ROBOTICS AND MECHATRONICS

EXTENDING AN ISOTROPIC VIRTUAL ENVIRONMENT MODEL WITH GEOMETRICAL INFORMATION IN MODEL-MEDIATED TELEOPERATION

C. (Christophe) van der Walt

MSC ASSIGNMENT

Committee:
prof. dr. ir. S. Stramigioli
dr. ir. D. Dresscher
prof. dr. J.B.F. van Erp

November, 2020

052RaM2020
Robotics and Mechatronics
EEMCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE
UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Extending an isotropic virtual environment model with geometrical information in model-mediated teleoperation

Christophe van der Walt

October 19, 2020

Abstract **- Haptic Teleoperation is a means of performing tasks that require human levels of dexterity without a human needing to be present. This has a plethora of applications in various industries. However, time delay poses certain problems, notably with dynamic stability, in a lot of these systems. Model Mediated Teleoperation (MMT) techniques provide a means of mitigating energy creation in time-delayed communication channels. MMT techniques in literature either estimate geometric or dynamic properties of the environment. This paper proposes a system that estimates both kinds of properties to render force for a environment consisting of a flexible object on a table. We obtained promising results, but inaccuracies in high-frequency performance and in dealing with environmental inhomogeneity require further study.**

1 Introduction

Haptic teleoperation over time-delayed communication channels is a concept with many stakeholders in the fields of disaster relief and remote employment. Ideally, teleoperation systems would allow for any task that requires human levels of dexterity to be performed from anywhere in the world. For these systems, it is important that the telerobot device tracks the motion of the operator accurately, and that the interaction force at the telerobot end is rendered to the operator accurately. This work focuses specifically on the latter aspect of teleoperation systems.

Sending information about motion and force directly over time-delayed communication channels can result in instabilities. A number of mathematically stable systems exist [13, 23, 26], but these trade off stability for performance. Model-Mediated Teleoperation (MMT) [30] offers some solutions to these performance problems. Impedance Reflection[27], a form of MMT, provides good performance for isotropic environments. However, it relies on a model structure for the environment, so it can exhibit poor performance if the perceived dynamics of the environment are different from the modeled dynamics. Point-cloud based MMT [29] provides good performance for randomly configured environments, but is sensitive to point cloud quality and can only render force normal to surfaces.

Here, we present a system which uses aspects of the approach by [29] to augment the work in [27] with information about the configuration of surfaces in the environment. The system is further augmented by using model structure switching (one model representing the dynamics of the system being in contact with an environment, and another representing free motion) to allow the operator to feel accurate force feedback from a non-isotropic environment.

The research question considered here can be formulated as follows:

"To what extent can a teleoperation system using Impedance Reflection augmented by information from an environmental point cloud accurately render force to an operator from an arbitrarily distributed 3D environment?"

2 Related Work

There are many methods of providing stable haptic control in teleoperation that are robust to time delays. [26, 23, 13] propose systems that are mathematically stable. These rely on (variable) damping or torque saturation based on a passivity condition to guarantee passivity, and thus stability. The advantage of these methods is their robustness. They can always stabilise a time-delayed haptic teleoperation system. However, for larger time delays, performance can be significantly hindered, in such systems. For [26] and [23], the operator feels damping in the haptic feedback that scales with the time-delay and the frequency of the operation. For [13], the user is unable to carry out actions at all, if the passivity condition is broken, which happens more readily for large time delays.

Model-Mediated Teleoperation (MMT), on the other hand, is able to provide haptic feedback without actively imposing passivity on the system. In this case, information about the telerobot's environment is used to recreate an estimate of this environment on the operator side. In such architectures, time delay provokes discrepancies in force rendering instead, without immediate effect on the stability. However, these systems are not always mathematically guaranteed to be stable.[30]

Within MMT, there are many different model structures and types for the environment, which affect how the system behaves. Neural Network-based approaches as in [16] can capture a high variability in environmental behaviour, but require time to train, and dont seem to have experimental validation in literature, as of yet. Impedance Reflection, as in [27] and [15] uses a relatively simple, application specific, and parametric model of the environment to generate haptic feedback. It will generally only model dynamics deemed necessary, and so will generally be less computationally intensive than a neural network based approach. However, complex environmental geometries introduce additional parameters into these models, which affects the performance. Point-cloud-based Model-Mediated Teleoperation (pcbMMT) [29], however, uses visual information to generate a point cloud and uses interaction with the point cloud to render force feedback, or estimate geometrical parameters related to the point cloud. This is useful for capturing force feedback when interacting with variable and arbitrarily configured 3D environments. A problem with such a method is that the interaction point always needs to be in view of the camera, which is a scenario that cannot always be guaranteed.

For impedance reflection, there are a limited number of model structures that have been studied. [21] and [19] can efficiently estimate mass-spring-damper-analogue models of an environment, which are appropriate for rigid environments with changeable geometry. [24] proposes a system which can capture flexible dynamics of an environment with a static geometry. There are no models in literature that capture flexible objects that are allowed to also move within an environment. In general, objects within a given environment are both movable and flexible, so a model structure capable of capturing both of these effects would make for a more accurate impedance reflection system. This is something this research attempts to solve.

3 Methods

This section presents all the concepts and design choices relevant to the proposed system. Firstly, we present the forms of Model-Mediated Teleoperation used: Impedance Reflection, and how pcbMMT is used to augment it with information about the the local configuration of surfaces. Some elaboration on the estimator used in the impedance reflection algorithm is also given. Finally, an overview of the full system is given.

The system detailed here produces haptic feedback from a model of the environment consisting of a single flexible object resting on a horizontally oriented surface, and free to move on the aforementioned surface, but also constrained to it. The mass is not allowed to leave the surface. What is more, the system also supports the user establishing and breaking contact with the object as they are manipulating it.

This allows us to perform proofs of concept on the ideas of model switching based on point cloud data, as well as impedance reflection with complex model structures. These concepts are all required to work before such a system can be generalised to work with multiple objects movable through an environment consisting of arbitrarily configured surfaces.

3.1 Model-Mediated Teleoperation

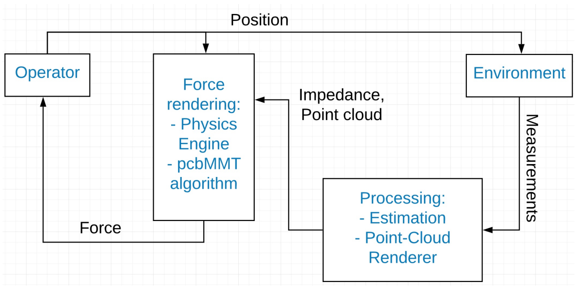
Position
Operator
Force rendering: - Physics Engine - pcbMMT algorithm
Impedance, Point cloud
Environment
Measurements
Processing: - Estimation - Point-Cloud Renderer
Force

Figure 1: Model-Mediated Teleoperation Process Flow

All Model-Mediated Teleoperation follows the process described in figure 1. An input reference motion is sent to the telerobot side of the teleoperation system. In most setups involving manipulators, a controller is needed to make sure the input motion is tracked. In this way, an operator can interact with the environment. Measurements are then collected about the state of the environment. These are then processed for ease of manipulation. The resulting information about the state of the environment will be sent to the operator side. This information is used to build up an impression of the environment locally. The input reference motion can then be used together with a force rendering protocol based on this local impression of the environment to compute an output force to send to the operator.

In Impedance Reflection, the processing step is an estimator that produces an estimate for a parametrisation of the environment's impedance. In pcbMMT, the processing involves taking in the raw camera data and turning it into a point cloud. In this section we explain some key concepts in both of these MMT techniques.

We distinguish here between two types of information from the environment: Geometric and Dynamic.

3.1.1 Impedance Reflection

Impedance-Reflection Teleoperation uses dynamic information about the environment. It uses a dynamic model structure of the environment, an online parametric estimation algorithm (Kalman Filter, RLS), and measurements of position, force, and their derivatives to compute the dynamic parameters on which the model depends. The parameters are then streamed to the master side, where the model structure in the force renderer is updated with the most recent parameter estimates. An output force can then be computed from the position of the telerobot device and the model relating position and force in the force renderer.

For example, consider a mass-spring-damper system as a model structure for the environment. The model structure for the dynamics is shown in equation 1. Measuring F_e, $\frac{d^2x_e}{dt^2}$, $\frac{dx_e}{dt}$, and x_e over time allows one to compute m, b, and k. These can then be sent to the master side, to populate the differential equation shown in equation 2, which can be solved for $F_m(t)$, knowing $x_m(t)$. In both of these equations $F_{e,m}$ refers to input force on the mass, and $x_{e,m}$ refers to the displacement of the mass.

$$F_e = m\frac{d^2x_e}{dt^2} + b\frac{dx_e}{dt} + kx_e \tag{1}$$

$$F_m = m\frac{d^2x_m}{dt^2} + b\frac{dx_m}{dt} + kx_m \tag{2}$$

One can expect the reproduction of F to be accurate if the model is complex enough to accurately capture the dynamics of the environment that are significant to the use case, and if its measurements of position, force and its derivatives, are sufficiently accurate.

If the environment cannot be modeled by a single model structure, logic-based model-structure switching can solve this problem. For example, based on whether or not the operator is in contact with an object or not switching can occur. One model structure then describes the dynamics of interaction with the object, and another the dynamics of free motion in the space, and the renderer switches models upon the operator making or breaking contact with a surface.

3.1.2 pcbMMT

pcbMMT uses geometrical information of the environment. A point cloud of the surface of the environment is measured and processed by an RGB-D sensor such as a Microsoft Kinect. This point cloud is then streamed to the master. On the master side, the master haptically interacts with the point cloud of the environment. The master position, also referred to in literature as Haptic Interaction Point (HIP), is tied to a proxy sphere by a spring. The master can pass through the point cloud of a surface, but the proxy, if designed appropriately, can not. When the master position passes through the surface, a force is sent to the operator equal to the force in the spring (see equation 3).

$$F_r = k\left(x_{proxy} - x_{HIP}\right) \tag{3}$$

The proxy itself is made up of 3 concentric spheres. The inner sphere is used as a region inside which points are not allowed. If points are located in this sphere, the proxy moves itself back such that this is not the case. The scenario in which there are points in the inner sphere is called entrenchment. The middle layer between the shell of the first sphere and the shell of the second is used to detect if the proxy is in contact with the point cloud. The biggest sphere is used to select points from which to compute a local surface normal estimate, which is in turn used by the algorithm that moves the proxy. This is done by computing a normalised, weighted average of the vector stretching from the HIP to all the points contained within the sphere. The choice of the size of these spheres is up to individual design and a good choice depends on the density, noise content and velocity of the point cloud. Figure 2 shows such a system interacting with both a continuous surface and a point cloud.

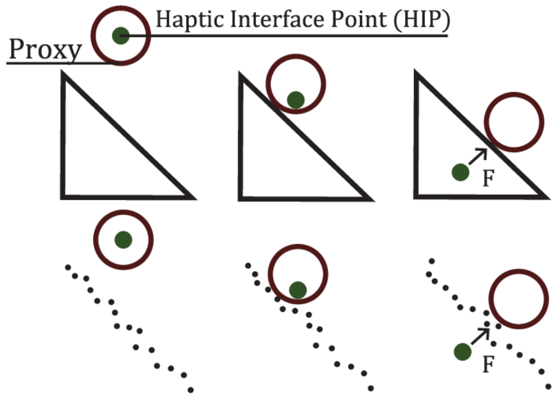

Figure 2: Proxy-based force rendering for interaction with surfaces [25]

For this system, we forgo the haptic rendering, as impedance reflection already outputs a force, and pcbMMT can only produce force feedback in a direction normal to a surface. What is used from this system is the contact estimation to determine whether or not the operator is in contact with an object.

