Probal Mitra
Günter Niemeyer
Telerobotics Lab
Stanford University,
Stanford, CA 94305, USA
{probal,gunter.niemeyer}@stanford.edu

Model-mediated Telemanipulation

Abstract

This paper presents a user-centered, model-mediated approach to bilateral telemanipulation under large communication delays. Assuming that the environment is only slowly or infrequently changing, it mitigates user perception difficulties and allows stable motion and force interactions with the remote environment. Rather than directly sending slave sensory data to the user, the method abstracts the data to form a very simple model of the environment. The model is transmitted to the master, where it is haptically rendered for user feedback without any lag. In return, the slave only executes force or motion commands consistent with the model. This effectively mediates the master–slave interaction. Particular attention is placed on the system behavior as it adjusts to unexpected environment interactions and as the model updates. The basic principles of the approach are demonstrated on a simple one degree-of-freedom telerobotic system operating with four seconds of round-trip delay.

KEY WORDS—teleoperation, telerobotics, telemanipulation, haptics, model mediation, time delay, force feedback, augmented reality

1. Introduction

The goal of bilateral telemanipulation is to allow a human user to manipulate and interact with a remote environment by means of master and slave robotic devices. Successful manipulation inherently requires some form of perception of the remote site and the response to the requested actions, necessitating closed-loop interaction. In addition, manipulation implicitly includes physical contact at the environment with the application of forces in addition to simple motion tracking. One assumes, therefore, the use of a haptic user interface via the master robot so that the user may intuitively command motion and forces to the remote slave robot. Haptic (force) feedback is combined with standard visual feedback to raise the user's sense of presence at the remote location and support the manipulation task.

The presence of substantial communication delays, however, will deteriorate both the stability and performance of a system based on force and motion exchange. Even when appropriately modified for stability, the lag in feedback distorts the user's perceptions and delays in commands challenge the ability to manipulate the environment.

We present here a model-mediated telemanipulation method, targeted at systems with delays ranging from hundreds of milliseconds to several seconds. It provides stable slave tracking as well as an intuitive haptic user interface with immediate feedback. It uses some abstraction in the data transmission without necessitating the high level of slave intelligence needed in supervisory control methods. In comparison to predictive displays modeling slave motion, the approach further uses rudimentary models of the environment for motion and force prediction.

The method, first introduced by Mitra and Niemeyer (2006) and depicted in Figure 1, observes the motion and forces at the user/master interface and transmits these commands to the slave. Both signals are necessary for the slave to perform appropriately as it interacts with the unknown environment. At the slave side, the robot tracks the incoming commands while simultaneously making use of sensor data (typically position sensing, but possibly also force, contact or video information) to construct and update an appropriately simplified model of its environment. As this model is learned, it is returned to the master with a delay. The model information may be as simple as a contact location or as complex as estimates of the shape, stiffness and dynamic properties of objects. The complexity should be correlated to the delay time, with its predictive horizon matching the delay. Indeed, when the model information is received at the master it is used to recreate a haptic simulation of the slave environment, effectively anticipating interactions for the user. The user is thus provided with an intuitive force-feedback interface which, because of the virtual model,

The International Journal of Robotics Research
Vol. 27, No. 2, February 2008, pp. 253–262
DOI: 10.1177/0278364907084590
©SAGE Publications 2008 Los Angeles, London, New Delhi and Singapore
Figures 1, 2, 7–10 appear in color online: http://ijr.sagepub.com

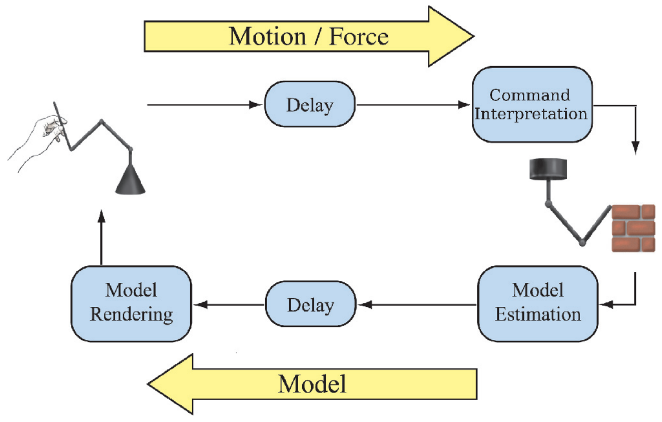

Fig. 1. The proposed telemanipulation scheme extracts a highly simplified environment model from the slave interactions and recreates the model virtually for user interactions. Simultaneously, user commands are processed before application at the remote site for consistency with the model.

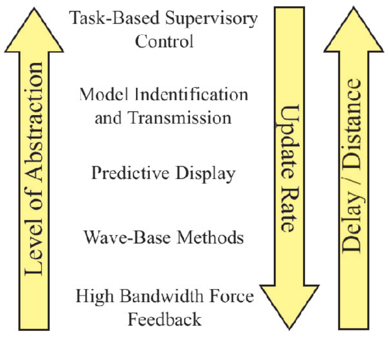
Task-Based Supervisory Control
Model Indentification and Transmission
Predictive Display
Wave-Base Methods
High Bandwidth Force Feedback
Level of Abstraction
Update Rate
Delay / Distance

Fig. 2. Different forms of interaction and feedback trade-off the level of abstraction and data complexity with update rates and robustness to delays.

allows them to generate relevant commands without feeling any delay-related damping or unnatural lag between action and reaction in the environment.

From a manipulator control perspective, the local model helps the user to provide an appropriate choice between motion and force commands, which the slave further checks in order to match it to the latest environment model. Model mediation can assist both the user's perception as well as manipulation capabilities.

