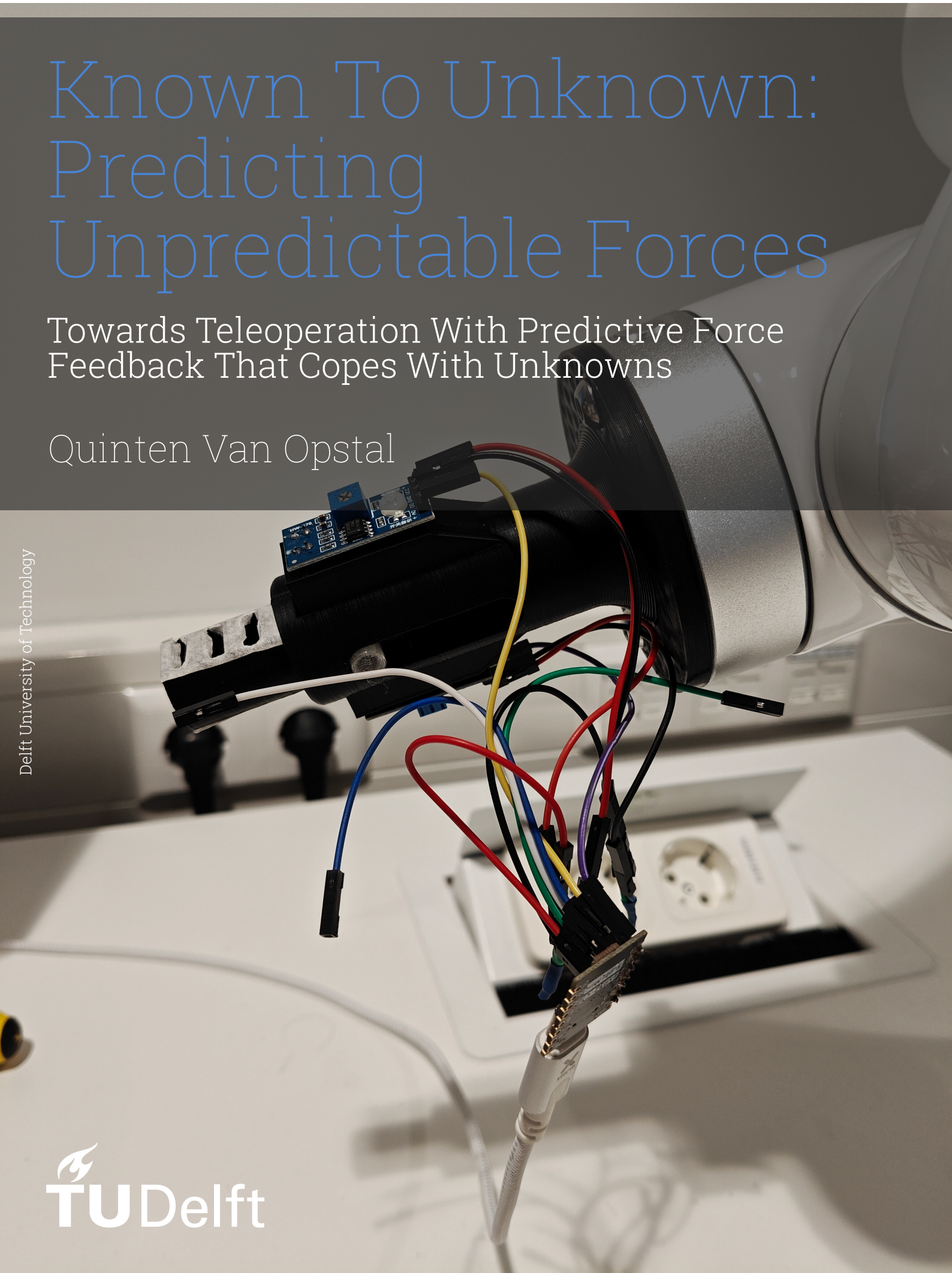
Known To Unknown: Predicting Unpredictable Forces

Towards Teleoperation With Predictive Force Feedback That Copes With Unknowns

Quinten Van Opstal



Delft University of Technology

TU Delft

Known To Unknown: Predicting Unpredictable Forces

Towards Teleoperation With Predictive Force Feedback That Copes With Unknowns

by

Quinten Van Opstal

to obtain the degree of Master of Science
at the Delft Universityof Technology,
to be defended publicly on the
Thursday February 5, 2026 at 1:30 PM.

