptual transparency, although its derivation is based on the steady state only. In
other words, we want to test if the users can distinguish between a non-delayed interaction with the virtual environment and an interaction in the presence of time-varying communication delay in the CS architecture.

6.3.1 Experimental setup and procedure. We adopt the same-different procedure [Gescheider 1997] to assess if subjects can discriminate between the interaction in:

(1) the non-delayed reference case (stimulus $S_R$, R: Reference)
(2) the presence of time-varying delay without delay compensation (stimulus $S_{NC}$, NC: no compensation)
(3) the presence of time-varying delay with delay compensation enabled (stimulus $S_{WC}$, WC: with compensation)

For this experiment, a single user has to push blocks one by one from a stack of blocks (see Figure 9). This task necessitates the initial acceleration of the blocks, i.e., it includes a transient state. Hence, it is suitable to evaluate if the analysis based on the steady state limits the applicability of the proposed approach.

In a single trial, the subject compared the two top blocks (i.e. a stimuli pair $S_R-S_R$, $S_R-S_{NC}$ or $S_R-S_{WC}$) by answering the following two questions: 1) “Did you perceive the blocks to be of the same or different weight?” and 2) “Did you perceive any other artifacts? Yes or No?”. All responses were written down by the experimenter. In case other artifacts were perceived, the subject was asked to explain them verbally. This additional question was posed in order to evaluate...
Compensating the Effect of Communication Delay in Client-Server-based Shared Haptic Virtual Environments

Fig. 9. Left: The experimental setup, including a zoomed in version of the VE used for the subjective evaluation. Right: histogram of the time-varying RTT at the client during the experiment. It can be seen that it follows approximately a gamma distribution.

<table>
<thead>
<tr>
<th>Table I. Answers summed over all subjects.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stimuli pair</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>$S_R - S_R$</td>
</tr>
<tr>
<td>$S_R - S_{NC}$</td>
</tr>
<tr>
<td>$S_R - S_{WC}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table II. Calculated $d'$ values for the individual subjects, as well as the overall mean and standard deviation between them.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stimuli pair</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>$d'_{NC}$</td>
</tr>
<tr>
<td>$d'_{WC}$</td>
</tr>
</tbody>
</table>

if there is any artifact, which was consistently perceived by the subjects. We chose to perform the subjective evaluation with a single user only, because the cooperative interaction of two users would lead to uncontrollable conditions during the experiment. A subject cannot answer the questions reliably, because the difference in the perceived weight compared to the previously tested block could be a result of an increased/decreased force applied by the other client during the cooperative interaction or by the delay simulation and compensation being enabled or disabled.

The order of the stimuli within a pair and the order of the pairs was randomly selected. The camera in the virtual environment and the workspace of the device had been rotated after each stimuli pair so that the user movement was always in the same direction. After nine comparisons were finished, the tower was rebuilt and the subject started again from the top. Each subject compared 72 pairs in total (24 per stimuli pair). A break was enforced after half of the comparisons.

15 voluntary subjects participated in the experiment (13 male, 2 female, aged between 22 and 45). Four were frequent users of haptic devices, six had already participated in former experiments in our lab, and the others were novice users. The procedure was first verbally explained and demonstrated by the experimenter. Then, a training session with at least 18 comparisons was conducted with each subject. During the training, the subject was informed that the object weight varied, but also pairs with the same weight were included. Note that the weight was actually not changed between the stimuli. Only the delay simulation and the delay compensation was enabled or disabled. As a consequence, the force feedback increased eventually, which was perceived as increased weight.

The delay histogram is plotted in Figure 9 to illustrate the delay simulation during the subjective experiment.

6.3.2 Experimental results. The same-different answers summed over all subjects are shown in Table I. The answers of each subject are analyzed using signal detection theory (SDT), to achieve robustness towards uncertainties in human decision making and response bias [Macmillan and Creelman 1991]. We assume the differencing strategy as the subjects’ decision rule [Macmillan and Creelman 1991], because several pairs of different stimuli were compared [Gescheider 1997].

The $d'$ value, which is a measure of the subjects’ sensitivity to the difference between two stimuli, is determined for the comparison pair $S_R-S_{NC}$ ($d'_{NC}$) and for the pair $S_R-S_{WC}$ ($d'_{WC}$) and given in Table II. Note that both $d'$ values are calculated with respect to the reference pair $S_R-S_R$. The $d'$ values are determined using the table given in Macmil-
ian and Creelman [1991]. Higher values of $d'$ indicate higher discriminability. Two stimuli are often assumed to be indistinguishable for a threshold value of $d' \leq 1.0$ [Gescheider 1997]. The averaged sensitivity indices are calculated as $d'_{NC} = 4.36 \pm 0.67$ and $d'_{WC} = 0.97 \pm 0.59$. According to the chi-squared goodness of fit test the data is normally distributed. Hence, a two-sample t-test (significance level of 5%) is performed and a statistically significant difference between the $d'_{NC}$ and $d'_{WC}$ group means can be reported ($t(28) = 14.66$, $p < 0.001$).