In this paper we argue that for successful teleoperation, the level of delay must be matched by a level of abstraction in the communication. With reference to Figure 2, and as discussed in Section 2, supervisory control provides a high abstraction and low update rate. In contrast, direct position/force data communication and feedback will inherently be limited to systems with tens of milliseconds of delay or less.

This work begins an exploration and determination of what level of data abstraction and rate of learning are necessary to overcome communication delays of the order of seconds. It is motivated by the observation that environments are often fairly static, evolving or moving slowly or infrequently. Even when objects are moving, the motion often follows particular patterns or is due to the robot's manipulation actions. This allows a model to be built and updated fairly slowly, with time constants of the order of the delay time. As the environment changes slowly and the model information passes through the delay, accurate but potentially time-consuming model-update algorithms can be used, while high update rates are used at the master for accurate local rendering. Relatively simple changes can be made to the slave controller to handle discrepancies between incoming motion/force commands and the actual environment, for example when forces are commanded without actual contact. This achieves stable closed-loop operation.

The principles of this telemanipulation technique are developed here in the context of a one degree-of-freedom (DOF) system using only vertical motion. The slave is either in free space or encounters a rigid floor at an unknown height. We show how even a simple model-generation method, based only on detecting the floor location, can succeed in producing stable behavior and useful haptic feedback to the user under several seconds of communication delay.

The paper is presented as follows: the next section considers existing teleoperation methods for various levels of delay and types of environment. In Section 3 the proposed control architecture is developed, and in Section 4 the details of a sample one DOF implementation are presented. Experimental results from an actual master–slave system based on this sample are given in Section 5. The paper concludes with a discussion of future avenues of research for model-mediated telemanipulation and a summary of the results obtained thus far.

2. Background

Teleoperation has experienced a rich history, producing a wide variety of control techniques, for different applications, environments and varying levels of communication delay. The nature of the remote environment and the level of communication delay between the master and slave sites determine the form and implementation of information interchange, particularly in terms of user commands and feedback.

In situations involving very little or no communication delay, direct communication of position and force information can in principle recreate the experience of direct contact and achieve telepresence. Generalized in "four-channel" systems (Lawrence 1993), these approaches provide users with accurate reproduction of the forces and impedances experienced by the slave in the remote environment. In practice, device, amplifier, sensor and computer characteristics limit performance, but basic bilateral teleoperator systems can achieve stable operation with little or no prior knowledge of the remote environment (Lawrence 1993; Daniel and McAree 1998; Hashtrudi-Zaad and Salcudean 2000).

When teleoperating a robot over even small to medium delays (i.e. tens to hundreds of milliseconds), below or near the human reaction time, basic force feedback becomes impractical and unstable (Hashtrudi-Zaad and Salcudean 2000). Advanced controllers have been devised, based on maintaining passivity under delay (Anderson and Spong 1992) and wave variable encoding of the force and motion information (Niemeyer and Slotine 2004; Ching and Book 2006; Tanner and Niemeyer 2006). Indeed these methods can restore some level of telepresence, but are still useful only when operating under delays of a second or less.

As the delay in the communication channel increases, such as in space or underwater telerobotics, actions and reactions appear out of phase and users adopt move-and-wait strategies, relying mainly on visual feedback (Sheridan 1993). The force–motion feedback approach must be replaced by more abstract control algorithms in which greater autonomy is passed on to the slave robot (Sheridan 1992, 1993). For example, a mobile robot can be teleoperated by providing pre-planned trajectories which are then modified by the slave in order to avoid collisions with unexpected obstacles (Makiishi and Noborio 1999; Chong et al. 2001; Tzafestas 2001). Control methods based on slightly more abstract information are also used, such as the joystick- and voice-based "telecommanding" approach of Wang and Liu (2004). Further still, for extremely large delays and greater environment uncertainty, "supervisory control" methods can be applied in which the slave robot operates almost entirely autonomously with only high-level task instructions sent at sparse time intervals (see, e.g., Gat et al. (1994) and Schooley et al. (1993)).

With increasing levels of information abstraction, however, there is less information available for feedback to the user, making it difficult to provide them with an intuitive interface for commanding the slave. For example, in Makiishi and Noborio (1999) and Wang and Liu (2004), users provide target locations to slave robots based only on delayed maps or images of the objects and robots in the remote environment.

Some control methods, such as those of Chong et al. (2001), Gu et al. (2002) and Tzafestas (2001), make use of rough environment models to create a virtual simulation interface with which the user can interact in a more intuitive manner. Bejczy et al. (1990) and Ching and Book (2006) used predictors to simulate expected slave responses to enhance the quality of visual and/or force feedback to the user. This combination of simulation and teleoperation, or "augmented teleoperation", has also been investigated outside the context of time-delayed systems. For example, virtual reconstructions of the slave environment can be overlayed on video feedback when image quality is poor, such as in systems with low-resolution cameras (Casals et al. 2002), or underwater environments (Utsumi et al. 2002). Alternatively, augmented interfaces can be used to restrict or refine user commands, by using force feedback to discourage users from commanding the slave into dangerous parts of the workspace (Turro et al. 2001) or singular configurations (Maneewarn and Hannaford 1999).

