Stability of Model-Mediated Teleoperation: Discussion and Experiments

Bert Willaert[1], Hendrik Van Brussel[1] and Günter Niemeyer[2]

[1] Dept. of Mechanical Engineering, KULeuven, Belgium
bert.willaert@mech.kuleuven.be
[2] Willow Garage Inc., Menlo Park, CA, USA
gunter.niemeyer@stanford.edu

Abstract. The design of a bilateral teleoperation system remains challenging in cases with high-impedance slave robots or substantial communication delays. Especially for these scenarios, model-mediated teleoperation offers a promising new approach. In this paper, we present a first stability discussion. We examine the continuous behavior using general control principles and discuss how the model structure and its predictive power affects system lag and stability. We also recognize the unavoidability of discrete model jumps and discuss measures to isolate events and prevent limit cycles. The discussions are illustrated in a single degree of freedom case and supported by single degree of freedom experiments.
Key words: Teleoperation - Model-mediation - Stability

1 Introduction

Bilateral teleoperation systems allow the human operator to act on a remote environment via a robotic slave device while providing force feedback through a master device. Typically, the user's desire to feel directly connected to the environment is opposed to the need for stable interactions under all operating conditions. This well-known trade-off between transparency and stability is most challenging for systems characterized by either (1) limited response times, or (2) substantial communication time-delays.

The most conservative approaches, e.g. based on wave variables, can achieve stability even with large response times and long communication delays, but fail to provide transparency [1]. At the other extreme, more transparent approaches, e.g. the traditional position-force architecture, can hide the slave dynamics from the user's perception, but are prone to instability and often need high damping or low scaling factors [2,3]. Other methods vary the transparency versus stability trade-off in realtime monitoring for instabilities [4,5,6], most often by temporarily adding damping to provide stability at the cost of loosing transparency.

In the above works, controllers communicate position, force, or combined sensory information between master and slave. A fundamentally different approach involves transmission of models for the environment [7,8]. This has been called *impedance reflecting*, *virtual-reality based* or *model-mediated* teleoperation and proposes to achieve transparency with an indirect connection. Models of various complexity have been used, including a mass-spring-damper model [9], a

spring-damper model [10], a pure spring model [11,12] or merely a rigid wall with variable location [8]. By choosing the model structure consistent with the environment, most previous works have been able to focus on transparency with limited stability considerations.

In this work we concentrate on the closed-loop stability of the model-mediated framework proposed in [13], which defines a bilateral controller communicating abstracted model and task information: the model captures the environment while the task encodes the user intent. At the slave side, the *model estimation* uses sensory data to continuously estimate a model of the environment. At the master side, this model is used by the *model rendering* to generate the haptic feedback. The human response to the haptic rendering is monitored by the *task estimation* to generate a task description. Back at the slave side, the *task execution* regulates the slave to accomplish the incoming task as well as possible using the current model of the environment.

As the model should estimate the actual environment, many works implicitly assume the existence of a single correct or best model fit. For an unchanging environment and with sufficient excitation, the model estimator converges and stability is achieved by assumption. However, any model is a simplification and the value of teleoperation is highest for operations in unstructured environments which are at best difficult to model. Therefore, model convergence can not be expected. Instead the model should be treated as a time-varying signal. It should capture the currently most salient aspect of the environment and continuously adjust as the user explores and operates.

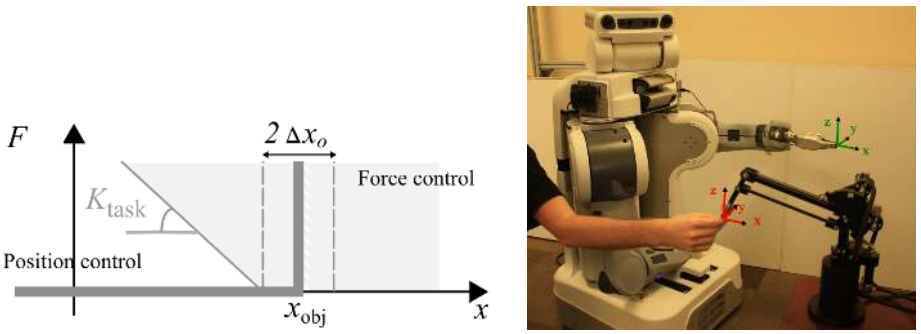
We examine closed-loop stability, **allowing environments to differ from the assumed model structure**. This encompasses two aspects: First, model errors lead to continuous *model adjustments* which close the loop with appropriate stability needs. We use classic loop gain and phase principles to review the behavior and discuss the implications of model structure. Second, discrete *model jumps* need to occur and have the potential to disturb the user or trigger limit cycles. We implement appropriate mitigation strategies. We guide and illustrate the discussion via a single degree of freedom case study and use the same framework to confirm the conclusions in experiments.