Student number:	5316030	
Thesis committee	Dr. H.J.C. Kroep,	TU Delft, Daily Supervisor
	Dr. G. Lan,	TU Delft, Committee Member
	Dr. R.R. Venkatesha Prasad,	TU Delft, Thesis Advisor
Project Duration:	April, 2025 - January, 2026	
Research Group:	Networked Systems Group Faculty:	Faculty of Electrical Engineering, Mathematics and Computer Science, Delft

Style: TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

Abstract

Long distance Haptic Bilateral Teleoperation (HBT) is used in applications such as surgery, training, remote operation, and disaster relief. One of the main challenges in these systems is communication delay. When force feedback relies directly on sensors in the remote environment, increasing delay quickly makes the system unstable and hard to control. To overcome this, previous approaches generated force feedback using a local simulation of the remote environment, while providing visual feedback through live video [20]. This reduces timing constraints, but only works when the simulation accurately represents reality. When the robot encounters objects that are not modeled and not visible to the operator, unpredictable forces occur and the system no longer behaves correctly.

This thesis investigates how such unpredictable forces can be handled in delayed HBT. We introduce a method that keeps predictive force feedback through simulation, while correcting the simulation using sensor data from the robot. A small 3D printed attachment was developed to detect unexpected contact events. These measurements are sent back to the operator side and used to update the virtual environment, allowing force feedback to be generated even for unknown objects. The approach was evaluated in a user study with 13 participants under varying delays and visibility conditions. The results show that when visibility is limited, reaction based feedback can be used instead of prediction based feedback. The findings indicate that combining simulation based prediction with remote sensing offers a practical solution for dealing with unpredictable forces in long distance HBT.

Quinten Van Opstal
Delft, February 2026

Acknowledgments

I would like to thank dr. Kroep for being the daily supervisor of this master thesis, even while he got a son, and got his PhD. While this happened he still made time for our weekly meetings. His insight into the subject and knowledge of the technology greatly helped. I also want to thank dr. Ranga Rao Venkatesha Prasad better know as dr. VP as my thesis advisor, the fact that I could show up to his office unannounced or make a phone call to talk with him created a great environment where you could ask anything. I also want to thank dr. Guohao Lan for being part of the thesis committee. Next I want to thank my family for supporting me while doing the thesis, and proofreading it. Lastly, I want to thank the participant in the user study for their time, and the reader for reading this thesis.

Contents

Abbreviations

HBT Haptic Bilateral Teleoperation. i, iii, 1–4, 8–11, 13, 41, 43
HTS Haptic training simulators. 5

K2U Known To Unknown. 3, 6

m2f Motion to force. 10, 11
m2p Motion to pixel. 10, 11, 37
m2r Motion to robot. 9, 10, 37
MPC Model Predictive Control. 6

r2p Robot to pixel. 9, 10, 20, 21, 37

1

Introduction

Between 1999 and 2006, 307 US first responders lost their lives providing aid during and after natural disaster. The main causes of their deaths are exposure to fire, toxic gases, or accidents occurring during cleanup and reconstruction efforts [12]. This data highlights the need for technology that can reduce the risk to humans in dangerous situations like disaster response.

Imagine a coordinated network of specialists located anywhere in the world, able to provide aid to disaster sites without ever being physically present. It could improve the quality of the aid provided, while also reducing the response time. These experts could deliver faster, more efficient, and safer disaster relief, in case of a tornado, a flood, or even a wildfire. This could all be achieved from the safety of a remote control center. This eliminates the need for the experts to expose themselves to direct danger. One way to try and achieve this is to use robots on the disaster site. These robots could not only execute tasks human first responders would do, but they could also be capable of performing tasks beyond human physical limits, such as lifting debris, navigating hazardous terrain, or handling dangerous materials, but how do you control their actions?