3.2 RLS Estimation and MMT Model Structure

To identify all the parameters in the model used for impedance, we choose a "Recursive-Least-Squares" (RLS) estimator. This method is chosen because it computes the estimate recursively, which is more efficient than a non-recursive method. Also, Least-Squares methods are used frequently in model-mediated teleoperation ([21],[22],[19],[28]), so considerations for the implementation of such an estimator are well documented.

RLS estimators minimise the Least-Squares error between the output of a system and the simulated output of its model, based on the parameters on which it depends. The coupled difference equations for computing this estimate recursively are in equations 4 and 5: [20]

$$\hat{\theta}_{k+1} = \hat{\theta}_k - \frac{P_k \psi_{k+1}}{1 + \psi_{k+1}^T P_k \psi_{k+1}} \left(\psi_{k+1}^T \hat{\theta}_k - y_{k+1} \right) \tag{4}$$

$$P_{k+1} = P_k - \frac{P_k \psi_{k+1} \psi_{k+1}^T P_k}{1 + \psi_{k+1}^T P_k \psi_{k+1}} \tag{5}$$

where $\hat{\theta}$ is the parameter estimate vector, ψ is the regressor vector, y is the output, and the model for which parameters are being estimated has the form:

$$y = \psi^T \theta \tag{6}$$

In this instance, we make use of a model of a moving mass connected to the fixed world with a nonlinear damper representing coulomb and viscous friction, and a spring. It is connected to the interaction point with the user via a Kelvin-Voigt contact model. We choose this model as it captures the behaviour of an object that is both flexible and movable in space. More complicated models describing these behaviours could be made, but extra parameters increase the convergence time of the estimator, which results in poorer performance. An IPM of this can be seen in figure 3.

The equations of motion of this system can be put in the form of equation 6, by choosing the values for ψ and θ shown in 7.

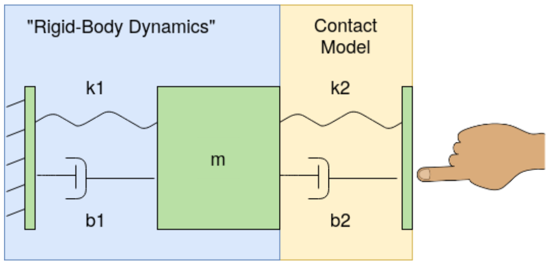

Figure 3: Environment Model for when the operator is in contact with the object

$$\psi = \begin{bmatrix} -\frac{d^3x}{dt^3} \\ -\frac{d^2x}{dt^2} \\ -\frac{dx}{dt} \\ -x \\ -\frac{d^2F}{dt^2} \\ -\frac{dF}{dt} \\ -sgn(\frac{dx_m}{dt}) \end{bmatrix}, \theta = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{bmatrix} \tag{7}$$

x and F represent the interaction force and interaction position. a, b, c, d, e, f, and g are the parameters that make up θ. Their relation to the parameters from figure 3 is detailed in equation 30.

In this instance, $\frac{dx_m}{dt}$ represents the velocity of the mass. However, it is impossible to directly measure this value in this scenario, as this is an internal state of the system. We therefore choose to take the velocity of the interaction point instead. This poses some problems if the difference in motion between the mass and the interaction point is big. We shall validate this assumption at a later stage.

Of course, we do not have access to all of the required regressors. We assume here that one only has access to position, velocity and force (the telerobotic manipulators from [6] and [5] have such limited sensing capabilities, for instance). The rest of the signals therefore need to be computed by numerical differentiation and low-pass filtered to suppress noise.

One should note that the proposed model is only valid for motion tangential to the plane on which the object rests, and while the operator is in contact with the object. In the axis normal to the plane, the mass is not allowed to move (due to the kinematic constraints imposed by the table), so the model equations condense into a simple Kelvin-Voigt contact model (equation 8). This is done because in this axis, the dynamics related to m, k_1 and b_1 will not be excited, which means that, to the estimator, these parameters have fully arbitrary values, which can lead to overfitting problems for the estimator.

$$F = \begin{bmatrix} -\frac{dx}{dt} & -x \end{bmatrix} \begin{bmatrix} b_2 \\ k_2 \end{bmatrix} \tag{8}$$

This makes the identification problem reliant on a specific reference frame, so identification should happen in a reference frame where one axis is normal to the surface, and the other two are tangential to the surface, so the appropriate dynamics model can be applied properly. In this scenario, the surface is horizontally oriented, so this consideration is trivial. However, if the surface were to be arbitrarily configured, this would become an important consideration.

When contact is broken, the model used, which is meant to represent motion through free space, is shown in equation 9.

$$F = 0 \tag{9}$$

3.3 The Full System

All the key concepts to understand the proposed control scheme have now been presented. This subsection presents the actual control scheme. To re-iterate, the system proposed here is a Model-Mediated Teleoperation system capable of reproducing force feedback from an interaction with a flexible object restricted in motion to a flat, horizontal surface.

In the proposed scheme, the master device sends a position command to the telerobot system. The telerobot system then interacts with the environment by carrying out this position command. From this interaction, position, velocity, and force are measured at the interaction point of the telerobotic device. These are sent to the estimator, which uses all these values to compute the appropriate regressors, and from there the parameter estimates.

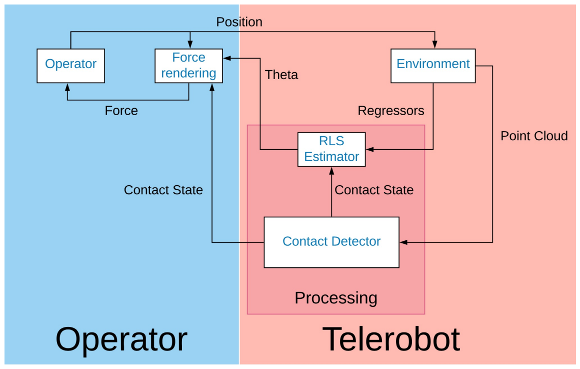

Figure 4: The full impedance reflection and point-cloud-based contact detection control scheme

In parallel to this, a contact detector uses a point cloud of the surface, and the proxy-based point cloud interaction system proposed in [29] to detect whether or not the operator is in contact with the surface. The parameter estimator saves its states and ceases operation while contact is broken, as it does not need to estimate any parameters as our model for free space is non-parametric.

The parameter estimates are then sent to the force renderer, which embodies an instance of the model structure. As such, it also switches to a zero-force model (equation 9) when contact is broken. If the operator is in contact, it also takes in the operator's position, uses it, and the parameters to solve the model ODE for the appropriate output force.

This force is then transferred to the operator.

4 Results and Discussion

This section provides some experimental validation to the claims made thus far. To reiterate, the principal claim is that the proposed system provides accurate MMT-based force feedback for environments composed of a flexible mass on a surface, exhibiting flexible dynamics, and friction relative to the surface.

4.1 The Experimental Set-Ups

Two set-ups are used to support the experiments.

The first set-up is a full implementation of the system described in section 3.3. However, instead of a dynamic environment consisting of a position-tracking controller, manipulator, a surface and a flexible object, position commands are sent directly to the surface of an in-simulation object (unless contact with the object has been broken), resting on an in-simulation surface. The flexible object is simulated using the model structure dynamics from section 3.2 with the addition of static friction modeled with a Karnopp model [18], as well as coulomb friction. The mass is set to 1 Kg, k_1 is set to 0, k_2 is set to 500 N/m, b_1 and b_2 are set to 60 Ns/m, and the Karnopp parameters used are $F_s = 0.1$N, $F_d = 0.05$N and $v_d = 0.005$ m/s. The setup is used in experiments 1 and 3. The high-level software implementation (in ROS [11]) of this setup is described in figure 5.

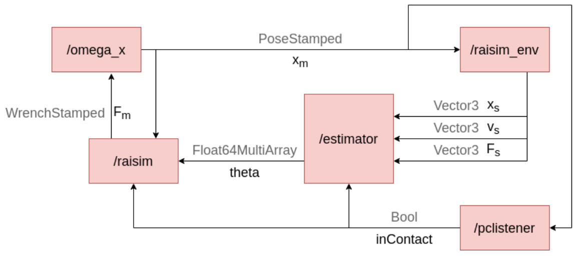

Figure 5: Software Set-Up for experiments 1 and 3

The second set-up consists of only part of the impedance-reflection system. The operator sends a position to the telerobot side. A motion-tracking controller ensures the position command is carried out by a Franka Panda Emika ([5]) robot arm. The robot arm can then interact with a flexible object on a table. The object weighs 459 grams and its flexible contact surface is made of Dragonskin 10 ([9]). Interaction force, position, and velocity are measured and sent to an RLS estimator as described in section 3.2. This setup is used in experiment 2. The high-level software implementation of this setup (in ROS [11]) is described in figure 6. The test object is shown in figure 7.

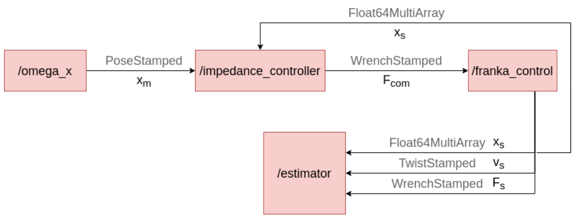

Figure 6: Software Set-Up for experiment 2

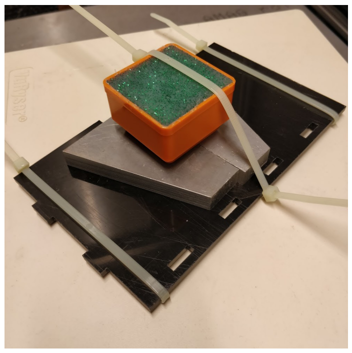

Figure 7: Flexible object used for experiment 2

4.2 Experiment 1

Experiment 1 evaluates whether the designed system can work with a simulated environment. To this end, the proposed system is made to interact with an in-simulation instance of a flexible object on a horizontal surface. The interaction involves moving the object along the table, as well as making and breaking contact with it. The input position enforced by the operator can be seen in figure 8. The onset of force feedback and the start of the experiment follows a wind-up phase where the system is excited enough for the estimator to gain enough information about the system.

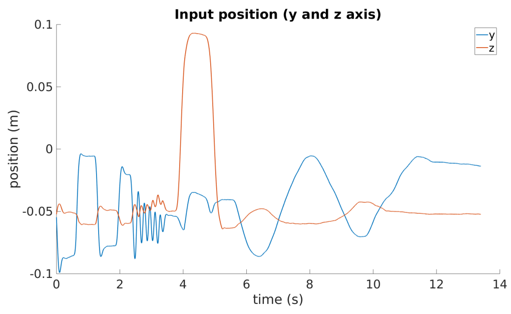

Figure 8: Input position tangential (y-axis) and normal (z-axis) to the surface of the table

Figure 9 shows measured and rendered force for this experiment, in an axis tangential to the surface.