2 Model and Task Signals

To examine system stability, we must view the model and task descriptions as time-varying, continuous signals. To further illustrate and understand this perspective, let us consider a case study in a single degree of freedom. We first select a model structure and then present the corresponding controller elements, following the high-level model-task loop.

2.1 Model Structure

The underlying motivation for this study is the use of teleoperation in daily life scenario's. When considering daily life scenario's, many environments consist of moderately hard objects and tasks usually require moving, manipulating, or



(a) (b)

Fig. 1. (a) The model: the simple, rigid contact model is represented by the thick gray line: forces are zero for $x < x_{\text{obj}}$ while they are positive if $x = x_{\text{obj}}$. (b) The experimental setup: the slave is the left 7-d.o.f. arm of the PR2 robot and the master is a 3-d.o.f. haptic device designed at the University of Leuven.

assembling these objects. Hence we select a model that contains the location of a hard object. The model is very simple while capturing the most salient environment feature, being the combination of and transition between free-space and object contact. We already see that as objects in the real world change or are moved, this model will need to adjust and becomes time-varying.

The model, as shown in Fig. 1(a), allows free space motion along a single degree of freedom up to a rigid, immovable object located at x_{obj}. The model is quasi-static, relating force F only to position x: the force is zero if the position falls in front of the object ($F = 0$; $x < x_{\text{obj}}$), while it is positive if the position matches the object location ($F > 0$; $x = x_{\text{obj}}$).

2.2 Model Estimation

An instance of the model is maintained at the slave side to estimate the current environment. To update its parameter x^s_{obj}, the model estimation first has to decide whether the slave is in contact or in free space, corresponding to the vertical and horizontal branch of the model as shown in Fig. 1(a). To do so, the model estimation uses the following basic contact detector:

$$\text{contact state} \begin{cases} \text{'in contact' if } F_e \geq F_{\text{threshold}} \\ \text{'free space' if } F_e < F_{\text{threshold}} \end{cases}, \tag{1}$$

where F_e stands for the measured or estimated interaction force with the environment and $F_{\text{threshold}}$ has to be chosen depending on the noise level of the measured or estimated interaction force F_e. While in contact, the object location is updated and estimated using a first order low-pass filter of bandwidth λ:

$$\dot{x}^s_{\text{obj}} = \begin{cases} \lambda(x_s - x^s_{\text{obj}}) & \text{if 'in contact'} \\ 0 & \text{if 'free space'} \end{cases}. \tag{2}$$

2.3 Model Rendering

A separate model instance is maintained at the master side for rendering the haptic feedback, with an object location x^m_{obj}. The master model is updated with the most recent incoming slave model

$$x^m_{\text{obj}}(t) = x^s_{\text{obj}}(t - T_d), \tag{3}$$

where T_d is a possible communication delay. The object is rendered as a one-sided pure stiffness, which means that the master device only exerts forces on the human operator if he/she has virtually penetrated the object. To define this behavior, a haptic proxy is used:

$$F_m = K^m_p(x_{\text{proxy}} - x_m) \quad \text{with } x_{\text{proxy}} = \begin{cases} x_m & \text{if } x_m \leq x^m_{\text{obj}} \\ x^m_{\text{obj}} & \text{if } x_m > x^m_{\text{obj}} \end{cases}, \tag{4}$$

Because the model represents a infinitely stiff object, the gain K^m_p should be set as high as practically possible.