Traditionally, robots are controlled by operators who remain in proximity to the machine. This is done to get accurate control and minimal communication delay. The accuracy and minimal communication delay is essential in providing effective and efficient relief. The operators must be able to quickly respond to changes in the environment, either caused by themselves, or something else. This defeats the purpose of having a centralized team, if they still need to be in the vicinity of the disaster site to operate the robots. Some recent developments in communication networks and control systems have made it possible to operate robots across distances, this is known as teleoperation.

Teleoperation allows the operator to control the robot, it doesn't allow them to feel the environment. For the operator to effectively interact with the remote environment, they should be able to sense what the robot encounters. This allows them to provide more accurate instructions based on the physical objects around the robot, and how the robot interacts with them. To do this, the robot must have sensors to sense its environment, and send the data back to the operator, so that they can react. Sending this data requires very fast communication between the robot and the operator to ensure the operating experience remains realistic. One of the most established frameworks for achieving effective teleoperation is Haptic Bilateral Teleoperation (HBT). In a HBT system, the operator controls a remote robot, through an interface that mirrors the robot's movements and provides tactile or force feedback. This bidirectional exchange of motion and force data enables the operator to "feel" interactions between the robot and its environment in real time, this enhances the situational awareness and control accuracy of the operator. Achieving this sense of presence requires two feedback channels: real time visual feedback from the robot's environment and haptic feedback that replicates the tactile sensations experienced by the robot.

Transmitting haptic data over long distances introduces latency. Even small delays like a couple of milliseconds in feedback can have detrimental impacts on the quality of the interaction and reduce the operator's ability to feel the remote environment. You can try to mitigate this obstacle by employing

predictive models that estimate the expected force feedback on the operator's side. This allows the system to maintain responsiveness and, even when visual feedback experiences minor delays [20].

Predictive feedback and video streaming can provide sufficient feedback for the operator to successfully execute a task under normal conditions, it becomes more difficult when the robot encounters an unexpected object or forces that are applied to the robot that are not included in the predictive model. The operator does not immediately perceive the robot's contact with the unmodeled obstacles, thus disrupting the illusion of presence and degrading control accuracy. These unpredictable forces are prevalent in unstructured or disaster stricken environments, where debris, unstable materials, and irregular terrain frequently can produce unforeseen interactions.

1.1. Technical issues

The main problem with long distance HBT is the communication delay. When the delay becomes too large, the operator no longer feels in sync with the robot, which makes precise control difficult. This is a big problem for generating force feedback, since even small delays can make the system confusing to use. One solution is to simulate the remote environment on the operator's side and generate force feedback based on the simulation. This virtually removes the delay for haptic feedback and makes the system feel more responsive. However, this only works when the simulation correctly represents the real world.

In real environments, the robot can encounter objects that are not part of the simulation, or they are in the wrong place. Whenever this happens, unpredictable forces occur. The operator does not feel that they encountered the unexpected object. This breaks the interaction and can reduce safety. To mitigate this, a sensor on the robot can be used to send feedback to the operator to update the simulation. One issue with this is that updating the simulation based on sensor data introduces extra delay, which can cause the virtual and real environments to be misaligned. This creates position offsets that are noticeable to the user. Visual feedback also adds delay, and achieving low enough latency for live video is difficult with standard hardware. All these issues make it difficult to provide real-time force feedback and visual feedback.

1.2. Current solutions

Most current solutions for long-distance HBT focus on handling delay. One group of approaches changes the control signals, which helps to keep the system stable. For example, using passivity-based control or adding damping [2, 16]. This approach can reduce instability that occurs with large delays. However, it often makes the system feel slower and less natural to the operator.

Another approach that is widely used is to rely on a predictive model of the remote environment. In this setup, the environment is simulated locally on the operator side, and force feedback is generated based on this simulation instead of sensor data from the robot [28, 35]. Because everything happens locally, the operator almost instantly experiences force feedback, independent of the network delay. Some approaches also use the simulation to predict future robot motion and interactions, in order to further improve responsiveness [28].