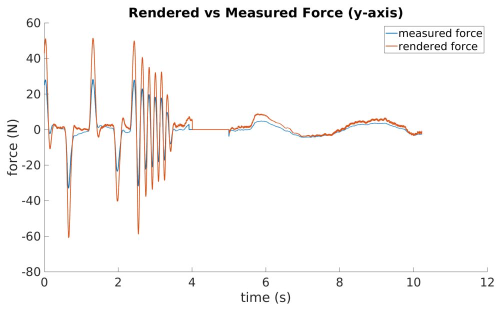

Figure 9: Measured and Rendered force for experiment 1

We notice that the tracking contains significant errors. For excitations containing high frequencies (i.e. the systems behaviour before t=4s), there can be up to a 100% relative error in the tracking. For lower frequencies, this tracking error is better, but can still reach up to 30%.

This is somewhat undesirable behaviour. To have a better idea of the source of this error, we look at two figures. Figure 10 shows the rendered and estimated force for experiment 1, as well as the estimator's estimate for force, obtained by multiplying the regressors by the latest parameter estimates. Figure 11 shows the same thing, but for a modified version of experiment 1 where the parameters are frozen in time after 5 seconds (it is assumed the estimate is sufficiently accurate by then). It should be mentioned that while the results from figure 11 are from an experiment that follows the same protocol as experiment 1 (aside from the parameter freezing), they are from a different run, and are a response to a similar, but not directly comparable input position.

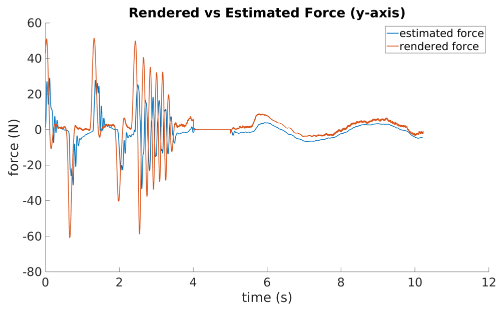

Figure 10: Estimated and Rendered force for experiment 1

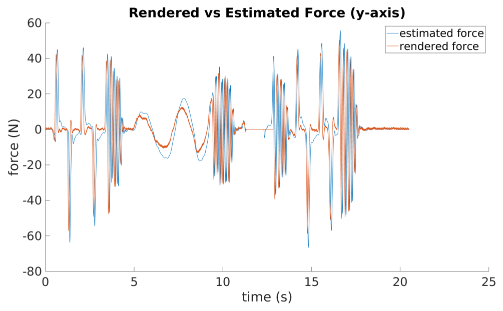

Figure 11: Estimated and Rendered force for experiment with no variation with time of the parameters

As can be seen from these graphs, when the parameters are imposed to be constant with time, the estimated force, that typically follows the measured force quite well, now starts to exhibit the same behaviour as the rendered force. Especially for the excitations containing high frequencies, it should be noted that the magnitude of the spikes match up in figure 11. This additional experiment shows that even in the state of operation where the parameters are allowed to move with time, the force renderer is embodying a model where the parameters are not allowed to change over time. The physics engine used to render force makes an instantaneous constant mass assumption, so the error cannot be due to dynamics dependent on the rate of change of mass [10]. If the mass were allowed to change, there should be a $v\frac{dm}{dt}$ term in the equations of motion, but seeing as neither the estimator nor the renderer model it, this cannot be cause for a problem. Further study is required to understand the precise mechanism behind what is going on, here.

4.3 Experiment 2

Experiment 2 is designed to test if the estimator can accurately capture the dynamics of a real flexible object sliding on a surface. In this experiment, we make a Franka Panda Emika robot [5] interact with a mass weighing 459 grams, topped with a flexible contact surface of Dragonskin 10 [9]. Figure 12 shows the interaction position over the runtime of the experiment. Measured force and estimated force (the parameters multiplied with the regressors) are measured here, as well as the time it takes for the parameters to form a stable configuration. In this instance, a stable configuration is a configuration for which eigenvalues of the system computed using the parameters have a negative real part. Figure 13 shows a plot of this over the runtime of the experiment, in an axis tangential to the surface (the y-axis from figure 12).

Here, the error between measured and estimated force is lower even than the rendering error in experiment 1. There are some deviations, but the tracking error seems to be about 5% of the RMS value of the measured force, which is

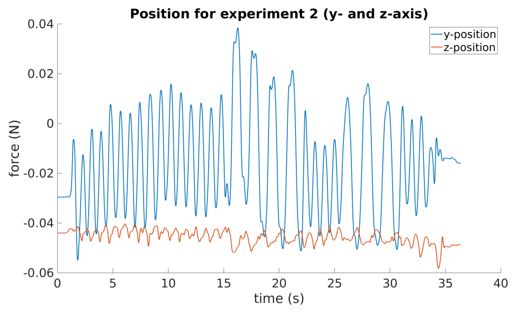

Figure 12: Input position in experiment 2

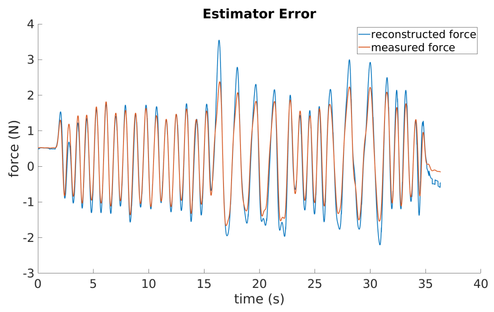

Figure 13: Measured and Estimated Force of the estimator in experiment 2

of the same order as the high-frequency tracking error seen in experiment 1. This validates the assumption made in equation 7. It should also be mentioned that the estimated parameters reach a configuration where the eigenvalues are stable 16.1 seconds into the experiment.

This tells us that the model structure is sufficiently descriptive to capture the dynamics of a flexible body moving with friction along a surface, with a relative error of 5%, and that the parameters produced by this estimator are compatible with an impedance reflection system, as they will not destabilise the force renderer on the operator side.

4.4 Experiment 3

Experiment 3 tests the system's robustness to time delays. Essentially, experiment 1 is performed once more, in much the same manner, except that a static 400 millisecond communication delay is imposed on the communication between operator and telerobot. Specifically, the sending of position from operator to telerobot, and the sending of impedance parameters and contact state from telerobot to operator are the signals that are delayed. As in experiment 1, measured force and rendered force are recorded. Figure 14 shows the measured force, as well as the rendered force shifted back in time by 400 ms. In this way, given that the rendered force is expected to lead the measured force by 400 ms, we expect the plots on the figure to be similar. As with experiment 1, the experiment starts after a wind-up phase.

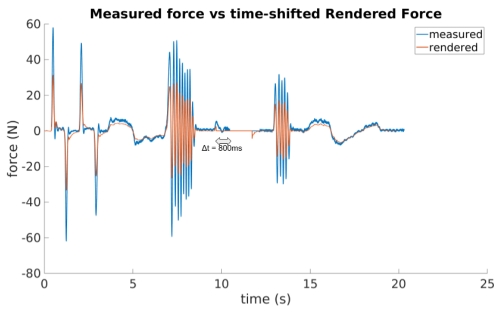

Figure 14: Measured and Rendered force in experiment 3

In the graph, we notice that the measured and rendered force essentially exhibit the same qualities as in experiment 1, except that the rendered force leads the measured force by 400 ms. One notes the same kind of poor low-frequency tracking and good high frequency tracking. What is important to notice is that when making or breaking contact with the object, there is a 400 ms delay between the measured force responding to the change and the rendered force doing so as well. It appears as 800 ms on the graph because of the backward time shift applied to the rendered force that was described in the previous paragraph.

The behaviour of the renderer when no model switching needs to occur is as good as we can hope for. Nothing changes except for the delay between rendered and measured force. However, the reaction to model switching poses a problem. Taking into account the delay in the input position being sent to the telerobot side, this means there is a full round-trip worth of delay between a model switch being carried out by the operator and seeing the effects of this model switch in the force feedback. The fact that this issue only gets worse with increasing time delay is a problem.

A solution to this could be to have two contact detectors on both the operator and telerobot side. The new operator-side contact detector would simply work with time-delayed data. The one on the operator side would send the contact signal to the renderer, and the one on the telerobot side would send the contact signal to the estimator. This would likely bring its own problems, but at least the operator would see appropriate force feedback from model switching that would be less tied to time delay.

5 Conclusions and Recommendations

The purpose of this research was to develop a system capable of providing accurate MMT-based force feedback for environments composed of a flexible mass on a surface, exhibiting flexible dynamics, and friction relative to the surface. This has been partially accomplished, here. The system still shows poor performance, especially at higher frequencies, and does not deal very well with model switching under high time delays. These two problems aside, performance of the system seems adequate.

Important further work will involve updating the design of the contact detector to make it more robust to time delays, exploring the source of error in the renderer further, and testing the full system designed here on a real-life environment.

A Appendix: Contents

This foreword has for purpose to present how the following sections relate to the main body of this document.

The first appendix explains the theory of impedance reflection in more detail than in the main body. It also discusses some complications involved in the practical implementation of impedance reflection systems.

Appendix C deals with explaining the actual mechanics underpinning pcbMMT. The actual algorithm for moving the proxy is described in detail, and, as with the previous appendix, some practical concerns are shown as well.

The next appendix goes in depth into the functioning of every component from 3.3. It presents what each component is doing at the low level, and it highlights points of interest in the design that are relevant to the discussion of the results.
Appendix E details a short feasibility study that justifies the choice of the Raisim physics engine as the back-end for the force renderer.

Appendix F goes into more detail on all the experimental set-ups used in the experiments conducted over the course of the research. It presents some of the equipment used, and shows how the set-ups are implemented in software.

Appendix G elaborates on the results and the discussion of the experiments.

Finally, Appendix H presents a more detailed conclusion, as well as a portion dedicated to further work that can be conducted based on this research.

B Appendix: Impedance Reflection

We start our description of Impedance Reflection with the operator device reading out a position command from the operator. The operator device is typically actuated by motors, so it will be able to impose an effort (force, torque, wrench) on the system it interacts with, and read out its flow (linear velocity, angular velocity, twist), as well other values related to its motion (positions, accelerations). This means that the input to a operator device will usually be some sort of value related to motion. We use position here as a manipulator will usually have a position readout at the very least, and it is used frequently in literature on impedance reflection ([27], [14]).

This position is then sent to the telerobot side, through a time delay. Ideally, this position command would then be imposed straight onto the environment. However, much like on the operator side, telerobotic systems will typically only be able to communicate in a effort-out, flow-in causality with the environment. So, a controller is needed to make sure the manipulator tracks the position command sufficiently well. The appropriate controller design to use here is very application-specific, and so will not be mentioned any further.

This allows the operator to dynamically excite the environment. However, force feedback still needs to be sent to the operator for this to be a viable haptic teleoperation system.

To this end, an array of forces, positions, and their derivatives are read out from the environment. These can then be used to compute dynamic and geometric parameters of an environment. As an example, [27] uses the model structure for the environment shown in equation 10.

$$F = k\left(x - x_r\right) \tag{10}$$

This model contains a geometric parameter x_r indicating the rest position of a spring, and a dynamic parameter k indicating the stiffness of the spring. Assuming the environmental dynamics can be described sufficiently well by this model, two sets of readouts of force and position would give you two equations with two unknowns that could then be solved for the parameters. Exactly how the parameters are solved for using the readouts and the model structure is highly variable. [30] provides a good description of the different "parameter estimation" algorithms used. It should be noted that these model structures typically relate force or torque (or one of their derivatives) to position (or one of its derivatives), so they can usually be rearranged into a ratio of effort and flow. In this case, the estimated values form a parametrisation of the environment's impedance.