2.4 Task Estimation

We employ a simple task representation to encode the user's intent, namely a pair of position and force values $(x, F)_{\text{task}}$. The force value F_{task} is set to the human force acting against the model, while the position value x_{task} is set to the master proxy location. The proxy location maps the user's position onto the model, i.e. it creates a position that is consistent with the displayed model. As such, the task is feasible against the model, independent of any practical limitations on the gain K^m_p.

2.5 Task Execution

The task $(x, F)^m_{\text{task}}$ estimated by the master is communicated to the slave

$$(x, F)^s_{\text{task}}(t) = (x, F)^m_{\text{task}}(t - T_d), \tag{5}$$

and compared against the most recent environment model. If the task lies in the grey zone in Fig. 1(a), i.e. if $F^s_{\text{task}} > K_{\text{task}}(x^s_{\text{obj}} - x^s_{\text{task}} - \Delta x_0)$, the user is most likely expecting contact and the slave is placed under force control using a natural-admittance type controller [14]. Otherwise, the slave is operated in position control mode.

3 Continuous Closed-loop Stability

Model mediated systems are nonlinear and preclude simple analyses. Nevertheless we can gain substantial insight by examining the high-level loop between master and slave. We see why model mediation can provide real improvements and how the model selection and its predictive power affects the stability.

3.1 Classic Stability: Gain Stabilization

Consider the classic stability problem of a telerobot with direct force feedback contacting a stiff environment [2,3]. The user's movements are measured and commanded to the slave, executed, causing the environment forces, which communicated back to and displayed to the user. From a linear system's perspective, an overall transfer function maps user motion to force feedback. Its gain is the environment stiffness. Its phase lag stems from two sources: (a) communication delay and (b) slave controller response lag. So if the slave tracking or communication are slow *and* the environment contact is stiff, the transfer function shows both large gain and phase and predicts instability. Commonly, to stabilize this system, the force feedback gain is lowered. Equivalently, the transfer function gain is reduced, equivalent to gain stabilization.

3.2 Model-Mediated Stability: Phase Stabilization

Model mediation can be seen as phase stabilization. Phase stabilization requires phase lead, which provides signals earlier in time, and hence is achieved by prediction. Model mediation effectively predicts and displays the environment interaction force that will happen when the task is executed, by assuming knowledge of the environment. Stability is thus determined by the model's predictive power, i.e. over which time horizon and how accurately predictions of environment interactions can be made.

Consider the model-mediated loop: user tasks are sent to the slave and eventually lead to feedback of model adjustments. The loop gain describes the size of model adjustments that result from a given task. Following the case study, when the user advances, an object may slide or deflect. Thus a model adjustment in the form of a shifted object location is necessary. The loop gain describes the size of the location shift for the size of the user's movement.

More generally, model adjustments occur when the predicted motions and/or forces differ from the observed values. Larger prediction errors require larger adjustments. Thus the size of prediction errors correlate to the loop gain. A model with greater predictive power lowers the loop gain and improves stability.

Meanwhile the loop phase describes how quickly the adjustments are received. This stems from three sources: (a) the communication delay, (b) how quickly the user commands are executed (slave controller response lag), and (c) the model estimator. Faster model estimation lowers the loop lag and improves stability, subject to a lower bound on lag given by the other two sources. In our case study, the single model parameter can be *estimated* very quickly.

The selection of an appropriate model structure and complexity must balance these loop gain and phase needs. Choosing a more appropriate structure will help predictive power. Choosing a simpler model will lower the number of parameters and increase the possible estimation speed. Of course, if the other sources of lag dominate the estimation time constants, the balance will shift to favor greater predictive power and devalue simplicity.

4 Discrete Model Jumps and Stability

Under normal operating conditions, model adjustments happen continuously and slowly. However, sudden and drastic updates to the model may be necessary when new environment features are discovered. In our case study, imagine detecting a previously unknown object. The model should immediately jump to the newly discovered location. But, due to any lag source, the master will be leading the slave when the contact is detected. Immediately displaying this object to the user requires the master to jump backwards and triggers a potentially dangerous discontinuity in force levels.