This predictive force feedback is usually combined with visual feedback, either by showing live video of the remote environment or by rendering a simulated view of the robot and its workspace. Previous research has shown that with prediction, systems can still feel usable even when τ_{m2p} reaches around 150 ms [20]. However, for these approaches to work, they need an accurate simulation. When the robot encounters something that is not in the simulation, the force feedback the operator experiences is different from the one the robot experiences, which can lead to unsafe situations or break the immersion.

Some work has tried to correct the simulation using sensor data from the robot [19]. While this helps to bring the virtual environment closer to the remote environment, it also introduces extra delay. This can often cause the virtual contact point to appear at the wrong location at first, creating position mismatches between the real and simulated environments. These mismatches are especially noticeable during contact with objects and can disturb the operator.

Overall, current solutions work best in structured environments where all objects are known beforehand. In more realistic environments with unpredictable forces or unknown obstacles, these methods start to

break down.

This thesis investigates how HBT systems can handle unpredictable forces while maintaining a responsive user experience. Specifically, this thesis aims to answer the following research question: ”How can a long distance Haptic Bilateral Teleoperation operator handle unpredictable forces?”. We answer this question by answering these 3 sub questions:

- ”What are the effects of having no direct line of sight on the area of operation?”
- ”Can reaction based force feedback be used when prediction is not possible?”
- ”How does the movement speed of the operator effect the performance of a HBT system”

1.3. Our approach

In this thesis, we introduces Known To Unknown (K2U), which combines predictive force feedback with sensing in the remote environment to handle unpredictable forces in long-distance HBT. A local simulation on the operator’s side generates fast force feedback, while sensors on the robot detect unexpected contact with objects. As long as the simulation matches the real world, the operator receives immediate and natural force feedback. When the robot encounters an object that is not in the simulation, the sensors detect the contact and send this information back to the operator’s side. The simulation is then updated by adding a virtual object so that force feedback can be generated based on the virtual object.

Because this update arrives later than the real collision, a mismatch appears between the real and virtual contact location. To reduce this effect, an adaptive offset is used that slowly aligns the virtual contact with the real one while the operator moves away. This keeps the interaction smooth and usable. This approach aims to keep the system responsive while still allowing the operator to handle unpredictable forces.

To evaluate the approach, a user study was conducted where participants controlled the robot under different added delays, visibility conditions, and simulation accuracy. Participants rated their experience and confidence in using the system. The results were used to analyze how well the approach performs when unpredictable forces occur and how delay influences the user experience.

The main contributions of this thesis are:

- A teleoperation setup that combines local simulation-based predictive force feedback with sensing in the remote environment to handle unpredictable forces.
- A custom 3D-printed sensor attachment for a robotic arm. It can detect unexpected contact with objects in the remote environment, and it sends this information back to the operator’s side.
- An adaptive offset method that reduces the mismatch between real and virtual contact locations caused by communication delay.
- A way for an operator to apply force to an object after they made contact with it, using haptic feedback.
- A user study evaluating the system under different delays, visibility conditions, and simulation approaches to understand how these factors influence the user experience.

The remainder of this thesis is as follows: chapter 2 reviews related work relevant to feeling the environment and haptic bilateral teleoperation. Chapter 3 introduces the systems used in this research. Chapter 4 introduces the theoretical background required to understand the concepts used in this research. Chapter 5 describes what we did to get an experimental setup, followed by chapter 6, which shows the implementation of the developed system, and how we evaluate it. Chapter 7 contains the results of the user study and discusses them, and it has a discussion section. Chapter 8 provides an overall conclusion of this thesis, and it has a future work section that outlines potential improvements and directions for future work.