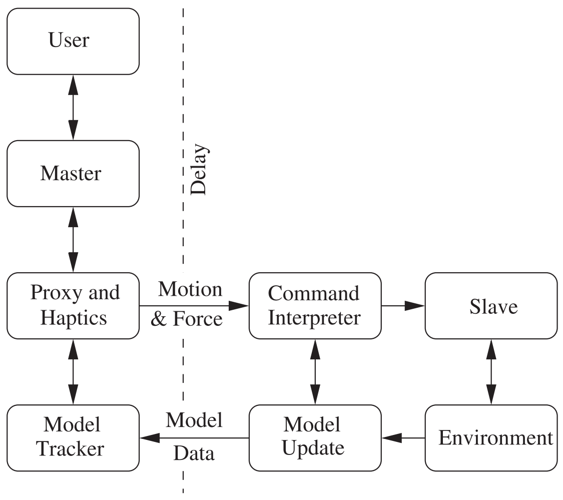

Fig. 3. The proposed algorithm uses a proxy and haptic model at the master site that mirror the slave and environment model.

3. Model Mediation

The proposed teleoperation scheme provides an increased level of data abstraction, as compared with traditional position or force-feedback loops. Instead of feeding back continuous position or force measurements, it extracts and transmits an environment model from these measurements and/or other sensor data. The user interacts locally with the haptically rendered model. Depicted as a general concept in Figure 1, we implicitly assume that the environment changes only slowly or infrequently by itself. The slave may move quickly and even alter the environment, if this can be captured in the model.

A more detailed view is given in Figure 3. The vertical layers represent the conceptual separation between user and environment. The user holds the master which by control is connected to the proxy. The proxy itself acts as a local stand-in for the slave robot. If rendered graphically as well as haptically, it also functions as a predictive display. Ultimately, a dynamic proxy with appropriate motion constraints may be used to capture the slave dynamics and workspace. The local model represents the actual environment.

Along the horizontal direction, Figure 3 illustrates the connections between the user's local components and those in the remote location. Both the user's force and motion are observed and commanded to the slave, where a special tracker controls the slave. In the other direction, the environment is estimated and the model is transmitted and tracked as accurately as possible. In the following we discuss the three novel components: the slave control and command interpreter, environment model and estimation, and the model tracking.

3.1. Slave Control and Command Interpretation

As previously noted, it is the aim of any bilateral telemanipulation scheme to allow the user to interact with and manipulate the environment. As such, the slave must execute both motion and force commands. When the model at the master closely matches the actual environment, the master's motion and forces will be consistent with the remote environment. Both commands can be achieved simultaneously, e.g. using simple PD and feed-forward control.

Under large delays, however, the commands may be inconsistent with the environment. For example, imagine the master model predicting contact and the user commanding a contact force. If the model is inaccurate or has not yet been updated owing to the delay, the actual environment may not present the expected contact. With the slave in free space, the commanded force would be impossible to achieve.

The slave controller must synthesize an achievable command, consistent with the encountered environment. In general, this should be a "safe" combination, such that motions are only executed without resistance or excess forces, while forces are only applied against contact. By construction, the master model will track the environment, so that the command inconsistencies should only last approximately one delay cycle. As such, the slave needs only to temporarily apply "safety" measures when tracking commands.

Indeed one might interpret the model rendition at the master as allowing the user to input motion or force commands as necessary. Equivalently, the model should inform the slave controller and command interpreter as to which commands to track. This is, in essence, a form of automated selection of force versus motion control.

3.2. Environment Model and Estimation

The commands sent by the user are based on interactions with the master's virtual model of the slave environment. The quality of these commands will, therefore, necessarily depend on the accuracy of the model displayed to the user. In turn, this depends on the estimation carried out by the slave.

Models of varying complexity may be used and can provide different levels of data abstraction. In its simplest form, a model should distinguish between free space and contact. Beyond this, a model might include the following:

1. *Geometric shapes*: this would estimate virtual surfaces to match the geometric structure of the environment.
2. *Material/dynamic properties*: these include surface stiffness, damping, friction and other properties.
3. *Moving objects and the motion constraints*: these would be included, in particular, if the environment contains distinct pieces.

Fundamentally, the model need only be accurate for the duration of one delay cycle, after which it can be updated again with new data. Furthermore, prior knowledge of the environment, such as the existence of objects and known geometries, may restrict general models and greatly simplify the estimation. The choice of sensors will also influence model quality, e.g. video, sonar or radar for estimating geometric properties and proximity sensors, force sensors and accelerometers for contact detection.

The complexity of the model, i.e. the level of detail to which each type of information is estimated, may also be chosen from the application at hand. For example, for a rigid peg-in-hole type task it may be assumed that all surfaces have nearly infinite stiffness, so estimation would concentrate on geometric and kinematic characteristics. On the other hand, if the slave environment contains soft, highly compliant objects, knowing the dynamic parameters is more important than extremely accurate geometric measurements.

Finally, it should also be noted that, unlike a tracking controller or haptic display, the model estimation algorithm does not have to run at high update rates. Indeed, this scheme is based on the notion that the *rate of model change* is relatively slow compared with the user's speed of motion or force detection bandwidth. In fact, it may be preferable to receive greater model detail at a slower rate than to have fast but inaccurate estimates.

3.3. Model Tracking

Updated models of the remote environment are sent through delayed communication to the master, where the information is used to generate a virtual world with which the user can directly interact. As the parameters in the master model track those estimated by the slave, care must be taken to introduce the updated values into the user's virtual world without causing instability in the haptic simulation or discomfort to the user.

The stability of haptic rendering may be measured and ensured by examining the energy associated with the proxy and master controller. Repositioning virtual constraints and the proxy may increase the potential energy of the PD controller, if it stretches the tracking error without any corresponding master movement. As such, virtual constraints may need to be introduced gradually, as discussed in the case study presented in Section 4.