The estimated impedance parameters are then streamed over another time delay to the operator side. On the operator side, there is a force renderer. This renderer will typically be another instance of the model structure. Force is computed from the parameter estimates that have been sent to it from the telerobot side, as well as the motion profile of the operator device. To continue the example from [27], if one knows k and x_r from the parameter estimator, and can read x from the telerobotic device, these can be used to compute a force to be rendered to the operator.

The idea is that the operator then receives haptic feedback without any time delay, and if the model structure is sufficiently descriptive of the environment, this force feedback will be close the interaction force on the operator side.

B.1 Stability

Such systems are stable only within a certain set of constraints. We take a look at what these are in this section.

First of all, impedance reflection assumes that the estimator works reliably and accurately. This is important as a lack of accuracy in these parameters can result in instabilities. The reasons as to why an estimator might output incorrect parameters will be explored later, but they are indeed a consideration when designing these systems. A simple example of when this becomes a problem is if the model structure is a mass-spring-damper system. Here, the estimated values of mass, spring constant, and damping coefficient need to be such that the eigenvalues of the system all have negative real part. If not, then the system will destabilise quickly.

Furthermore, this stability analysis assumes that the rate of change of the parameters is sufficiently low. Indeed, parameters changing over time will change the energy inside the force renderer, so linear stability analyses such as eigenvalue-based methods are not sufficient to guarantee stability. So, if the rate of change of these parameters is high, stability is no longer guaranteed.

B.2 Practical Considerations

Here, we take a look at some extra problems that arise for certain specific implementations of impedance reflection systems that are relevant to this research.

Firstly, there is the matter of transient dynamics. This is only relevant to model structures that take the form of differential equations. In general, a fixed solution to these differential equations requires initial conditions. If the initial conditions used by the renderer on the operator side are different from those in the environment, then the part of the rendered force that depends on the initial conditions (termed the *transient response*) will be different from the actual force on the telerobot side, resulting in inacuracies.

Secondly, there is the matter of estimator design. Mismatches between the model structure and the perceived dynamics of a system should be treated with care. A model that underfits the perceived dynamics of the system will obviously perform poorly as it cannot capture all the effects that are present in the measurements. However, an overfitted model can also pose problems. Overfitting can happen either if the model captures effects that are unlikely to be measured at any point, or, in some cases, when the environment is only excited in certain specific use cases where some of the effects cannot be measured. For example, if the environment exhibits a vibration mode that is only measurable in a certain frequency range, and the operator only excites the environment at frequencies outside this range, the model will overfit the perceived dynamics. Overfitting results in high variability in the parameter estimates, which means that in general, the parameter estimates will not be accurate. If these parameters populate the force renderer, they can still produce accurate force feedback, but only if the parameters result in a stable system, which is not always the case.

C Appendix: Point-Cloud-Based Model Mediated Teleoperation

As mentioned before, pcbMMT allows for the rendering of force from a point cloud. Essentially, the input position prescribed by the operator to the operator device is followed by a proxy sphere, that has a movement protocol which does not allow it to pass through surfaces sampled by a point cloud.

Force is then rendered as shown in equation 11, where x_{proxy} is the position of the proxy, x_{HIP} is the input position (stands for Haptic Interaction Point), and k is a sufficiently high spring constant that ensures minimal separation between the proxy and the input.

$$F_r = k\left(x_{proxy} - x_{HIP}\right) \tag{11}$$

The proxy itself consists of three spheres, of radius r_1, r_2, and r_3 respectively, centered around the position of the proxy. The use of these spheres will be explained at a later stage.

The movement protocol for the proxy is a multi-stage process, and proceeds as follows:

In a first step, an estimate is made for the point cloud's local velocity relative to the HIP, so that the point cloud cannot pass over the proxy in one time step without the proxy being able to register contact with the point cloud. This velocity is determined recursively, correcting at each step by the orthogonal projection of the proxy's last step value onto the surface normal of the point cloud. According to [25], this is a good estimate for the distance the cloud has moved towards the HIP in the last timestep, for sufficiently high sampling frequencies. The values for surface normal and proxy step are determined by other parts of the protocol. We then get equation 12 for this relative velocity $\hat{v}$ at time index k. Here, n represents the surface normal, s the last proxy step, and f_c the operating frequency of the renderer.

$$\hat{v}_k = \hat{v}_{k-1} + \hat{\boldsymbol{n}}_{k-1}^T \boldsymbol{s}_k f_c \tag{12}$$

Then, the proxy is moved by a distance equal to this velocity multiplied by the operating frequency of the renderer so it stays at a similar distance from the point cloud since it was last updated (equation 13).

$$\boldsymbol{s}_k = \frac{1}{f_c}\hat{v}_k \hat{\boldsymbol{n}}_{k-1} \tag{13}$$

After this, the surface normal estimate is computed. It only computes a normal if there are points within r_3. Let there be N points in r_3, and x_i^{pc} be a point within r_3, the normal is computed by normalising the result of equation 14.

$$\hat{\boldsymbol{n}}' = \sum_{i=1}^{N} \frac{x_{proxy} - x_i^{pc}}{\|x_{proxy} - x_i^{pc}\|}\psi\left(x_{proxy} - x_i^{pc}\right) \tag{14}$$

where ψ is the so-called Wendland function (defined in equation 15. This allows the normal calculation to prioritise points closer to the HIP over points further away in a smooth manner.

This means that r_3 should chosen big enough for the estimate to be on average insensitive to the amount of points, but not so big as to hinder its ability to estimate a surface normal for sharp transitions in the point cloud.

$$\psi\left(r\right)=\begin{cases}1 & if\, r\leq r_1\\ \left(1-\frac{r-r_1}{r_3-r_1}\right)^4\left(\frac{4(r-r_1)}{r_3-r_1}+1\right) & if\, r_1<r<r_3\\ 0 & if\, r\geq r_3\end{cases} \tag{15}$$

Next, the so-called *state* of the proxy is determined. If there are any points within the sphere of radius r_2, but none within the sphere of radius r_1, the proxy is *in contact*. If there are points within r_1 of the proxy position, the proxy is called *entrenched*. In all other situations, the proxy is *in free motion*. For the choice of r_1 and r_2, [25] suggests the distance between r_2 and r_1 be equal to or bigger than the magnitude of the noise in the point cloud. r_1 should also be sufficiently small that the proxy can rest close to the point cloud without getting entrenched.

If the proxy is in free motion, the size of the step is determined by the minimum over all d_i that solve equation 16, for all points in the point cloud, unless the value of $\|u_k\|$ is smaller than the solution, in which case it should be equal to the former. Here, u_k is the vector pointing from the proxy to the HIP, and i denotes the index of a point in the cloud. This step size basically has the proxy follow the HIP, unless a point in the cloud is in the way. If it is, the proxy stops $\frac{r_1+r_2}{2}$ away from the point.

$$\frac{r_1+r_2}{2}-\|x_i^{pc}-x_{proxy,k}-d_i\frac{\boldsymbol{u}_k}{\|\boldsymbol{u}_k\|}\|=0 \tag{16}$$

x_i^{pc} denotes a point in the point cloud.

Then, the proxy moves by the minimal d_i, denoted d_k in the direction of the HIP:

$$\boldsymbol{s}_k=d_k\frac{\boldsymbol{u}_k}{\|\boldsymbol{u}_k\|} \tag{17}$$

If the proxy is entrenched, the step size is determined by the maximum over all d_i that solve equation 18. This is the smallest step the proxy can make to remove all the points in the cloud from the r_1 sphere.

$$\frac{r_1+r_2}{2}-\|x_i^{pc}-x_{proxy,k}-d_i\hat{\boldsymbol{n}}_k\|=0 \tag{18}$$

Then, the proxy moves by $d_k=max_i d_i$ along the surface normal estimate:

$$\boldsymbol{s}_k=d_k\hat{\boldsymbol{n}}_k \tag{19}$$

If the proxy is in contact, the step size depends on how big $\|\boldsymbol{u}_k\|$ is. If it is smaller than r_1, then d_k is equal to $\|\boldsymbol{u}_k\|$ scaled by a factor $\gamma < 1$ (equation 20). Otherwise, it is the value of r_1 scaled by a factor $\zeta < 1$ (equation 21). This scaling is such that the proxy can never move so far that it entrenches itself on the other side of the cloud, thus punching through.

$$d_k = \gamma \|\boldsymbol{u}_k\| \tag{20}$$

$$d_k = \zeta r_1 \tag{21}$$

The direction in which the proxy moves depends on whether the operator is pushing into the point cloud, or pulling away from it. If the operator is pushing in, the proxy moves along the orthogonal projection of $\boldsymbol{u}_k$ onto the plane denoted by the surface normal estimate (we call it $\boldsymbol{u}_{k,p}$), otherwise, the proxy just moves along $\boldsymbol{u}_k$. Equation 22 then shows the value of the step.

$$\boldsymbol{s}_k = \begin{cases} d_k \frac{\boldsymbol{u}_{k,p}}{\|\boldsymbol{u}_{k,p}\|} & if\ \hat{\boldsymbol{n}}_k^T \boldsymbol{u}_k > 0 \\ d_k \frac{\boldsymbol{u}_k}{\|\boldsymbol{u}_k\|} & if\ \hat{\boldsymbol{n}}_k^T \boldsymbol{u}_k \leq 0 \end{cases} \tag{22}$$

The sign of $\hat{\boldsymbol{n}}_k^T \boldsymbol{u}_k$ denotes whether or not the operator is pushing in or pulling out of the surface.

C.1 Proxy Sticking

Here, we discuss an implementation-specific issue that arises as a consequence of there being high amounts of noise in the point cloud. If the proxy is in contact and the operator suddenly pulls the HIP out of the surface quickly, the amount of steps it will take will be described by equation 23.

$$n = ceil\left(\frac{r_2 - r_1}{\zeta r_1}\right) \tag{23}$$

For high amounts of noise, the gap between r_1 and r_2 will be quite big. Also, ζ needs to be smaller than 1 to ensure no punch-through, and r_1 needs to stay small so that the proxy can rest sufficiently close to the surface during contact. So, the proxy needs to rest very closely to the surface, in high-noise situations, it can be that n is quite high, which means it can take significant amounts of time for the proxy to catch up with the HIP, which the operator will feel as the system trying to stick to the surface.

C.2 Analytically solving the shortest distance problem

The proxy movement algorithm described here involves finding minimal values of d_i that solve an equation on two separate occasions (see equations 16 and 18). While such values for d_i could be found by numerical search methods, what is shown in this section is that solving equations of this type essentially involves solving quadratic equations, which can be easily solved analytically.