2

Related work

In this section, we discuss previous work related to long-distance Haptic Bilateral Teleoperation (HBT) and how researchers have dealt with delay in teleoperation systems. First, we look at traditional approaches that focus on stability and low-latency communication. Then, we go over solutions that use simulations and predictive force feedback to reduce the effect of delay. Finally, we discuss work that tries to update simulations using sensor data and the problems that arise when the real world does not match the virtual environment. This gives an overview of the current state of the field and shows where the limitations of existing solutions are.

2.1. Haptic Bilateral Teleoperation (HBT)

HBT allows an operator to control a remote robot while receiving real time force and motion feedback, this creates a sense of physical presence in a remote environment [14, 31]. The first telemanipulators used in the mid-20th century have come a long way, with advances in sensing, actuation, and control. Early research focused on achieving stability and transparency. Even when communication delays are present, trying to ensure that force feedback remains accurate and safe despite delays or in communication channels [9, 22, 24].

Recent research looks into immersive and intelligent teleoperation. These systems often use sensors, lightweight actuators, and wearable interfaces to improve tactile feedback, thus improving user comfort and precision [32]. Digital Twin driven Mixed Reality (DTMR) system, combines real time modeling with haptic rendering to improve immersion and the ability of an operator to execute a task [11].

One challenging application of HBT is humanoid robot teleoperation due to high degrees of freedom, stability, responsiveness, and corresponding force feedback becomes more complex[8]. Most current research aims to merge robust control with immersive human–robot interaction, pushing HBT closer to achieving immersive and reliable spatial awareness.

2.2. "No delay" teleoperation

Currently, one of the biggest challenges in teleoperation is minimizing latency. Even small communication delays of a couple of ms can significantly reduce task performance, stability, and user experience. Studies have shown that delays greater than 150 ms can significantly reduce precision, and raise operator workload [34]. In medical and industrial applications where precision and safety are essential, achieving near real time or “no delay” teleoperation is vital. Research has explored multiple latency mitigation strategies, including predictive control, and adaptive feedback. Collectively, they aim to preserve the operator’s perception of immediate responsiveness [34].

Recent developments in communication systems have made sub 10 ms latency possible under optimal network conditions. For example, Black et al. [4] achieved end-to-end latencies of approximately 1 ms over Ethernet and 40 ms over 5G. They used a WebRTC based mixed reality teleoperation framework. The results indicated that human response times are now the limiting factor. Typically, the human

Figure 2.1: Example of a ”no delay” teleoperation robot, the Da Vinci surgery robot picture from cghmc

response time is between 150 and 500 ms, which is larger than the network delay.

Low latency teleoperation has been successfully used in robotic surgery, here real time haptic feedback is directly tied to performance and safety. True force feedback can reduce applied intra corporeal forces by nearly half compared to visual only feedback [15, 23]. Systems using bilateral haptic control and direct force reflection show improved task precision and reduced tissue damage [6, 26]. An example of ”No delay” teleoperation can be seen in figure 2.1.

2.3. High Latency Teleoperation and Full Simulation

Increased delays in the communications network can make it harder to control things in real time. A complete virtual simulation can be used to better engage operators and make sure they can quickly receive force feedback. When the delays are too large, executing a task with a machine from a distance can become unreliable. The signals sent and received will not match up properly, so the operator can’t rely on force or visual feedback. Simulations of the remote environment can be used to allow an operator to receive force or visual feedback with a much smaller delay, eliminating most of the problems introduced by a large delay. This allows the operator to better execute a task, and can make sure the robot doesn’t execute any unwanted action based on a delay in feedback. Building simulations, with accurate physical modeling and digital twin representations, can provide realistic training and control rehearsal. Even if the environments doesn’t allow real time feedback [11]. The simulation enables users to maintain spatial awareness, thus making them able to perform complex tasks under several seconds of communication delay.

Haptic simulation improves spatial awareness by including tactile and kinesthetic feedback into virtual training systems. Haptic training simulators (HTS) replicate physical interactions and provides force feedback that improve user’s experience with contact and material resistance [21]. Surgical Simulation systems, with haptic feedback enabled, significantly improve user skill acquisition compared to purely visual or joystick, based simulations [8]. This is especially relevant in the contexts of surgical training or hazardous teleoperation, because real world training would be unsafe or impossible. High computational resources are often required to achieve such a simulation, and thus remain limited in accessibility due to hardware costs and calibration complexity [4].