Changing constraints can also have an effect on user comfort, by causing a discrepancy between the virtual display and the user's expectations. As constraints are updated, transition

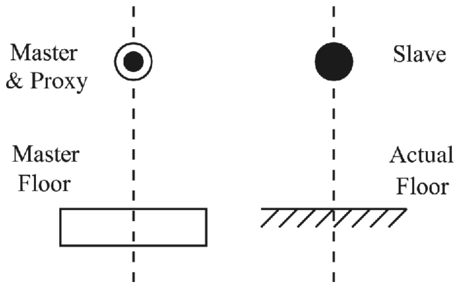

Fig. 4. The one DOF system creates a virtual floor for the master device in coordination with the estimated location of an environment floor.

periods can be used during which some level of error is tolerated between the local and remote models. This allows users to become safely and intuitively accustomed to the updated environment. Transitions may be done transparently, so as to hide changes as much as possible, e.g. by fading in forces and images. Alternatively they may explicitly draw attention to the new constraints, e.g. by giving the user a short force impulse or displaying blinking graphics. The choice of transition method will depend on users' preferences and their ability to adapt to changing environment models; this specific topic has been further investigated by Mitra et al. (2007).

4. Single DOF Case Study

To demonstrate the potential of this teleoperation approach, consider a sample implementation involving a one DOF master–slave telemanipulation setup. Only vertical motions and forces are considered, and the environment model only describes a rigid floor at an unknown location (see Figure 4). The master sends force and motion commands, which are tracked by the slave robot. Meanwhile, the slave robot updates its model of the environment by detecting contact with the floor and recording the location of the contact. The master model then generates a haptic virtual floor based on the floor location estimate sent by the slave. The user is thus able to directly "touch" the floor without feeling either the time delay or the inertia of the slave.

The individual elements that compose the entire architecture are detailed in the following sections.

4.1. Master-side Controller

The master device only exerts forces on the user if they have penetrated the virtual floor. A dynamic haptic proxy (Niemeyer and Mitra 2004) is used, and master forces are generated based on PD tracking of the proxy:

$$F_m = k_{pm}(x_p - x_m) + k_{vm}(v_p - v_m), \tag{1}$$

where F_m is the force exerted by the master, x_p, x_m, v_p, and v_m are the proxy and master positions and velocities respectively. The PD gains k_{pm} and k_{vm} correspond to the master position and velocity gains. They are tuned to produce as high a stiffness as can be achieved by the master device, so as to recreate the high stiffness of an actual rigid floor.

4.2. Master Proxy Motion

The master proxy embodies a representation of the slave mechanism. Like the slave, it should track the master motion when in free space and stop at the object boundaries when in contact. As such, it interfaces the master's physical and observed motion with the virtual floor.

A dynamic proxy is used for several reasons. First, giving the proxy its own velocity allows for finer control during the creation and removal of contact constraints, as required in the model update described below. Further, a dynamic proxy can be used to portray slave mechanical limits in both position and maximum speed to the user. This implements a virtual slave (Mitra and Niemeyer 2004) such that motions commanded by the user are consistent with the slave capabilities.

The dynamic proxy motion is governed by

$$\dot{x}_p = \dot{x}_m + \lambda(x_m - x_p) \tag{2}$$

subject to

$$x_p \geq x_{mfloor}, \tag{3}$$

where λ is a time constant for the first-order dynamics and x_{mfloor} is the location of the master's virtual floor.

4.3. Master Model Update

To reproduce the environment accurately, the location of the master virtual floor must track that of the estimated floor in the slave environment. The simplest implementation would be

$$x_{mfloor} = x_{sfloor}, \tag{4}$$

where x_{sfloor} is the most recently received slave estimate of the floor location.

As discussed in Section 3.3, such an approach could potentially result in abrupt, destabilizing and potentially confusing forces being exerted on the user. Instead, in this implementation we chose to only track the new floor location subject to a passivity constraint

$$x_{mfloor} = x_{sfloor} \tag{5}$$

subject to

$$x_{mfloor} \leq x_p. \tag{6}$$

The constraint ensures that the virtual floor is never pulled above the proxy level. This avoids unexpected upward forces

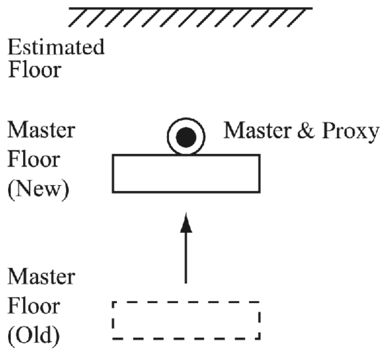

Fig. 5. When the floor estimate is above the master, the virtual floor is implemented as high as possible without making the proxy jump or injecting energy into the system.

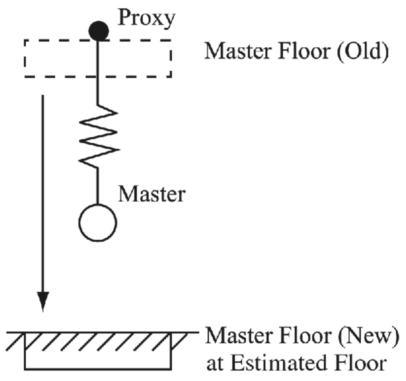

Fig. 6. Upon removal of a floor, the proxy gradually and gently converges to the master location to avoid abrupt force cues.

on the user and prevents spring-potential energy from being injected into the haptic simulation.

By ensuring a passive response, the stability of the haptic interface is thus guaranteed. With the floor just at the proxy level, downward motion is blocked. During upward motion, the model tracking slides the floor along with the proxy so that the user appears to remain in contact until the correct location is reached.