In general, the information encoded in such temporal model discontinuities or jumps is important. It indicates a radical change in the expected environment and should not be filtered or removed from the user's perception. However it poses the danger of introducing a discontinuity into the task, which could trigger a jump in the slave movement or force, and ultimately excite unmodelled dynamic elements in the environment, robot, or controllers. In turn, with an imperfect model structure, the estimation may react with another drastic or discrete jump, escalating into a limit cycle or other stability problems. Hence, model jumps should remain isolated and contained from propagation. Note that longer lags between master and slave will enlarge the size of necessary model jumps and increase the need for explicit handling and containment.

Explicit handling will appear in nearly all subsystems. Model estimation must determine when it is necessary to initiate jumps. Model rendering must make sure the users are informed without confusion and without triggering impulsive reactions in the task. And the task execution must decide how to interpret a task that was based on an obsolete model without exciting further jumps. In the following, we demonstrate this in our case study.

4.1 Triggering Model Jumps

The model estimation can determine two distinct events that require a model jump, i.e. the detection of an unknown object and the disappearance of an expected object. These situations are detected and acted upon as follows:
Detection: if ($x_s < x^s_{\text{obj}} - \Delta x_0$ and 'in contact') reset ${}^s_{\text{obj}} = x_s$,
Removal: if ($x_s > x^s_{\text{obj}} + \Delta x_0$ and 'free space') reset ${}^s_{\text{obj}} = +\infty$.

4.2 Rendering Model Jumps

A previous user study has compared several methods of rendering model jumps [15]. In our implementation, we select a gradual, constant-time introduction or removal of the object if the user is or would be in contact with the object:
Detection: if ($x_m > x^s_{\text{obj}}$) introduce x^m_{obj} at x_m and move to x^s_{obj} over a period of time t_{move},
Removal: if ($x_m > x^m_{\text{obj}}$) move x^m_{obj} from x^m_{obj} to x_m over a period of time t_{fade} before removal.

Table 1. Gains and parameters used for the experiments

K_p^s: 2000 N/m	G_v: 20 Ns/m	Δx_o: 0.004 m
K_v^s: 10 Ns/m	m_d: 7 kg	t_{move}: 500 msec
K_{task}: 250 N/m	K_p^m: 4000 N/m	t_{fade}: 300 msec

4.3 Task execution under Model Jumps

Task execution depends on the current environment model. Explicit handling is thus necessary to avoid discontinuities in the controller when the model jumps.

Just before an object removal the controller is most likely in force control mode, as the user is pressing against the expected object. At the moment of the removal, the slave position x_s is leading the object location x^s_{obj} by Δx_0 (see 4.1). The removal switches the controller into position control mode, while the task position x^s_{task} is still the *old* object location. To ensure smoothness, a position offset is added to the current task and then gradually removed over a period of time t_{fade}.

At the moment of an object detection the interaction force is equal to the threshold force for the contact detector $F_{threshold}$ while the task force F^s_{task} is still at zero. If the threshold is small, no transition measure is necessary. Otherwise, a force offset may be introduced and gradually removed.

5 Experiments

Following the single degree of freedom case study, we substantiate our discussions in experiments performed in three stages. First, we confirm the limitations of two classical control architectures. Next, we demonstrate that the model-mediated approach performs well independently of lags if the model adequately captures the physical environment. Finally, we show how performance degrades when lag appears and the environment is inconsistent with the assumed model. In all tests, the system begins out of contact. The user moves to and interacts with the unknown environment. The figures 2(a) - 3(d) show the data, including the master (blue) and slave (green) position and force signals plotted against time as well as each other. The position graphs further show the object locations in dashed lines, as estimated by the slave and rendered at the master.

5.1 Experimental Setup

The experimental setup is shown in Fig. 1(b). The slave is the left 7-d.o.f. arm of the PR2 robot. The master is a haptic device designed at the KU Leuven. The Cartesian y-axes of both devices are linked to create a 1-d.o.f. system, using the transpose of the Jacobians to transform all control forces to motor torques. The master device is low impedance (mass $\approx$ 0.4 kg, $F_{friction} \approx 1$ N) and the applied motor torques provide a good estimate of the user forces. The PR2 arm has a higher impedance (mass$\approx$ 7 kg and $F_{friction} \approx 3$ N) and uses a 6 axis force/torque sensor integrated in the wrist. The threshold for the contact detector, $F_{\text{threshold}}$, was set to 1 N. Tuning of the position and force controller

results in a 2.7 Hz ($\sqrt{\frac{K_s}{M_s}}$) and 0.5 Hz bandwidth ($\frac{G_v}{M_s}$) respectively. The latter results in poor force tracking which is clearly visible in the experimental results. Flexibility and backlash in the PR2 arm lead to an uncertainty in its gripper position and thereby the model estimation of $\Delta x_0 = 4$mm. The model update filter is set to a bandwidth of 10 Hz. Table 1 summarizes all properties.