Integrating AI driven assessment and predictive modeling can enhance full simulation teleoperation. Teleoperation based learning frameworks allow the robots to learn tasks from human demonstration within virtual environments, this allows for simulation to come closer real world performance [21]. Simulation is a crucial tool for sustaining surgical training during periods of restricted physical access [13]. These works demonstrate that by combining virtual environments, haptic feedback, and full simulation teleoperation allow for an effective alternative for training under high latency or unreliable communica-

tion conditions.

2.4. Predictive force feedback

Latency is an issue in teleoperation, it occurs when an operator controls a robot over large distances. Delayed force feedback can disrupt the spatial awareness of the operator. Predictive force feedback addresses this. It estimates the forces the operator should feel before actual sensor data is available. In telesurgical applications, predictive force feedback frameworks have been shown to improve user experience in haptic feedback even under less than ideal network conditions. A predictive model that uses 5G for communication, meant for needle insertion, demonstrated that an estimation of resistance can produce more reliable haptic cues, mitigating the negative effects of communication delay [5].

The robot's behavior during small time frames can be predicted with Model Predictive Control (MPC). The operator then sends modified actions to compensate for delays and uncertainties. When used with online force estimation, a controller can maintain stability, while the operator receives accurate predictive force feedback [17, 18]. Predictive force feedback has been used in collaborative haptic interaction, where predicting contact forces improves coordination between human and robot partners. This method reduces the cognitive load of the operator by aligning the predicted force feedback with the operator's intended movement [33]. There also exist machine learning based approaches to force prediction. They use Deep learning models trained with audio–tactile signals to generate highly accurate estimations of force feedback, this can be used when direct force sensing is impractical [25].

Machine-learning-based prediction has been used to be able to improve the realism of force feedback in mixed-reality environments. Here, estimating future interactions with the environment allows haptic devices to generate force feedback [29]. In human–robot collaborative object transport, predictive estimation of force feedback has demonstrated to improve task accuracy, and user emersion [10]. Together, these contributions show that predictive force feedback is a key piece in the next-generation of teleoperation systems. They enable stable, and intuitive interactions with the remote environment despite network latency or incomplete sensory information.

	latency	haptic feedback	visual feedback
Teleoperation			
Batty [3]	low	simulation	live video
Boabang [5]	low	simulation	live video
Chioson [6]	low	sensor	live video
Fan [11]	medium	sensor	simulation
Gonzalez [15]	medium	simulation	simulation
Patel [26]	low	sensor	live video
salvato [29]	medium	simulation	simulation
Training			
Feizi [13] Lelevé [21] Rangarajan [27] Al-Saud [30]	n/a	simulation	simulation
Prediction			
Dominguez [10]	n/a	sensor-based	n/a
Jordana [17]	n/a	prediction	n/a
Kleff [18]	n/a	prediction	n/a
Miederhauser [25]	n/a	sensor-based	n/a
Watanabe [33]	n/a	simulation	n/a
Our paper			
Known To Unknown (K2U)	medium	simulation & sensor	live video

Table 2.1: Comparison between the contents of different related works.

2.5. Comparison

Various other works were discussed throughout this chapter, a systematic comparison can be seen in table 2.1. We devided the literature into three categories. The first category is teleoperation, it includes papers that look into remote robot control and the challenge of maintaining accurate force feedback with larger delays. The second category is training, includes works that employ simulation or haptic rendering to allow for training, with the main objective of improving user performance in procedural tasks. And the final category is prediction, contains studies whose main contribution lies in the development of predictive models for force feedback prediction.

Several characteristics were used to create a comparison between the related works. These characteristics are all present in a subset of the related work. The first characteristic is in what latency each tested system operates, there are 2 options: low latency meaning that the operator is really close, and moderate latency when there is a significant delay between the operator preforming actions, and the robot executing them. Another important characteristic is how haptic force feedback is generated. The forces rendered have 3 possible sources, they are rendered through physical sensing, via simulation based models, or using machine learning based predictions.