An example of when this could happen is shown in Figure 5. If the slave detects a contact above the location of the master and/or proxy, the master floor is only raised to the proxy level.

In the dual situation, if the slave detects that the floor is no longer present and the master floor is removed, the dynamic proxy gradually moves to the master location. From (2), the first-order dynamics smooth the proxy motion and thus the force applied to the user, while still maintaining the basic constraint that the proxy never penetrates the virtual floor. This is depicted in Figure 6.

In practice, the constraint based model update is implemented in a simplified, discrete form:

$$x_{\text{mfloor}}[n] = \min(x_{\text{sfloor}}[n], x_{\text{p}}[n-1]), \tag{7}$$

$$x_{\text{p}}[n] = \max(x_{\text{mfloor}}[n], x_{\text{p}}[n-1] + \Delta T \dot{x}_{\text{p}}), \tag{8}$$

where n represents the current time step and ΔT is the servo-period.

4.4. Slave-side Controller and Command Interpreter

The goal of the slave controller is to reproduce the user's intended motion and force trajectory as accurately as possible, subject to constraints from the environment. This requires the slave to interpret the incoming master commands and first determine whether or not they are safe to track. If they are not, the slave must adjust until acceptable commands are received.

Assuming a force-controlled slave, the incoming master motions are tracked using a standard PD controller:

$$F_{\text{pd}} = k_{\text{ps}}(x_{\text{p}} - x_{\text{s}}) + k_{\text{vs}}(v_{\text{p}} - v_{\text{s}}), \tag{9}$$

where F_{pd} is the PD force command, k_{ps} and k_{vs} are the slave position and velocity tracking gains, respectively, and x_{s} and v_{s} are the slave position and velocity, respectively.

Meanwhile, to assure that the force commands are also executed, the PD force is combined with the master force command

$$F_{\text{des}} = F_{\text{pd}} - F_{\text{m}}, \tag{10}$$

where F_{des} is the force command given to the slave robot and F_{m} is the force exerted by the master on the user. The master force F_{m} is inverted to command the force exerted by the user on the master.

Two things should be noted here. First, the gains need not be the same as (or even related to) those chosen in the master controller; in fact, they should be tuned according to the slave robot's dynamic abilities and inertial and friction properties rather than those of the master. Second, the slave is tracking the proxy motion. Thus, when the master reaches contact, force commands are generated from the master force and not via penetration into the virtual wall.

When both master and slave are in free space, the scheme produces $F_{\text{m}} = 0$ so that $F_{\text{des}} = F_{\text{pd}}$, i.e. the slave uses PD tracking. If the master then comes into contact with its virtual floor, the PD tracking continues with an additional force $-F_{\text{m}}$ until the slave also comes into contact with the floor. Once both robots are in contact, that is $x_{\text{p}} = x_{\text{s}}$, the slave controller directly tracks the force exerted by the user on the master, i.e. $F_{\text{des}} = -F_{\text{m}}$, with no PD term.

Finally, if the slave detects contact of which the master is unaware, the proxy may penetrate this constraint and the PD tracking force could become large. To safeguard against unintended force applications, the PD tracking force F_{pd} is limited to a small threshold force F_{thresh}, in the downward direction when the slave detects contact:

$$F_{\mathrm{des}} = F^{*}_{\mathrm{pd}} - F_{\mathrm{m}} \quad \text{with } F^{*}_{\mathrm{pd}} = \max(F_{\mathrm{pd}}, -F_{\mathrm{thresh}}). \tag{11}$$

The choice of F_{thresh} will depend on the slave model updating scheme, discussed in the following.

4.5. Slave-side Model Update

The slave environment consists of a rigid floor at an unspecified location. With only one DOF and the assumption of rigidity, this means that the only environment information which must be estimated is the floor location x_{sfloor}. When no floor is detected, the estimate is set to a large negative number, placing it below the workspace of either robot.

The estimated floor location can be updated in two steps: detection of contact between the slave and the real floor, and determination of the contact location. The first step (detecting the contact event) can be implemented relatively simply, depending on the nature of the slave robot. For a non-backdriveable force-controlled robot (such as the PUMA 560 of Park and Khatib (2006)), measurements from a force sensor can be used to determine contact:

$$\begin{aligned} \text{if:} \quad & |F_{\mathrm{meas}}| > F_{\mathrm{thresh}} \\ \text{then:} \quad & \mathit{contact} = \mathit{true} \\ \text{else:} \quad & \mathit{contact} = \mathit{false} \end{aligned} \tag{12}$$

where F_{meas} is the force measured by the sensor and F_{thresh} is a threshold force determined by sensor noise and disturbances. Typically, this threshold will be determined by trial and error, and chosen to be as small as possible without causing false contact detections. If the slave robot is small and light (such as SensAble Technologies' PHANToM™) there may be no force sensor available at the tip; in this case the PD force command and motion of the slave can be used as indicators:

$$\begin{aligned} \text{if:} \quad & |F_{\mathrm{pd}}| > F_{\mathrm{thresh}} \text{ and } |v_{\mathrm{s}}| = 0 \\ \text{then:} \quad & \mathit{contact} = \mathit{true} \\ \text{else:} \quad & \mathit{contact} = \mathit{false} \end{aligned} \tag{13}$$

so that a steady-state tracking error is interpreted as a contact. Proximity sensors, as well as other contact or non-contact sensors, may also provide valuable information if they are available.