5.2 Experimental Results

Classical Architectures: Model-Free controllers. As mentioned earlier, stable and transparent interaction with very different environment types is really challenging in case of a slave robots with a limited response time, even in the absence of significant communication time delay. The performance of the two extreme classical controllers is a good illustration of this.

Fig. 2(a) shows the system under position-position control, where the master and slave track each other's position. This stable architecture is able to achieve a 1:1 force scale, but the user feels the slave inertial and residual friction forces, overlaid on the environment forces. For a high-impedance slave robot, especially in free space, this controller results in poor transparency.

Fig. 2(b) shows the system under position-force control, when the feedback force is directly applied to the master. While this completely hides the slave's inertia and friction, a 10:1 force reduction is required in order to obtain stable interaction. This reduction is necessitated by the high mass and slow response times of the PR2-arm [2,3] and results in poor transparency during contact.

Model-Mediation: Consistent Environment. In the second set of experiments, the system uses model-mediated control to interact with one fixed hard object. Fig. 2(c) and 2(d) show the behavior without and with and artificial round-trip communication delay of 150 ms. Note that with a 2.7Hz bandwidth, the slave has a settling time greater than 150 ms and adding the communication delay is no worse than doubling the overall lag.

Overall, Fig. 2(c) and 2(d) show excellent position and force tracking in both contact and free space. These experiments show that in case the model adequately captures the physical environment, a model-mediated controller can give the free-space feeling of the position-force architecture while achieving the stability and 1:1 force scaling of the position-position architecture.

As the model adequately captures the physical environment, there are no significant continuous model adjustments. However, discrete model jumps are needed when the object is first detected (at 0.8s/1.5s in Fig. 2(c)/ 2(d)). At these times the slave lags the master (by 5 mm/15 mm) with the additional time delay creating more lag. As described in 4.2, the master fades in the model leading to the observed gradual re-convergence of master and slave positions. Especially in Fig. 2(d), this effect is visible in the position-force graph as a non-physical artifact which directly alerts the user of the object detection without triggering subsequent effects.

Discrete jumps are also needed after the object is secretly removed (at 6s/7s). Unaware of the removal, the system allows the user to apply forces against the

rendered model. When the slave attempts to track these forces, it moves and detects the object disappearance (see 4.1). The master renders this removal by fading out the model and fading the forces to zero (see 4.2). Again, the non-physical effect, clearly visible in the position-force graph, alerts the user of the discontinuity without triggering additional effects.

Model-Mediation: Inconsistent Environments. In the last set of experiments we challenge the model-mediated controller with inconsistent environments that are not fixed (sliding object) or not hard (soft object), again without and with a round-trip delay of 150 ms. Fig. 3(a) shows the model-mediated controller in case of pushing a hard object over a surface ($F_{fr,\text{obj}} \approx 6$ N). Upon initial contact, the user's force rises while the object remains stationary. Once the user's force exceeds the static friction, the object slides. The model is continuously updated allowing the user to feel the object motion. Motion ceases when the applied force drops. Fig. 3(b) shows the same controller in case of compressing a fixed soft object($K_{\text{obj}} \approx 600$ N/m). Again the model is continuously updated so that both position and forces are tracked and the user feels a soft object. The experiments show that, despite the significant loop lag due to the limited response time of the slave, the predictive power of the model is sufficiently high for these contact scenario's.