For works that utilize a visual feedback, we additionally classify the type of visual feedback used, such as live video streaming, or purely simulated feedback. Using the characteristics, we compared the reviewed literature approaches to mitigating the problem of latency. Their approaches are to do it by visual or force simulation, have the operator really close, or don't use a remote robot.

3

The system

This chapter explains the overall system that was built for this thesis. First, we give a high level overview of Haptic Bilateral Teleoperation (HBT) and how the different parts of the system work together. Then, we describe the operator side, the remote side, and the communication between them. We also explain how the simulation is used to generate predictive force feedback and how sensor data from the robots side can be used to handle unexpected contacts.

3.1. Haptic Bilateral Teleoperation (HBT)

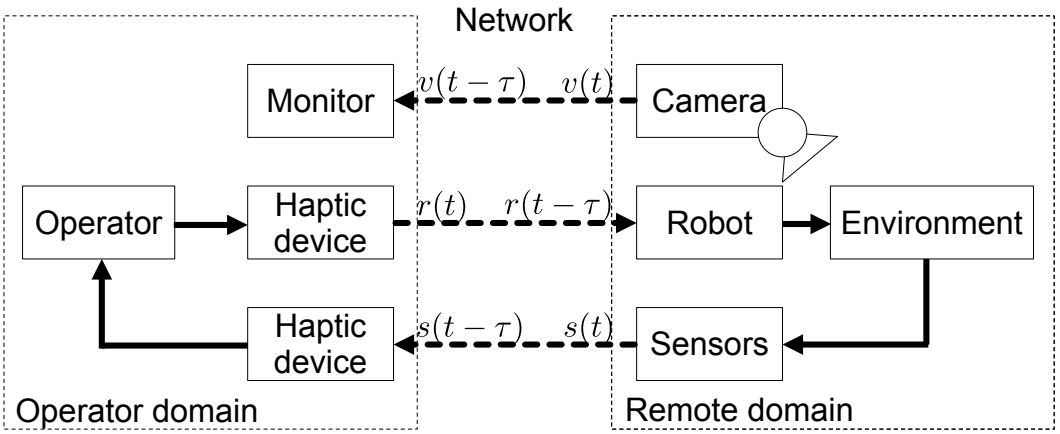


Figure 3.1: Inner workings of HBT. The operator performs actions with the haptic device, it sends the actions over the network to the robot, who executes them. This affects the environment, which is monitored by the sensors. They send their data back to the haptic device over the network. It lets the operator experience force feedback. Changes in the environment also get monitored by a camera. Who sends a video stream to a monitor over the network, The operator can observe these changes on the monitor.

The main goal of *Teleoperation* is to enable an operator to execute complex tasks in a remote environment by controlling a robot. The operator doesn't need to be present in the remote domain. In *Haptic teleoperation*, the operator controls the robot through motion and applied forces, which are measured by a haptic device. *Haptic Bilateral Teleoperation (HBT)* extends this, by transmitting not only the operator's actions to the robot but also the robot sends visual and force feedback back to the operator. This two-way communication allows the operator to feel the remote environment in real time, creating

spatial awareness despite the distance between the robot and the operator. A basic HBT system can be understood as consisting of three components: the operator domain, the remote domain, and the communications network. As illustrated in figure 3.1, the operator and robot reside in different physical spaces, the operator domain, and the remote domain. The network serves as the medium through which information flows between the two other components.

The *operator domain* consists of the human operator and a haptic device with which they can interact. The operator manipulates the haptic device, which records its position and forces applied to it. These are translated into control commands for the robot. These commands are transmitted over the network to the remote domain. Meanwhile, the haptic device generates force feedback based on data from sensors in the remote environment. The operator experiences these forces, so they can feel when the robot has a collision. The operator also receives visual feedback in the form of a live view of the robot's surroundings. The combination of both visual and force feedback allows the operator to build a spatial awareness and thus act as though they were physically present with the robot.