Having detected a contact event, the estimated floor location is set to the current location. When no contact is detected, the algorithm must decide whether the lack of contact agrees with the existing model or if the rigid floor is no longer near the previous location. The former would occur when the slave robot is actually above the rigid floor. The latter case arises when the slave moves past the estimated floor location without encountering a contact event. If this penetration depth exceeds a minimum error threshold x_{thresh}, the model updating scheme assumes the floor location estimate is incorrect and removes the floor.

By a careful choice of x_{thresh}, a symmetry can be established between the threshold force required to *create* the floor estimate (F_{thresh}) and the threshold distance required to *remove* the floor estimate. Suppose that x_{thresh} is chosen according to

$$x_{\mathrm{thresh}} = \frac{F_{\mathrm{thresh}}}{k_{\mathrm{p}}}. \tag{14}$$

Given the PD tracking force described earlier (Equation (9)), this choice means that if the master and slave are stationary then the slave floor estimate will be removed when the PD tracking force exceeds F_{thresh}, in the same way that the floor estimate was created when the contact force exceeded F_{thresh}.

5. Experimental Results

The above implementation was tested on a pair of master and slave PHANToM™ robots. The performance was measured under different conditions and the results are presented here.

The slave environment consisted of a rigid table (the "floor") which could be removed, and for both master and slave robots only vertical motion was considered (zero forces were applied or commanded in the horizontal directions). The two robots were run with a communication delay of two seconds in each direction, i.e. a four-second round-trip delay.

Four experiments were conducted for the following situations:

1. Free space motion.
2. Unexpected discovery of a floor of unknown location.
3. Contact with the floor once the location is known.
4. Unexpected disappearance of the floor.

These four situations are illustrated in Figures 7–10, respectively. Each figure shows:

(i) the master position (solid curve) and virtual floor location (dashed curve) in the top graph;

(ii) the slave position (solid curve) and floor estimate (dashed curve) in the middle graph; and

(iii) the user (dashed curve) and slave (solid curve) forces in the bottom graph.

5.1. Free Space Motion

In free space, the slave tracked the master motions after the communications delay. This is depicted in Figure 7. Note the small PD tracking forces being commanded to the slave, in particular when the commanded motion makes sudden changes. As expected, the master force remains at zero.

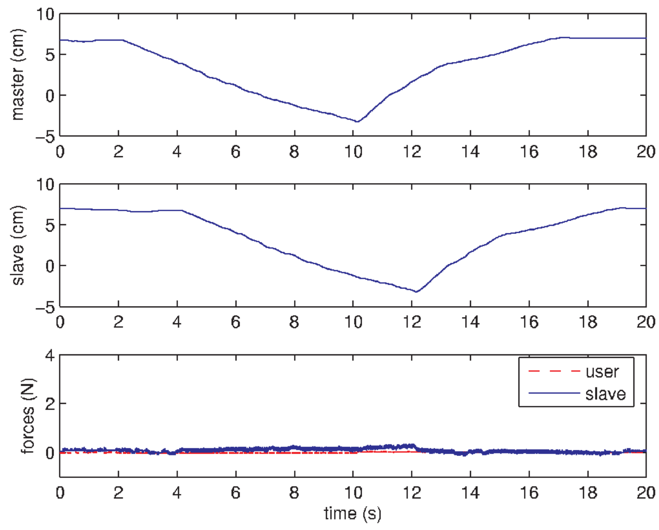

Fig. 7. Free space motion shows the slave tracking the master motion with minimal pd tracking forces.

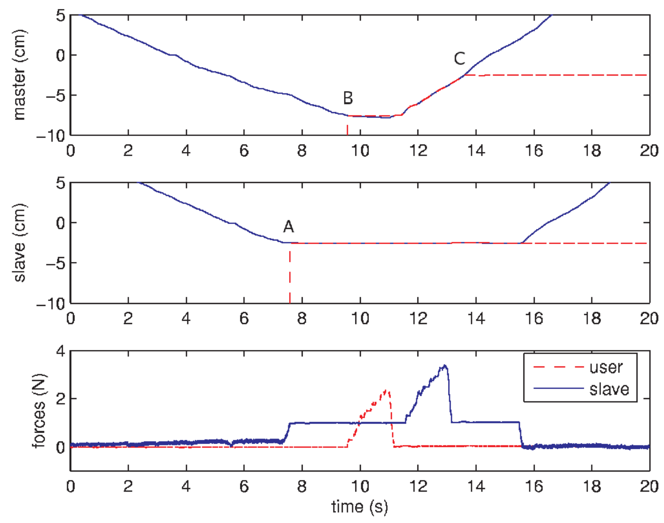

Fig. 8. Unexpected rigid contact shows slave forces limited to the prespecified detection threshold even though the master has substantially penetrated the contact location.

5.2. Floor Discovery

Starting with both robots in free space and no knowledge of an environment floor, the master was moved down and the slave made unexpected contact. The results are shown in Figure 8.

Before contact, the slave tracks the master motion using minimal forces. Upon contact (point A in Figure 8), as the PD tracking force increases to the prespecified level the slave acquires an estimate of the floor location. After the communication delay, this location is received at the master (point B in Figure 8), which then implements the virtual floor *as closely as possible without violating passivity*. The master sees a new force as the user hits the virtual floor, which the slave reproduces two seconds later. As the master pulls back, the master floor rises with the master until it reaches the known slave floor location (point C in Figure 8). During this time the master force remains at zero (as the user is not pushing down) and the slave maintains contact with the prespecified level. Upon withdrawal from the surface, master and slave models have converged and subsequent contacts will occur at the appropriate location.