For the experiments with the additional time-delay, this is no longer the case. Fig. 3(c) and 3(d) show the delayed model-mediation interacting with the sliding object and the soft object. In comparison to Fig. 3(a) and 3(b), we see a clear distortion of the position-force graphs. This performance degradation can be explained by the combination of the limited predictive power of the model, i.e. the need for significant model updates (high loop gain), in combination with a big loop lag. A model with better predictive power is necessary to retain performance under the delay here. For example, a spring model [10,11,12] may be better suited to handle soft objects in this case.

6 Conclusions

In this work, we further formalized the model-mediated framework described in [13]. We discussed stability and introduced the notion of continuous *model adjustments* versus discrete *model jumps*.

We discussed qualitatively how the model choice impacts the overall system stability during continuous model adjustments: the selection of a model structure has to balance the loop gain and phase. Better predictive power reduces loop gain while faster model estimation lowers lag. Systems with longer inherent lag will favor models with better predictive power.

The experiments demonstrate this in two ways: (1) as long as the model captures the environment adequately, i.e. the model has an excellent predictive power (small loop gain), extra lag in the closed-loop does not result in performance degradation. (2) if the model is, however, not consistent, i.e. the model has a limited predictive power (higher loop gain), the performance does degrade

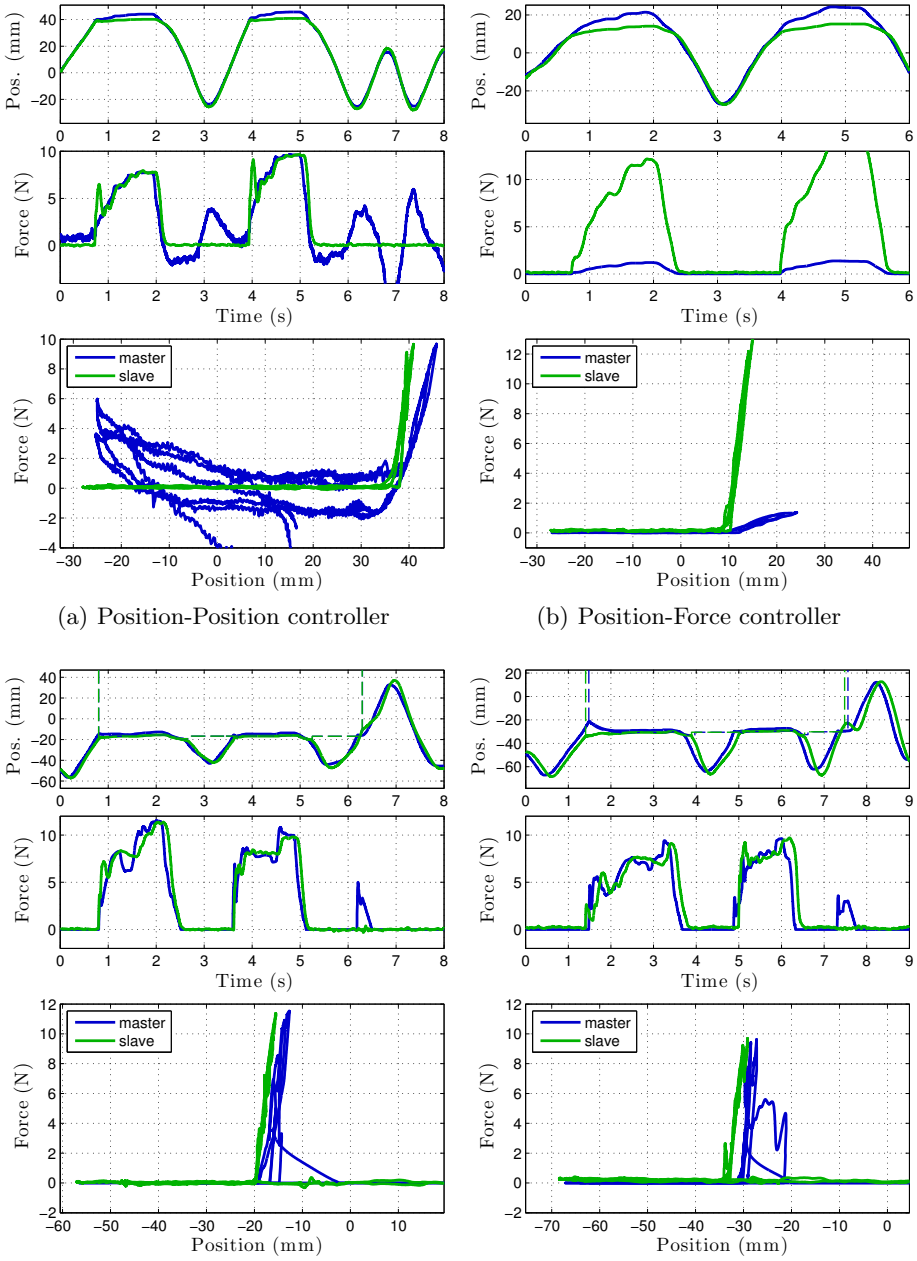
if the overall loop lag increases. For scenarios when the environment can have very difference characteristics and large lags are unavoidable, the use of multiple models in one controller should be explored in future work.