Both equations 16 and 18 can be written in the form of equation 24. a represents a real-valued scalar, and $\boldsymbol{b}$ and $\boldsymbol{c}$ represent real-valued vectors. In the following, "$\| \cdot \|$" represents the L2-norm.

$$a - \|\boldsymbol{b} - d_i \boldsymbol{c}\| = 0 \tag{24}$$

This can be rewritten as shown in equation 25.

$$a - \sqrt{\left(b_x - d_i c_x\right)^2 + \left(b_y - d_i c_y\right)^2 + \left(b_z - d_i c_z\right)^2} = 0 \tag{25}$$

Squaring, we get equation 26.

$$a^2 - \left(b_x - d_i c_x\right)^2 - \left(b_y - d_i c_y\right)^2 - \left(b_z - d_i c_z\right)^2 = 0 \tag{26}$$

Expanding and rearranging, we get equations 27 and 28.

$$-\left(c_x^2 + c_y^2 + c_z^2\right) d_i^2 + 2\left(b_x c_x + b_y c_y + b_z c_z\right) d_i + a^2 - \left(b_x^2 + b_y^2 + b_z^2\right) = 0 \tag{27}$$

$$\|\boldsymbol{c}\|^2 d_i^2 - 2(\boldsymbol{b} \cdot \boldsymbol{c}) d_i + \|\boldsymbol{b}\|^2 - a^2 = 0 \tag{28}$$

This leaves us with a quadratic equation that can be solved for d_i:

$$d_i = \frac{2\left(\boldsymbol{b} \cdot \boldsymbol{c}\right) \pm \sqrt{\left(\boldsymbol{b} \cdot \boldsymbol{c}\right)^2 - 4\left(\|\boldsymbol{b}\|^2 - a^2\right)\|\boldsymbol{c}\|^2}}{2\|\boldsymbol{c}\|^2} \tag{29}$$

Negative and non-real values can be rejected, and what is left gives us the solutions to these equations.

D Appendix: The Full System Design

This section elaborates on important components in the design detailed in subsection 3.3.

D.1 The environment

The model used for this Model-Mediated Teleoperation system only supports certain specific configurations for the environment. The type of environment that we concern ourselves with in this work is elaborated on here.

As mentionned in subsection 3.3 of the main body, the system only works with an environment composed of one (flexible) object restricted to move on a flat, horizontal surface. Friction and compliance between the surface and the object is allowed, and the object is allowed to move along the surface with no further restrictions.

D.2 The estimator

The process flow for the estimation process is shown in figure 15.

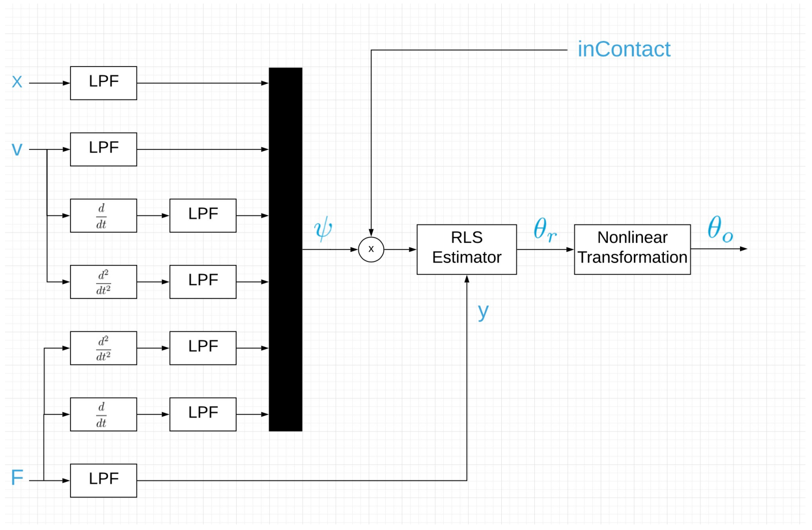

Figure 15: The estimation process

For several robotic systems that could be used as telerobotic devices such as the Franka Emika [4] and the KUKA LWR (supported at the low-level by Stanford's FRI Library [6]), the only data available to the user is position, velocity, and force data. Therefore, given that the estimator requires higher order derivatives of these, as well, the signals need to be differentiated. This is done here by a backward Euler technique. They are then low-pass filtered to suppress noise.

The exact value and order of these will depend on the sensors used to generate the data. The non-differentiated signals are also differentiated too to compensate for any potential phase lag in these filters.

Special care needs to be given to the design of the aforementioned filters. The cutoff frequency needs to be such that enough noise can be filtered out to make the differentiated signals usable, but the if the filters distort the information-containing part of the signals, then this can lead to unpredictable behaviour for the estimator, as it would then be working off of distorted data.

The signals are then concatenated into the ψ vector, except for the force, which is the model output. ψ is then multiplied by 1 if `inContact` is true, and 0 if not. Setting ψ to 0 is a means of pausing the estimator without suspending the runtime of the difference equation solver performing the parameter estimate. This works because $\psi = 0$ results in the value by which the RLS estimator's internal states are incremented being set to 0.

The estimation is then performed, and then the regrouped parameters for which the estimator solves are converted the the real parameters by back-solving the system of equations shown in 30, analytically.

If θ_r denotes the regrouped parameter vector $[a\, b\, c\, d\, e\, f\, g]^T$ that the estimator outputs, then they are related to the parameters in Figure 3, and coulomb friction F_c by equation 30.

$$\begin{cases} a = \frac{mb_2}{k_1+k_2} \\ b = \frac{mk_2+b_1b_2}{k_1+k_2} \\ c = \frac{k_1b_2+k_2b_1}{k_1+k_2} \\ d = \frac{k_1k_2}{k_1+k_2} \\ e = \frac{m}{k_1+k_2} \\ f = \frac{b_1+b_2}{k_1+k_2} \\ g = \frac{k_2}{k_1+k_2}F_c \end{cases} \tag{30}$$

A parameter vector composed of m, k_1, k_2, b_1, b_2, and F_c is then sent on by the estimator process.

D.3 The contact detector

The contact detector uses the pcbMMT technique described in appendix C and a point cloud of the surface of the environment to detect whether the operator is in contact with the object.

The point cloud and the input position (noted in Figure 16 as x_{HIP}) are fed to the proxy movement algorithm. The proxy is then moved according to the stipulations from appendix C. To reiterate, the proxy movement algorithm moves the proxy such that it follows the input position as closely as possible without also penetrating the surface. When in contact, points in the cloud rest between r_1 and r_2, whereas outside of contact, there are no points within that range.

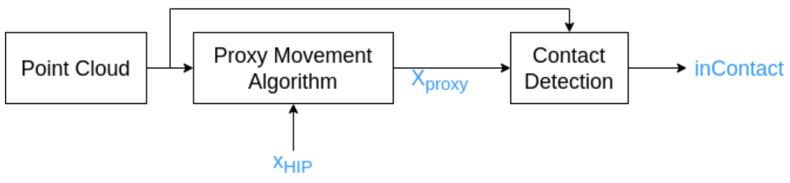

Figure 16: The contact detector

With this in mind, in a second step, the system checks whether any points in the cloud are within r_2 of the proxy. If so, the contact detector outputs `true`, and if not, it outputs `false`.

D.4 The force renderer

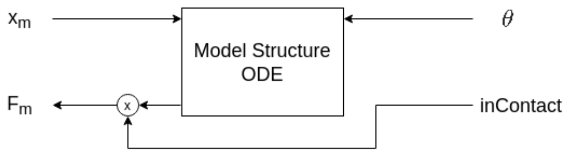

Figure 17: The force renderer

The force renderer solves the equations of motion of the model described in figure 3 by numerical integration. The only input to the differential equation is the input position. The parameters identified by the estimator are sent to the renderer, and their value in the equations of motion is updated as soon as a new estimate comes in. Moreover, if contact is broken, the output force is set to zero to reflect the zero-valued impedance of free space.

One should note that solving an ODE for the force numerically and in real time incurs errors that compound over time, as explicit solvers need to be used. These errors depend on the accuracy of the numerical solver, as well as on the operating frequency of the solver. The renderer should therefore run as fast as is feasible.

E Appendix: Choosing a Numerical Integration Library for the Force Renderer

As previously mentioned, the type of solver used to solve the model ODE is an important part of ensuring the accuracy of the rendered force. Here, we test 5 different physics engines capable of solving the dynamics of the model structure for accuracy.

Four of the physics engines used here are the ones supported by the Open Source Robotics Engine Gazebo simulation tool: Open Dynamics Engine ([8]), Bullet ([1]), Simbody ([12]), and Dart ([2]). The last one is Raisim ([10]), a physics engine developed by ETH Zurich's Legged Robotics Laboratory.

The first four physics engines are tested for accuracy by simulating the model of the environment devised in section 3.2 for $k_1 = k_2 = 10$ N/m, $b_1 = b_2 = 1$ Ns/m, and $m = 1$ Kg.

They are fed 2 sine waves of varying frequency as input. The input-output ratio is then recorded and compared to the theoretical value. The selected frequencies are $\omega = 1$ rad/s to cover the low frequency behaviour and $\omega = 4.32$ rad/s as it is the system's natural resonance frequency.

The results for ODE, Dart, Bullet, and Simbody are recorded in figure 18.

x	$\omega = 1$	$\omega = 4.32$
Expected	0.64	4.2
ODE	0.58	0.99
Bullet	1.43	1.15
Simbody	0.65	0.98
Dart	0.35	0.9

Figure 18: Amplitude transfer characteristic of the system for Gazebo's physics engines

The error between simulated response and expected response, especially at the system's resonance frequency, is often of the same order of magnitude as the value of the response, if not higher. This is not adequate.

Raisim is evaluated in a slightly different manner as its selection was made as a reaction to the poor performance of the previous 4. The simulated system has for parameters $k_1 = 0$ N/m, $k_2 = 20$ N/m, $b_1 = 0.1$ Ns/m, and $b_2 = 0.5$ Ns/m.

It is evaluated at $\omega = 1$ rad/s, $\omega = 6.27$ rad/s, $\omega = 10.3$ rad/s, and $\omega = 24.3$ rad/s. The results are recorded in figure 19.

These results are considerably better. The errors are generally at least an order of magnitude smaller than the values themselves. With this in mind, Raisim will be used as the physics engine for the force renderer in the experiments.

x	$\omega = 1$	$\omega = 6.27$	$\omega = 10.3$	$\omega = 24.3$
Expected	0.67	105.93	31.62	21.63
Raisim	0.70	108.85	31.79	21.27

Figure 19: Ampltitude transfer characteristic of the system for Raisim

F Appendix: Experimental Set-Ups

This section describes the set-ups used for the experiments carried out. The physical hardware used is discussed, as well as the software used in the set-ups, and its general structure for each set-up.

F.1 Software

F.1.1 ROS Melodic Morenia

ROS[11] is a middleware system that allows separate executables to run independently of each other and communicate with eachother through a standardised interface. The set-ups described herein use a version of ROS called Melodic Morenia.

Individual executables are packaged into *nodes*. These nodes can output data by *publishing* data to a *topic*. Nodes requiring input data can then *subscribe* to these topics to read the data. Data that is published and subscribed to adheres to specific data formats called *messages*. More information on the types of messages is detailed in [11].

ROS also has some tools for controlling the timing of code execution. In a scenario where other processes are not interfering with the execution of the ROS code (e.g. when running the code on a sufficiently powerful computer), this can be used to ensure a steady operating frequency for the ROS node, which can be used for instance to perform integrations and differentiations.

F.1.2 Raisim

Raisim [10] is an open source dynamics engine. It has functionality for integrating the motion of individual rigid bodies, as well as rigid body mechanisms. It uses the method from [17] for the integration.

It also simulates collision dynamics for objects with arbitrary geometries.

F.2 Hardware

F.2.1 Omega 7

The Force Dimension Omega 7 (figure 20) is a Haptic Controller. A human operator can impose a 6-DOF pose on the handle, as well as set the position of a trigger on the handle. In terms of force the device can output to the operator, it can exert a 3-DOF force on the handle, and a force can be set on the trigger as well. [3]

In the context of this work, only the translational position of the Omega device is read, and only the 3-DOF force exerted on the handle is used in terms of force feedback.