With respect to the second question, the subjects reported that they did not feel any other artifact for most of the trials. Subjects sometimes reported some kind of “discontinuities” or “jumps” in the force feedback for stimuli $S_{NC}$ or $S_{WC}$, which could originate from large delay changes as discussed in the following. Three subjects reported that stimuli $S_{NC}$ sometimes were of “varying weight”, which we attribute to the time-varying delay.

6.4 Discussion

The presented recorded force signals and the results of the subjective evaluation clearly show the effectiveness of the proposed delay compensation scheme. In the subjective evaluation, the subjects could barely distinguish the weight of the objects in the comparison of the stimuli without delay ($S_R$) and the stimuli with time-varying delay and delay compensation enabled ($S_{WC}$).

But still, $S_R$-$S_{WC}$ pairs were less often rated to be the same as $S_R$-$S_R$ pairs. An analysis of the signals recorded during the experiment did not reveal an obvious reason for this. Besides general uncertainty in human decision making, we suspect that a large delay change could be a main reason. The smooth stiffness adaptation (see Section 5.3) might lead to a discrepancy between desired and actually displayed stiffness. Hence, investigating methods to smoothly update the stiffness are an interesting direction for future work. An optimal trade-off between fast but smooth stiffness adaptation needs to be found.

The artifacts experienced by the users, which they described as “discontinuities” or “jumps”, stem from large delay changes. Remember that we keep the transmission rate of state updates at the rate of the haptic loop of 1 kHz in this paper. If the delay increases on the forward or the backward channel, the server or client won’t receive new state updates for the amount of time the delay has increased. In this case, we hold the last received state until the next update is received. If the delay decreases, the server or client will have several state updates in their receiving buffer. In this case, we do not display the latest update directly. Instead, we smoothly converge to the latest received state to avoid big jumps in the object/device state. Large delay changes might still lead to abrupt changes in the force feedback. Interestingly, 76.9% of the “discontinuity”/“jump” answers were given for $S_{NC}$ trials ($S_{WC}$: 23.1%). This suggests that the reduced stiffness during the $S_{WC}$ trials helps to hide the delay changes. The effects of large delay changes, especially for higher stiffness values, and how to efficiently hide them to the user need to be investigated in more detail. Buffering at the client [Ishibashi et al. 2002] can be considered to avoid the aforementioned effects of delay changes. This comes, however, at the cost of an increased overall delay, but the proposed compensation scheme then does not need to adapt the stiffness to delay changes. On the other hand, a small playout delay should be always preferred as large delay leads to reduced operability [Matsumotoy et al. 2000].

No subject reported the impression of interacting with a softer object as an artifact during the interaction. At the beginning of the contact, the original stiffness is used. The stiffness is reduced only after the rendered force exceeds the static friction force. This might be the reason that the reduced stiffness is well hidden from the user.

In summary, we conclude that the proposed compensation scheme does not introduce obvious artifacts that are consistently perceived by the subjects, while at the same time being effective in hiding the effect of communication delay to the user, although the derivation of the scheme is based on the steady state only.

7. CONCLUSION

This work investigates transparency for client-server-based SHVEs with a centralized rigid body physics simulation.

The centralized physics engine on the server leads to the artifact of objects not moving immediately at the initial contact. This leads to increased forces rendered at the client. Based on the presented model, we derive formulas to estimate the perceivable delay and the perceivable delay change. The proposed scheme uses a reduced stiffness to
compensate the force error displayed to the user in the presence of communication delay. The objective and subjective evaluation shows that the proposed scheme effectively compensates for time-varying delay in such a CS architecture during cooperative multi-user interactions.

With the proposed compensation scheme, the object still does not move immediately at the initial contact with an object. For large delay values, this becomes noticeable haptically and visually and the operability of the system reduces. As a consequence, a local physics simulation becomes necessary. A plausible, but not accurate physics simulation could be applied at the client as an alternative way to compensate for the delay in the CS architecture. This is an interesting direction for future work, although it means a fundamental change in the architecture investigated in this paper.

REFERENCES


Clemens Schuwerk et al.


Received July 2015; revised September 2015; accepted October 2015