This is one of the reasons why we also recognized the need for discrete model jumps and discussed how and why these discrete events should remain isolated, i.e. they should not trigger subsequent effects.

Being a first systematic stability discussion of model-mediated teleoperation, we hope this research not only demonstrates the value of model-mediation but opens a tantalizing avenue to further explore the relationship of a model's predictive power with other system parameters.

References

1. G. Niemeyer and J-J.E. Slotine. Stable adaptive teleoperation. *IEEE Journal of Oceanic Engineering*, 16:152–162, 1991.
2. R.W. Daniel and P.R. McAree. Fundamental limits of performance for force reflecting teleoperation. *The Int. J. of Robotics Research*, 17(8):811–830, 1998.
3. B. Willaert, B. Corteville, D. Reynaerts, H. Van Brussel, and E.B. Vander Poorten. A mechatronic analysis of the classical position-force controller based on bounded environment passivity. *The Int. J. of Robotics Research*, 30(4):444–463, 2011.
4. B. Hannaford and J. Ryu. Time domain passivity control of haptic interfaces. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 1863–1869, 2001.
5. L.J. Love and W. J. Book. Force reflecting teleoperation with adaptive impedance control. *Trans. on Systems, Man and Cybernetics (Part B)*, 34(1):159–165, 2004.
6. D. Ryu, J-B. Song, J. Choi, S. Kang, and M. Kim. Frequency domain stability observer and active damping control for stable haptic interaction. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 105–110, 2007.
7. B. Hannaford. A design framework for teleoperators with kinesthetic feedback. *IEEE Transactions on Robotics and Automation*, 5(4):426–434, August 1989.
8. P. Mitra and G. Niemeyer. Model-mediated telemanipulation. *The International Journal of Robotics Research*, 27(2):253–262, February 2008.
9. K. Hashtrudi-Zaad and S.E. Salcudean. Adaptive transparent impedance reflecting teleoperation. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 1369–1374, Minneapolis, Minnesota, April 1996.
10. A. Achhammer, C. Weber, A. Peer, and M. Buss. Improvement of model-mediated teleoperation using a new hybrid environment estimation technique. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 5358–5363, 2010.
11. C. Tzafestas, S. Velanas, and G. Fakiridis. Adaptive impedance control in haptic teleoperation to improve transparency under time-delay. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 212–219, 2008.
12. B. Willaert, P. Goethals, D. Reynaerts, H. Van Brussel, and E. Vander Poorten. *Advances in Haptics*, chapter 13: Transparent and Shaped Stiffness Reflection for Telesurgery, pages 259–282. IN-TECH, 2010.
13. June Park. *Improving Teleoperation by using models and tasks.* PhD thesis, University of Stanford, 2009.
14. W. S. Newman. Stability and performance limits of interactions controllers. *Journal of dynamic systems, measurement, and control*, 114(4):563–570, 1992.
15. P. Mitra, D. Gentry, and G. Niemeyer. User preferences and performance in model mediated telemanipulation. In *World Haptics Conf.*, pages 268–273, 2007.