The Omega 7 communicates with the rest of the software with a ROS interface node that publishes the pose of the handle as a `PoseStamped` message, the translational and rotational

Figure 20: Omega 7 Haptic Controller [3]

velocities of the handle as a `TwistStamped` message, and subscribes to a `WrenchStamped` message that contains the translational force meant for output.

F.2.2 Franka Emika

The Franka Emika is a 7-DOF serial manipulator (see figure 21). Its c++ API exposes low level controls that allow developers to set forces and torques at the joint level, and at the end-effector level, and read forces, torques, positions, and velocities at the end-effector and joint level. The end-effector, in the context of this work, is a cone-shaped probe.[5]

Figure 21: The Franka Emika Robot [5]

The Franka Emika interfaces with the rest of the software with a ROS node that subscribes to an end-effector wrench message (`WrenchStamped`), and publishes an end-effector pose and twist (as a `Float64MultiArray` and a `TwistStamped` message respectively).

F.3 The Set-Ups

F.3.1 Set-Up 1

The first set-up is an instantiation of the final design. It uses the Omega 7 to impose a position command of a simulation of the environment in Raisim. The simulation contains a box-shaped object on a flat surface. Flexible properties of this box are modeled by a Kelvin-Voigt model, and when it moves along the surface, it is subjected to linear viscous friction, as well as friction effects defined by a Karnopp model[18]. The input position is the surface of the object.

The operator can make or break contact by distancing themselves from the surface. A point-cloud based contact detector is included as well.

The contact detector uses a virtual, artificially generated point cloud of the environment. It is a flat plane occupying the same height as the box in `/raisim_env`.

ROS is used to link all the modules described above together. Figure 22 describes the ROS structure of the Set-Up. Details on the ROS structure of the nodes is presented too.

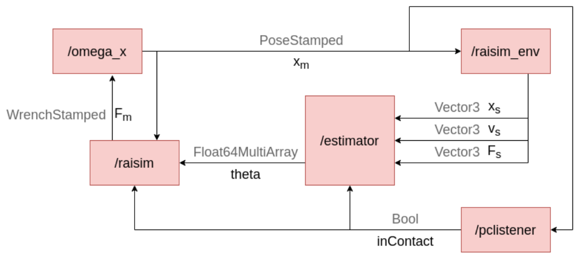

Figure 22: Set-Up 2: ROS Structure

Node:	`/omega_x`
Description:	ROS interface for the Omega 7 haptic device
Subscriptions:	`geometry_msgs/WrenchStamped Fm`
Publications:	`geometry_msgs/PoseStamped xm`

Node:	`/raisim_env`
Description:	Contains a simulation of the environment in Raisim
Subscriptions:	`geometry_msgs/PoseStamped xm`
Publications:	`geometry_msgs/Vector3 xs` `geometry_msgs/Vector3 vs` `geometry_msgs/Vector3 Fs`

Node:	`/estimator`
Description:	Solves the RLS equations for the parameter estimates, and pauses estimation when `inContact` is False

Subscriptions:	`geometry_msgs/Vector3 xs`
	`geometry_msgs/Vector3 vs`
	`geometry_msgs/Vector3 Fs`
	`std_msgs/Bool inContact`

Publications:	`std_msgs/Float64MutliArray theta`

Node:	`/raisim`
Description:	Uses raisim to solve the model structure ODE for force

Subscriptions:	`std_msgs/Float64MultiArray theta`
	`geometry_msgs/PoseStamped xm`
	`std_msgs/Bool inContact`

Publications:	`geometry_msgs/WrenchStamped Fm`

Node:	`/pclistener`
Description:	Detects contact through interaction with a point cloud
Subscriptions:	`geometry_msgs/PoseStamped xm`
Publications:	`std_msgs/Bool inContact`

F.3.2 Set-Up 3

This set-up tests part of the impedance reflection process on a real robot. The omega sends position commands to a position-tracking controller, which takes in those commands, and well as the Franka robot's end-effector pose to send a control wrench to the Franka robot. The Franka robot then sends end-effector position, velocity, and force to the estimator.

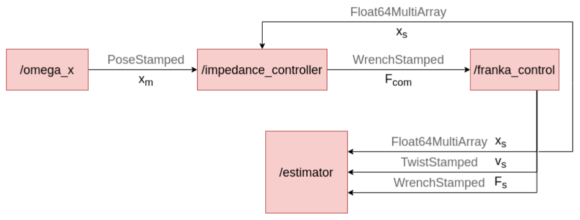

Figure 23: Set-Up 3: ROS Structure

Figure 23 shows the ROS structure of the setup. Also shown are the ROS interfaces for the nodes.

Node:	`/omega_x`
Description:	ROS interface for the Omega 7 haptic device
Subscriptions:	`geometry_msgs/WrenchStamped Fm`
Publications:	`geometry_msgs/PoseStamped xm`

Node:	`/impedance_controller`
Description:	Position-Tracking Controller

Subscriptions:	`geometry_msgs/PoseStamped xm`
	`std_msgs/Float64MutliArray xs`

Publications:	`geometry_msgs/WrenchStamped Fcom`

Node:	`/franka_control`
Description:	ROS Interface for the Franka Emika Robot
Subscriptions:	`geometry_msgs/WrenchStamped Fcom`

Publications:	`std_msgs/Float64MultiArray xs`
	`geometry_msgs/TwistStamped vs`
	`geometry_msgs/WrenchStamped Fs`

Node:	`/estimator`
Description:	Solves the RLS equations for the parameter estimates, and pauses estimation when `inContact` is False

Subscriptions:	`std_msgs/Float64MultiArray xs`
	`geometry_msgs/TwistStamped vs`
	`geometry_msgs/WrenchStamped Fs`

Publications:	`std_msgs/Float64MutliArray theta`

G Appendix: Results and Discussion

This section contains the experiments conducted to validate the performance of the proposed teleoperation system, and a description of their results.

The claim made in this thesis is that this system allows for impedance-reflection based MMT for environments composed of a flexible mass on a surface, exhibiting flexible dynamics, and friction and compliance relative to the surface. The system should also allow the operator to detach from and reattach to the object with accurate force reproduction.

G.1 Experiment 1

Experiment 1 is intended to validate the entire system with an in-simulation environment. It essentially involves manipulating the aforementioned Set-up 1 in various use cases. The value of the mass is set to 1 kg, k_1 is set to 0, k_2 is set to 500 N/m, and b_1 and b_2 are set to 60 Ns/m. For the Karnopp model representing static and coulomb friction, the parameters used are $F_s = 0.1$ N, $F_d = 0.05$ N, and $v_d = 0.005$ m/s.

Input position to the system contains appropriate frequency content to excite the object's highest vibrational mode of 3.6Hz. It is also such that the operator breaks contact with the object and reestablishes it at least once. As the system is being manipulated, the rendered force by the system, as well as the "measured" force sent to the estimator, are measured. The object is mostly moved along the surface following a line aligned with the system's y-axis.

The expectation is that the rendered force tracks the measured force closely, but not perfectly, because the discrepancy between the dynamics of the system model and the environment (notably stiction), and differences in transient dynamics between the virtual environment and the "real" one.

Finally, it should be noted that the low-pass filters used for the estimator in this experiment are 6-th order Butterworth filters with a cutoff frequency of 10Hz.

Figure 24 shows the input position, measured force, and rendered force for the entire runtime of the experiment. The forces look similar, but seem to separate at higher frequencies. Figures 25 and 26 show the same data, but over periods of time where low frequencies and high frequencies are particularly dominant.

To evaluate exactly how frequency dependent this difference is, figure 27 shows the frequency spectrum of the difference of the two signals, plotted on a log-log scale. Indeed, the error stays below 0.25N until roughly 0.3Hz. From then on, the error plateaus at around 1N, which is of the same order of magnitude as the measured signal at those frequencies (figure 28).

Figure 29 shows the measured force used by the estimator, and the estimated force based on the parameter estimate. The error in this plot is noticeably lower than the one in figure 24. This means that the model structure is complex enough to reproduce the dynamics of the environment.

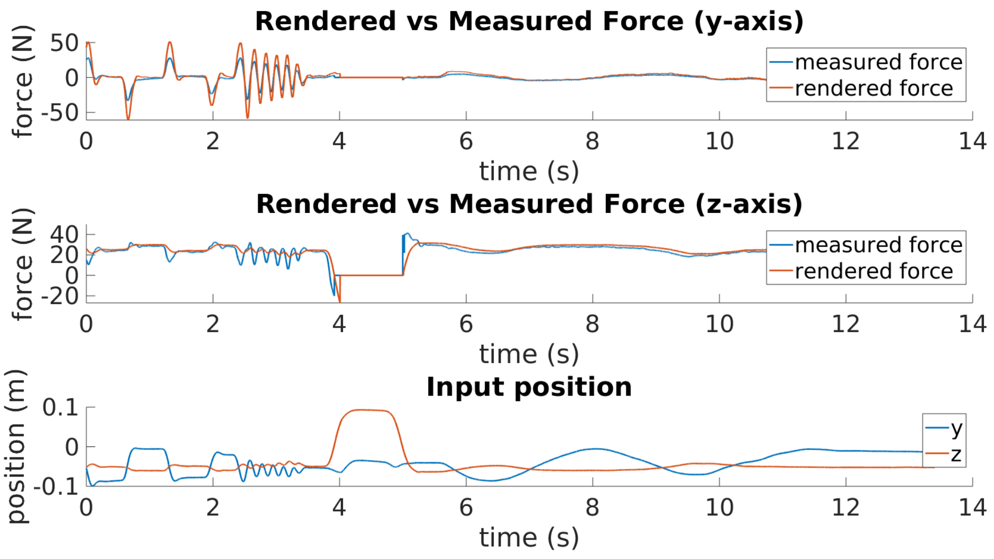

Figure 24: Input position, measured force, and rendered force (y- and z-axis)

To look at the source of the errors we do have, we look at two figures. Figure 30 shows the rendered and measured force for experiment 1, as well as the estimator's estimate for force, obtained by multiplying the regressors by the latest parameter estimates. Figure 31 shows the same thing, but for a modified version of experiment 1 where the parameters are frozen in time after 5 seconds.

These graphs essentially show that when the parameters are imposed to be static with respect to time, the estimated force, instead of staying close to the measured force, starts to look like the rendered force. The spikes visible in the higher frequency excitations now have a similar magnitude in the rendered force as in the estimated force. This means that somehow, the force renderer, even in the situation where the parameters are variable with time, behaves like the parameters are static with time. This means that there is an error in the way the force is rendered.

In a real system like this, a potential source of error would be that there would be force term dependent on the rate of change of the mass in the system. However, raisim itself makes a constant mass assumption in its solver [10], so it is unlikely that anything like that is causing an error in the renderer. Another potential source of error is in Raisim's solver itself. The experiments in appendix E showed that the engine is quite accurate for a physics engine, but its solver is quite complicated. Potentially, a simpler solver library such as [7], which uses traditional solvers like Runge-Kutta solvers, would at least have more predictable performance.