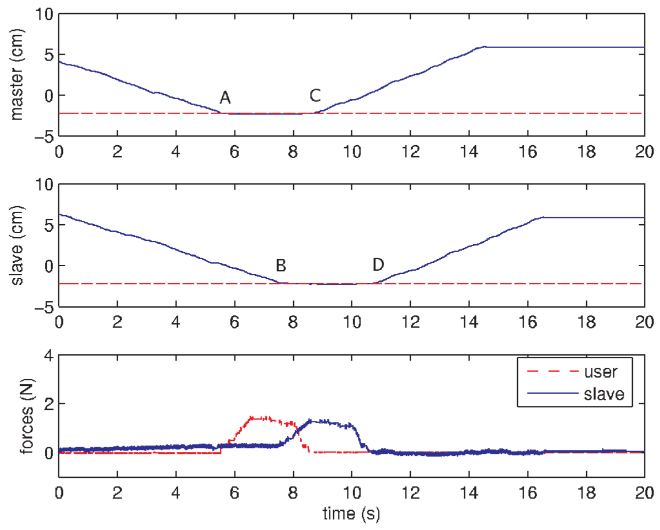

Fig. 9. During contact with a known floor, the master penetrates the virtual limit slightly to generate forces which are then reproduced by the slave.

5.3. Known Floor Interaction

The third test examined master and slave performance when the rigid floor location had already been accurately estimated by the slave and the master model had converged to the same value. Effectively, this is the second time that the user touches the floor. The results are shown in Figure 9.

The master stops at the virtual floor (point A in Figure 9), so that the slave tracking will also make contact after the communication delay (point B in Figure 9). Contact forces are then generated as the user penetrates the virtual floor (between points A and C in Figure 9). The slave tracks these forces but does not penetrate or attempt to penetrate the surface (between points B and D in Figure 9) as it tracks the master proxy location which also remains on the surface. Indeed, both slave position and force track the master quantities through the delayed transmission.

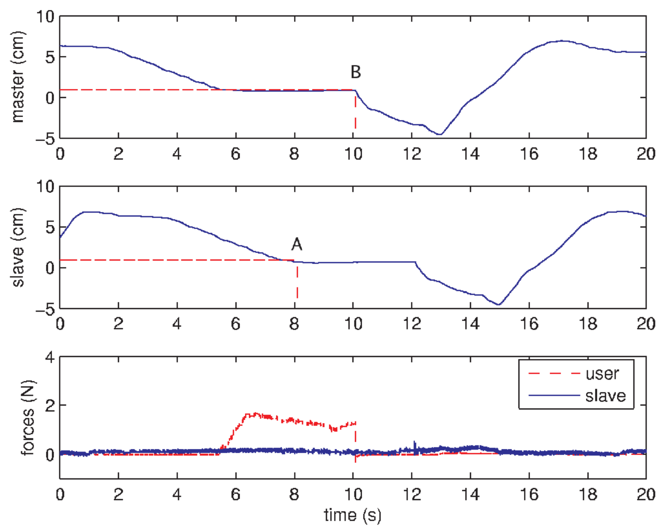

Fig. 10. Unexpected absence of the floor implies that the slave cannot track the master force and the virtual floor is removed as the model tracks.

5.4. Floor Disappearance

To examine all possible situations, the floor was physically removed without informing the slave or master model. As such, the slave expected contact and had to re-learn the absence thereof. The master model also removed the floor as the information arrived. The results are shown in Figure 10.

When the slave attempts to apply the master force, it penetrates the expected floor location (point A in Figure 10). Moving past the prespecified motion threshold, the floor estimate is removed and slave forces remain near zero. Until this information is received by the master, the user still perceives a floor and may continue to push against it. This situation will persist for one round-trip communication delay cycle, after which the master floor is also removed and regular motion tracking resumes (point B in Figure 10).

6. Conclusions

In this paper, the initial concepts have been presented for a bilateral teleoperation scheme designed to operate under several seconds of delay in unknown or partially known environments. Rather than depending on direct force and motion feedback between master and slave, which could render the system unstable and/or unusable, we have shown how model estimation at the slave site can be used as feedback to the master device. This information drives a virtual haptic model, allowing a user to interact naturally with the recreated environment. The ensuing force and motion commands thus take the slave and its environment into account and can be tracked by the slave robot through the delay.

The proposed approach and presented implementation was successful in locating and detecting a single rigid contact, and allowing users to apply forces to the contact in a controlled fashion. The model was extremely simple, although sufficiently rich to capture the encountered behaviors, recreate the experiences for the user and ultimately enable the slave to track both position and force across the substantial communications delay. These results are believed to be very promising, although it should be appreciated that the model complexity will need to be increased to capture more varied interactions. Indeed returning to Figure 2, the level of data abstraction in the presented work is low, as contact information is forced into a binary contact/no-contact decision.

Model complexity may need to be increased to identify contact stiffness, to allow for non-rigid interactions, as well as detecting contact surface normals, to allow operation in multiple dimensions. In all cases, the master model should track the slave estimated model, subject to passivity constraints to prevent system instability. Ultimately, complex geometries may be recognized in the slave environment and matched against known libraries of objects. This will increase the achievable abstraction level and further improve the operation under large delays.

Further research is also planned for more detailed comparisons of the proposed method against existing schemes, such as wave variable and predictive approaches, in the context of delayed telemanipulation.

In summary, this work is believed to be an important first step in bridging the gap between high-update-rate force-feedback architectures and complex supervisory controllers. System stability may be increased without loss of performance or perception, under the assumptions that the models are sufficiently rich to capture the environment and that this environment is not rapidly or continually changing.