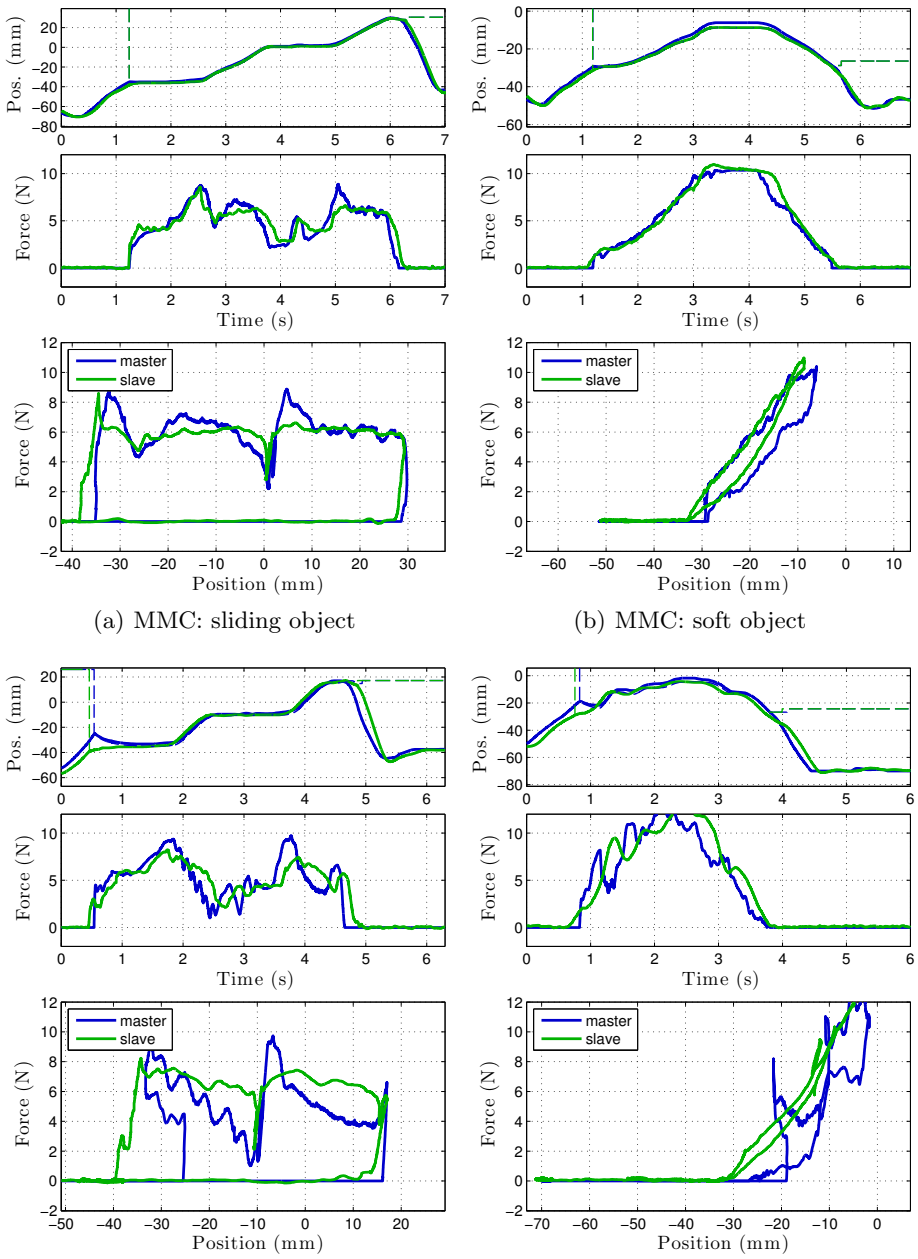
(a) Position-Position controller

(b) Position-Force controller

(c) MMC: rigid object

(d) MMC: rigid object and 150 ms time delay.

Fig. 2. (a-b) Poor performance with classical control schemes. (c-d) Model-mediated controller (MMC) and a rigid environment: good performance achieved by a well-matched model independent of the overall lag.



(a) MMC: sliding object

(b) MMC: soft object

(c) MMC: sliding object and 150 ms delay.

(d) MMC: soft object and 150 ms time delay.

Fig. 3. Model-mediated controller (MMC) and a sliding and a soft object: (a-b) good performance despite the poorly matched model and (c-d) poor performance due to the extra lag.