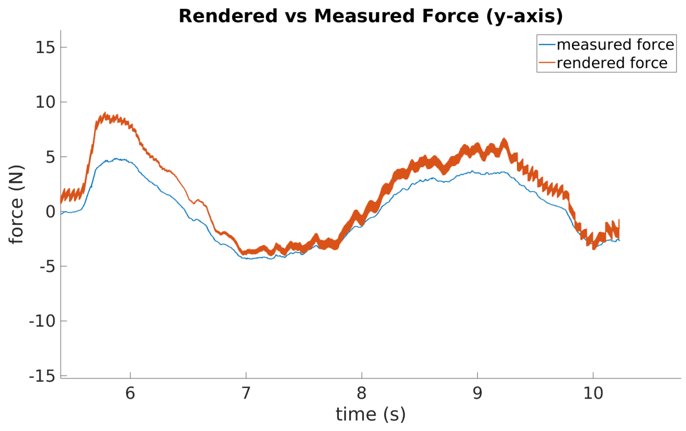

Figure 25: Measured and Rendered Force at low frequencies (y-axis)

G.2 Experiment 2

Experiment 2 is intended to validate the performance of the estimator when provided with a real environment, as opposed to the simulated one used in experiment 1. Here, setup 3 is used with an object on a table weighing 459 grams and with a contact surface of Dragonskin 40 [9], which is able to slide along a table (see figure 32). The idea behind this experiment is to show that the model structure of the estimator is such that stable, accurate parameter estimates can be produced by the estimator from interactions with a real dynamic environment.

The operator manipulates the object by sliding it along the table. What is monitored here is the measured force sent to the estimator, the estimated force obtained by multiplying the estimated parameters by the regressors, and the time it takes for the estimated parameters to begin representing a stable system. The input position of the aforementioned manipulation is shown in figure 33.

The expectation is that the output error of the estimator should be low throughout the course of the experiment. Some discrepancies between measured and reconstructed force should be noted, notably from some of the friction effects in the environment not being well represented in the model. We expect it to take a finite amount of time for the parameters to converge to a state where they represent a stable system.

Measured and rendered force over time for the experiment are shown in figure 34. The parameters form a stable configuration after 16.1 seconds.

The mean value of the estimator error over the course of the experiment is 0.045 N. This is roughly 5% of the RMS value of the measured force over the course of the experiment. This error is of the same order of magnitude as the in-simulation estimator output error, so this is as expected.

The size of the error together with the fact that the parameters do indeed form a stable

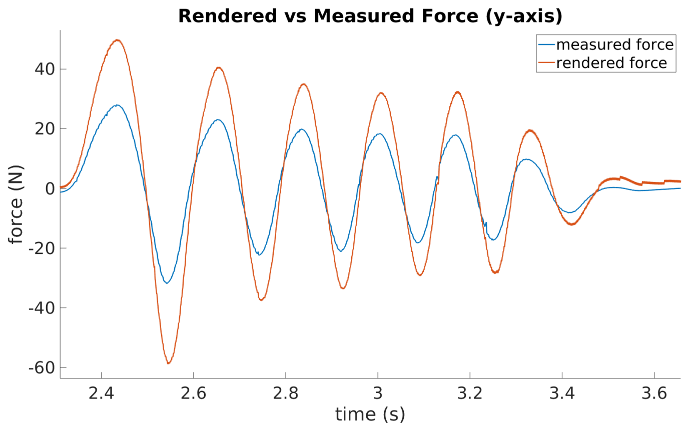

Figure 26: Measured and Rendered Force at high frequencies (y-axis)

configuration after sufficient excitation seem to suggest that the model structure can indeed model flexible objects capable of sliding along a surface, and produce parameters that are usable by an impedance reflection system.

The time it takes for parameters to represent a stable system could potentially pose a problem for the future, however. Of course, this experiment was designed to see if the parameters would converge at all, not to see how quickly they possibly could. Indeed, 16.1 seconds seems to actually correspond to a change in the way the system is being excited (figure 34 shows an increase in the force amplitude at around 16 seconds). Where the problem lies is in the fact that in general, it is still likely that it will take a significant amount of time for parameters to reach a stable value. This means that any time a new object is encountered, the force feedback needs to be provided by other means, seeing as turning on the renderer while the parameters are unstable will result in unpredictable behaviour for the system. Alternatively, an informed guess regarding the values of the parameters could be made, using visual or other external data.

G.3 Experiment 3

Given that the proposed system is a technique for teleoperation, it is important that it be robust to time delays. This is what the third experiment is intended to test. Essentially, experiment 1 is repeated, but with a static 400 millisecond communication delay between operator and telerobot.

This means using set-up 1 again, but 400 ms delays are imposed on `/raisim_env` 's subscription to x_m, as well as on `/raisim` 's subscriptions to *theta* and *inContact*. The input position fed to the system by the operator is to be similar in quality to that of experiment 1. Once again, measured force and rendered force are measured.

If the system is behaving as it should, the rendered force should exhibit the same behaviour as in experiment 1, except that it should lead the measured force by 400 ms. Figure 35 shows the

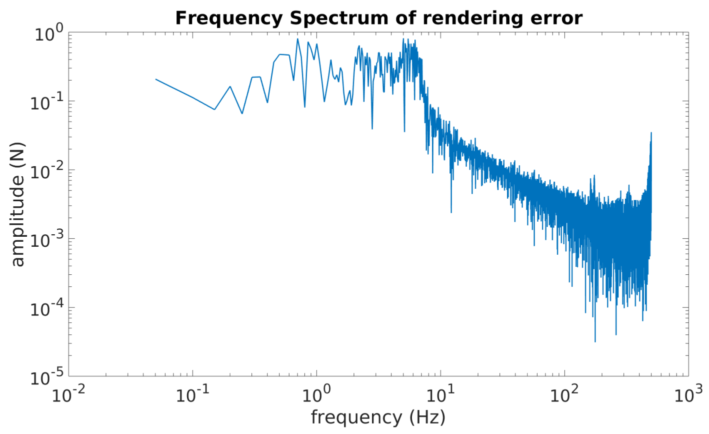

Figure 27: Frequency spectrum of error (y-axis)

measured and rendered force over the course of this experiment, with the rendered force shifted backwards in time by 400 ms. We therefore expect the graphs to appear similar, in the ideal case, since the rendered force normally should lead the measured force by 400ms.

The rendered force seems to follow the measured force in the same manner as it did in experiment 1. Low frequency tracking is rather inaccurate, but high frequency tracking is a lot better. In the graph one notices a 800 ms delay between the measured force becoming zero due to the operator losing contact and the rendered force doing the same. This is to be expected as the contact state is delayed in time by 400 ms, so the rendered force should become zero 400 ms later than the measured force. This appears as 800 ms on the graph because of the 400 ms time shift communicated to the rendered force.

This last effect is undesirable as it means a full round-trip's worth of time delay (considering position commands need to be sent to the telerobot side over the time delay too) between a model-switching action on the operator side (making or breaking contact with an object) and its effect on the force feedback. This presents a bottleneck in the system's robustness to time delay, as when there is no model switching, there is practically no measurable effect due to time delay in the force feedback.

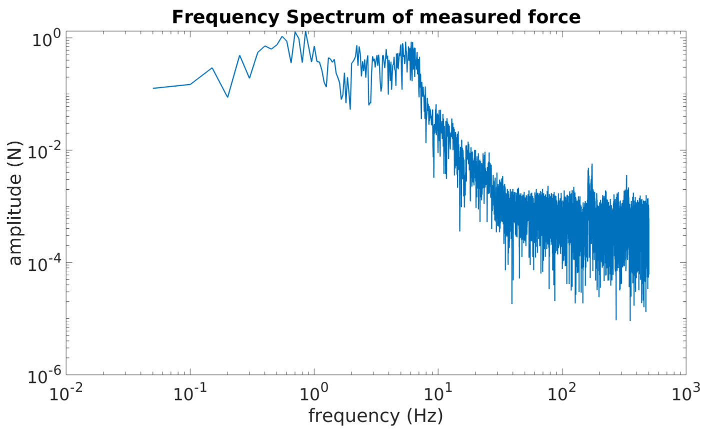

Figure 28: Frequency spectrum of measured signal (y-axis)

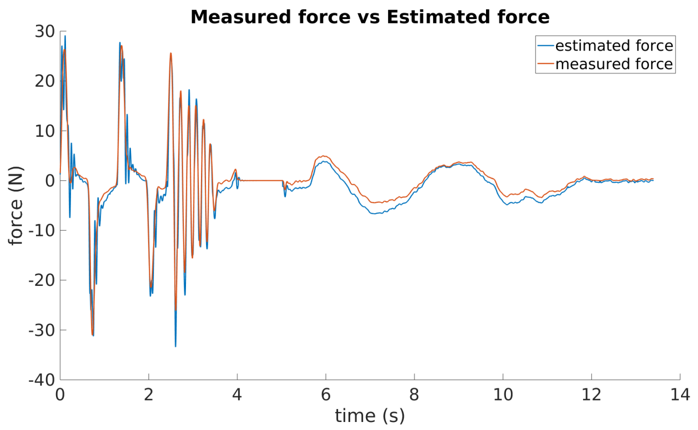

Figure 29: RLS Estimator Error

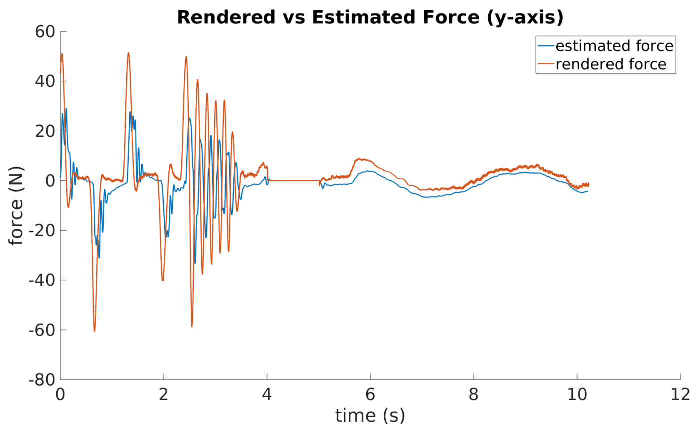

Figure 30: Estimated and Rendered force for experiment 1

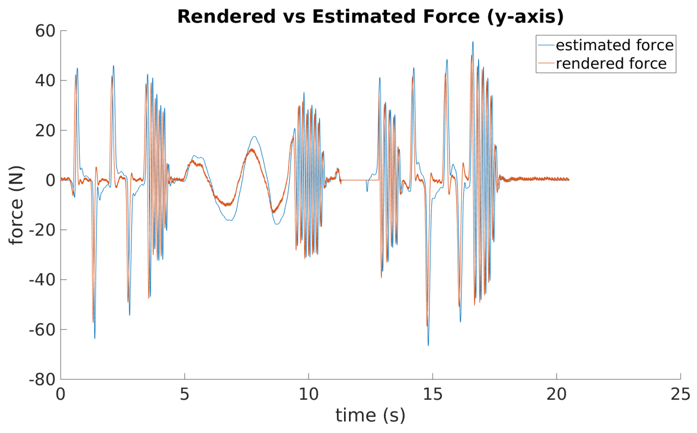

Figure 31: Estimated and Rendered force for experiment with no variation with time of the parameters