Acknowledgments

This paper was made possible by the support of the National Science Foundation (IIS-0547672), the National Aeronautics and Space Administration (Institute for Space Robotics) and the assistance of Professor Oussama Khatib and Dr Jaeheung Park of Stanford University's Artificial Intelligence Lab. The authors would also like to thank the anonymous reviewers for their invaluable feedback in preparing this paper.

References

Anderson, R. J. and Spong, M. W. (1992). Asymptotic stability for force reflecting teleoperators with time delay. *International Journal of Robotics Research*, **11**(2): 135–149.

Bejczy, A. K., Kim, W. S. and Venema, S. C. (1990). The phantom robot: predictive displays for teleoperation with time delay. *Proceedings IEEE International Conference on Robotics and Automation*, vol. 1, pp. 546–541.

Casals, A., Fernandez, J. and Amat, J. (2002). Augmented reality to assist teleoperation working with reduced visual conditions. *Proceedings IEEE International Conference on Robotics and Automation*, vol. 1, pp. 235–240.

Ching, H. and Book, W. J. (2006). Internet-based bilateral teleoperation based on wave variable with adaptive predictor and direct drift control. *Journal of Dynamic Systems, Measurement, and Control*, **128**: 86–93.

Chong, N. Y., Kotoku, T., Ohba, K., Komoriya, K. and Tanie, K. (2001). Exploring interactive simulator in collaborative multi-site teleoperation. *Proceedings IEEE Workshop on Robot and Human Interactive Communication (ROMAN)*, pp. 243–248.

Daniel, R. W. and McAree, P. R. (1998). Fundamental limits of performance for force reflecting teleoperation. *International Journal of Robotics Research*, **17**(8): 811–830.

Gat, E., Desai, R., Ivlev, R., Loch, J. and Miller, D. P. (1994). Behavior control for robotic exploration of planetary surfaces. *IEEE Transactions on Robotics and Automation*, **10**(4): 490–503.

Gu, J., Augirre, E. and Cohen, P. (2002). An augmented-reality interface for telerobotic applications. *Proceedings IEEE International Workshop on Applications of Computer Vision (WACV)*, pp. 220–224.

Hashtrudi-Zaad, K. and Salcudean, S. E. (2000). Analysis and evaluation of stability and performance robustness for teleoperation control architectures. *Proceedings IEEE International Conference on Robotics and Automation*, pp. 3107–3113.

Lawrence, D. A. (1993). Stability and transparency in bilateral teleoperation. *IEEE Transactions on Robotics and Automation*, **9**(5): 624–637.

Makiishi, T. and Noborio, H. (1999). Sensor-based path-planning of multiple mobile robots to overcome large transmission delays in teleoperation. *Proceedings IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4, pp. 656–661.

Maneewarn, T. and Hannaford, B. (1999). Augmented haptics of manipulator kinematic condition. Proceedings of Telemanipulator and Telepresence Technologies VI. *Proceedings of SPIE—The International Society for Optical Engineering*, **3840**: 54–64.

Mitra, P., Gentry, D. and Niemeyer, G. (2007). User perception and preference in model mediated telemanipulation. *Proceedings IEEE 2nd Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (World Haptics)*.

Mitra, P. and Niemeyer, G. (2004). Dynamic proxy objects in haptic simulations. *Proceedings IEEE International Conference on Robotics, Automation, and Mechatronics*.

Mitra, P. and Niemeyer, G. (2006). Model mediated telemanipulation. *Proceedings of the ASME International Mechanical Engineering Congress and Exposition*.

Niemeyer, G. and Mitra, P. (2004). Dynamic proxies and haptic constraints. *Proceedings of the Workshop on Multi-point Interaction in Robotics and Virtual Reality*, Barbagli, F., Prattichizzo, D. and Salisbury, K. (eds). New York, Springer.

Niemeyer, G. and Slotine, J.-J. E. (2004). Telemanipulation with time delays. *International Journal of Robotics Research*, **23**: 873–890.

Park, J. and Khatib, O. (2006). A haptic teleoperation approach based on contact force control. *International Journal of Robotics Research*, **25**(5–6): 575–591.

Schooley, L. C., Zeigler, B. P., Cellier, F. E. and Wang, F.-Y. (1993). High-autonomy control of space resource processing plants. *IEEE Control Systems Magazine*, **13**(3): 29–39.

Sheridan, T. B. (1992). *Telerobotics, Automation, and Human Supervisory Control*. Cambridge, MA, MIT Press.

Sheridan, T. B. (1993). Space teleoperation through time delay: review and prognosis. *IEEE Transactions on Robotics and Automation*, **9**(5): 592–606.

Tanner, N. A. and Niemeyer, G. (2006). High-frequency acceleration feedback in wave variable telerobotics. *IEEE Transactions on Mechatronics*, **11**(2): 119–127.

Turro, N., Khatib, O. and Coste-Maniere, E. (2001). Haptically augmented teleoperation. *Proceedings IEEE International Conference on Robotics and Automation*, pp. 386–392.

Tzafestas, C. S. (2001). Teleplanning by human demonstration for VR-based teleoperation of a mobile robotic assistant. *Proceedings IEEE Workshop on Robot and Human Interactive Communication (ROMAN)*, pp. 462–467.

Utsumi, M., Hirabayashi, T. and Yoshie, M. (2002). Development for teleoperation underwater grasping system in unclear environment. *Proceedings IEEE International Symposium on Underwater Technology*, pp. 349–353.

Wang, M. and Liu, J. N. K. (2004). A novel teleoperation paradigm for human-robot interaction. *Proceedings IEEE Conference on Robotics, Automation, and Mechatronics*, pp. 13–18.