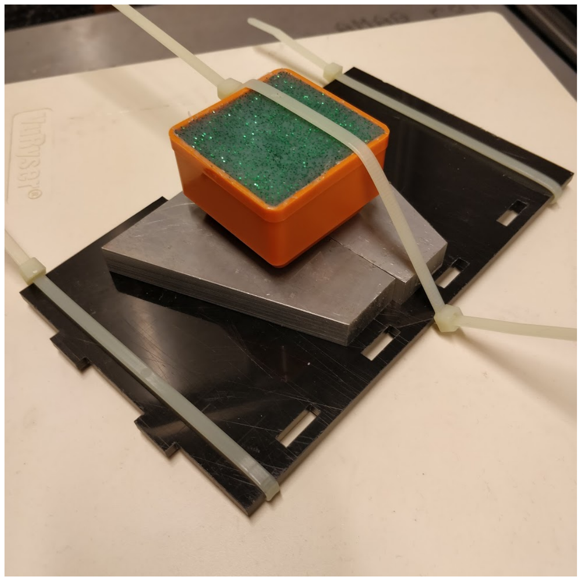
Figure 32: The test object used for Experiment 2

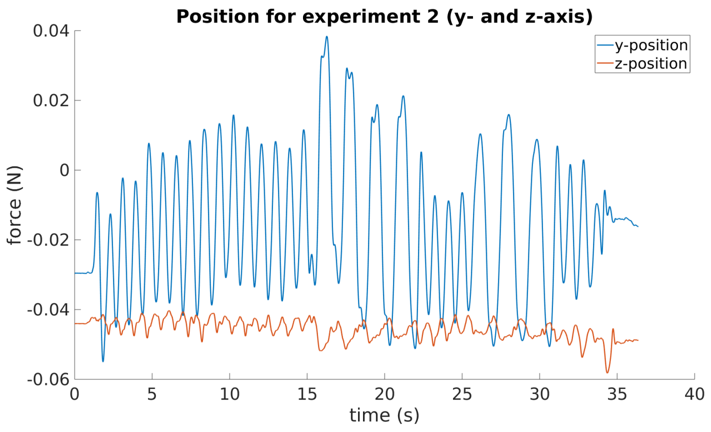

Figure 33: Input Position (y- and z-axis)

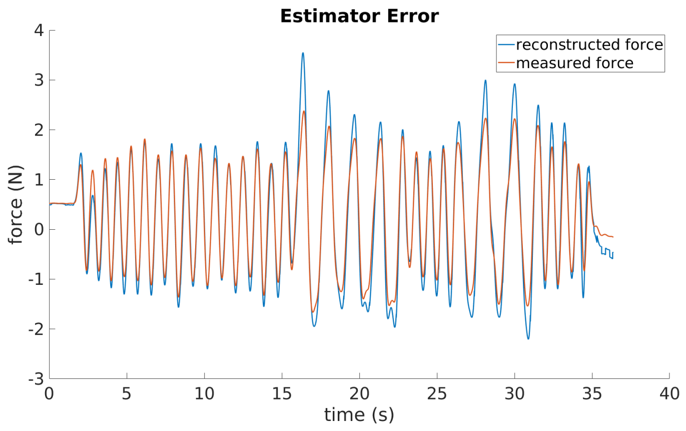

Figure 34: Measured and Reconstructed force

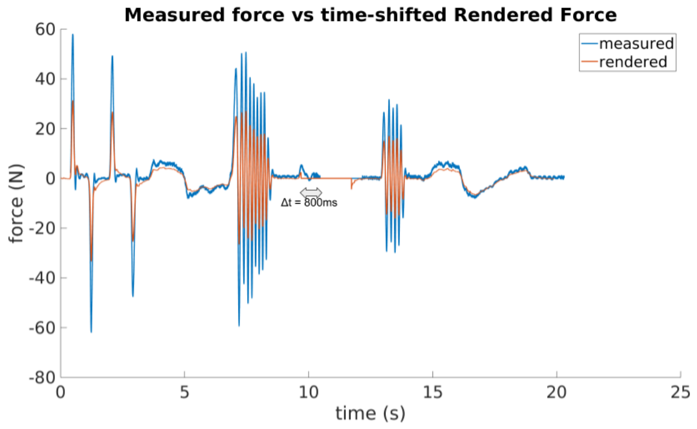

Figure 35: Measured and Rendered force for experiment 3

H Appendix: Conclusions and Recommendations

This section is a reflection on whether the experimental results give us sufficient information to appropriately answer the research question, and for the aspects where they do not, an overview is given of the work that could be undertaken to give a more conclusive answer.

H.1 Conclusions

To reiterate, the major claim made in this paper is that the proposed MMT system provides accurate force feedback for environments consisting of a single, flexible object on a flat surface.

While the operator is in contact with the object, the force feedback is sufficiently accurate at frequencies under 0.3Hz. Above this value, though, this is not the case. In this frequency range, the tracking error is on the order of magnitude of the signal. This is possibly due to the estimator under-estimating how much friction there is in the system. This assessment still stands for communication delays up to 400 ms.

When the operator makes or breaks contact with the object in the environment, the behaviour of the system is ideal under no time delay. However, under time delay, it takes a full round strip for an operator feeling a force response due to them carrying out an action that has them make or break contact with the object. This effect is unnoticeable for small time delays, but becomes very significant as time delays increase.

H.2 Recommendations

Here, we distinguish between further work required to clear up any open ends left by the experiments conducted, and further work that would involve improving upon the solution proposed in this paper.

H.2.1 Additional points of interest to explore

To solve the problem of the model switching being sensitive to time delay, a second contact detector could be added to the operator side of the system, using time-delayed point cloud information, to provide information about whether or not the operator is in contact with the object to the renderer. This would mean that time-delay would almost have no effect on the rendering system at all, provided the geometry of the object changes sufficiently slowly with respect to time that there is minimal difference between the currently measured point cloud and the lagged version thereof.

Finally, no experiments were conducted using measured point clouds. An experiment examining how well contact detection works on a real robotic setup using an actual RGBD camera would make the results obtained with the simulated environment somewhat more conclusive.

H.2.2 Further Work

Some improvements to the class of environments this system can handle could be made. Augmenting the system with the ability to handle more objects, and the ability for these objects to rest on surfaces with arbitrary geometry would make the system significantly more generally applicable. Another step towards more widespread applicability would be to give the system the ability to reproduce force feedback when an object is being carried through free space by the operator.

Additionally, because of how frequency dependent the accuracy of the force feedback is, a second method that is more reliable at high frequencies could be used in conjunction with the method proposed in this paper with a data fusion algorithm to ensure accurate feedback across the frequency spectrum. An impedance reflection algorithm modeling the environment as a variable spring could potentially fulfill this task.

Finally, when dealing with multiple objects, a means of dealing with the stability of the estimated parameters needs to be put in place. The unstable parameter put out at the start of the estimation process cannot be used, so the renderer can be turned off, another method of producing force feedback can be used, or an initial guess could be made regarding what the parameters should be, using visual information, for instance.

References

[1] Bullet Physics Code Repository. https://github.com/bulletphysics/bullet3. Accessed: 2020-09-10.

[2] Dartsim Code Repository. https://github.com/dartsim/dart. Accessed: 2020-09-10.

[3] Force dimension - products - omega.7 - overview. https://www.forcedimension.com/products/omega-7/overview. Accessed: 2020-07-21.

[4] Franka Control Interface (FCI) Documentation. https://frankaemika.github.io/docs/libfranka.html. Accessed: 2019-11-19.

[5] Franka Website. https://www.franka.de/. Accessed: 2020-07-21.

[6] FRI Wiki. https://cs.stanford.edu/people/tkr/fri/html/. Accessed: 2020-07-23.

[7] ODEint Code Repository. http://headmyshoulder.github.io/odeint-v2/. Accessed: 2020-10-12.

[8] Open Dynamics Engine Code Repository. https://bitbucket.org/odedevs/ode/src/master/. Accessed: 2020-09-10.

[9] Product Page Dragonskin 10. https://www.smooth-on.com/products/dragon-skin-10-slow/. Accessed: 2020-09-09.

[10] Raisim - Github. https://github.com/raisimTech/raisimlib. Accessed: 2020-10-12.

[11] ROS Wiki. http://wiki.ros.org/. Accessed: 2020-07-21.

[12] Simbody Code Repository. https://github.com/simbody/simbody. Accessed: 2020-09-10.

[13] M Franken, S Stramigioli, S Misra, C Secchi, and A Maccheli. Bilateral Telemanipulation WIth Time Delays: A Two-Layer Approach Combining Passivity and Transparency. *IEEE Transaction on Robotics*, 2011.

[14] P Goethals, G De Gersem, M Sette, and D Reynaerts. Accurate Haptic Teleoperation on Soft Tissues through Slave Friction Compensation by Impedance Reflection. *Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'07)*, 2007.

[15] B Hannaford. A design framework for teleoperators with kinesthetic feedback. 1989.

[16] J-Q Huang and F.L. Lewis. Neural-Network predictive control for nonlinear dynamic systems with time-delay. *IEEE Transactions on Neural Networks*, 2003.

[17] J Hwangbo, J Lee, and M Hutter. Per-contact iteration method for solving contact dynamics. *IEEE Robotics and Automation Letters*, 3(2):895–902, 2018.

[18] D Karnopp. Computer simulation of stick-slip friction in mechanical dynamic systems. *Journal of Dynamic Systems, Measurement and Control, Transactions ofthe ASME*, 1985.

[19] H Li and A Song. Virtual-Environment Modeling and Correction for Force-Reflecting Teleoperation With Time Delay. *IEEE Transactions on Industrial Electronics*, 2007.

[20] L Ljung. *System Identification: Theory for the User (Second Edition)*. 1999.

[21] L.J. Love and W.J. Book. Environment estimation for enhanced impedance control. *Proceedings IEEE International Conference on Robotics and Automation*, 1995.

[22] F Mobasser and K Hashtrudi-Zaad. Predictive Teleoperation using Laser Rangefinder. *Canadian IEEE Conference on Electrical and Computer Engineering*, 2006.

[23] G Niemeyer and J-J Slotine. Stable Adaptive Teleoperation. *1990 American Control Conference*, 1990.

[24] R Pillat and A Nagendran. Compliance Estimation during Bilateral Teleoperation of a Robotic Arm. *IEEE International Conference on Robotics and Biomimetics*, 2012.

[25] F Ryden and H Chizeck. A Proxy Method for Real-Time 3-DOF Haptic Rendering of Streaming Point Cloud Data. *IEEE Transactions on Haptics*, 2013.

[26] J-H Ryu, D-S Kwon, and B Hannaford. Stable Teleoperation with time-domain passivity control. *Proceedings 2002 IEEE International Conference on Robotics and Automation*, 2002.

[27] C Tzafestas, S Velanas, and G Fakiridis. Adaptive impedance control in haptic teleoperation to improve transparency under time-delay. 2008.

[28] C Weber, V Nitsch, U Unterhinninghofen, B Frber, and M Buss. Position and force augmentation in a telepresence system and their effects on perceived realism. *Proc. 3rd Joint EuroHaptics Conf. Symp. Haptic Interfaces Virtual Environ. Teleoper. Syst.*, 2009.

[29] X Xu, B Cizmeci, A Al-Muaimi, and E Steinbach. Point Cloud-Based Model-Mediated Teleoperation With Dynamic and Perception-Based Model Updating. *IEEE Transactions on Instrumentation and Measurement*, 2014.

[30] X Xu, B Cizmeci, C Schuwerk, and E Steinbach. Model-Mediated Teleoperation: Toward Stable and Transparent Teleoperation Systems. *IEEE Access*, 2016.