The *communications network* sends data between the operator and the robot and the other way around. Its specific implementation, whether wireless, wired, local, or long-range, is not important to the working of HBT. What matters is that information sent from one domain to the other actually reaches it. The information that is sent, can be control commands, force feedback signals, and a live video stream. All actions performed by the operator must be sent to the robot, and every measurable force exerted on the robot in the remote domain must be sent back to the operator. The quality of this communication plays a central role in the user's experience: delays, packet loss, or jitter directly influence the user's experience and spatial awareness of the controller when controlling a remote robot.

The *remote domain* contains the robot, a camera, and force sensors. The camera and sensors are used to gather both force and visual feedback. The robot receives and executes commands sent from the operator domain, and the consequences of these actions are sensed by sensors. These sensors measure properties such as contact forces, motion constraints, and movement in the environment. These measurements are transmitted back to the operator domain to be rendered as force feedback. A camera system records the robot and its surroundings, providing a live video stream used by the operator to monitor the remote environment. The actions of the operator are executed in the remote domain by the robot, the effect of these actions are measured and recorded by sensors and cameras. This measured data gets sent back to the operator in the form of force feedback, or a live video stream.

3.1.1. Timing restrictions

A primary challenge in HBT is that communication occurs over a network. When data is sent long distances, it introduces significant delays. This can significantly degrade the user experience of the operator, when the delay gets to large the operator can't effectively control the robot [1]. Consider an operator who wants the robot to place an object on a table. When an operator does this in their own domain, the operator would feel resistance the moment the object makes contact with the table, and the downwards motion stops. They also get visual confirmation that the object is now on the table. HBT uses the same principle, when the robot makes contact with an object, it is detected, and the haptic device receives this information and restricts the operator's movement accordingly, allowing the operator to feel the contact with the object.

This works well over short distances with negligible latency. However, when a network is introduced, it inherently adds a delay, we denote this delay by τ_{network}. As illustrated in figure 3.1, the robot at time t executes commands that were sent at time $t - \tau_{\text{network}}$. The monitor displays images recorded at $t - \tau_{\text{network}}$, and the same applies to the force feedback derived from the sensor measurements.

Thus, when the operator performs an action on the haptic device, the total delay before the robot moves is denoted by *Motion to robot (m2r)*. We can represent it with,

$$\tau_{\text{m2r}} = \tau_{\text{network}} + \tau_{\text{haptic}} + \tau_{\text{robot}},$$

where τ_{haptic} is the delay between preforming an action on the haptic device and sending it to the remote domain, τ_{robot} is the processing and actuation delay of the robot. The delay measured from the robot moving, to it being visible on the monitor is called *Robot to pixel (r2p)*, and can be represented by

$$\tau_{\text{r2p}} = \tau_{\text{network}} + \tau_{\text{camera}} + \tau_{\text{monitor}},$$

where τ_{camera} is the camera capture and transmission delay, and τ_{monitor} is the monitor's display delay. Combining m2r and r2p gives us *Motion to pixel (m2p)*, where

$$\tau_{\text{m2p}} = 2\tau_{\text{network}} + \tau_{\text{haptic}} + \tau_{\text{robot}} + \tau_{\text{camera}} + \tau_{\text{monitor}},$$

this is the delay between the operator preforming an action, and it being visible on the monitor. Lastly, we call *Motion to force (m2f)* the time required before the force feedback corresponding to an action is rendered.

$$\tau_{\text{m2f}} = 2\tau_{\text{network}} + \tau_{\text{haptic}} + \tau_{\text{robot}} + \tau_{\text{sensor}} + \tau_{\text{force}},$$

where τ_{sensor} is the sensor measurements delay and force is the delay for receiving sensor data and generating force feedback. For the operator to control the robot accurately and perform complex tasks, the total delay must be very small, otherwise the system becomes difficult to control or unstable [2]. In practice, this requires τ to be in the order of 10 ms. No widely deployed commercial network can currently meet this requirement [16].

3.2. Live footage, and predictive force feedback

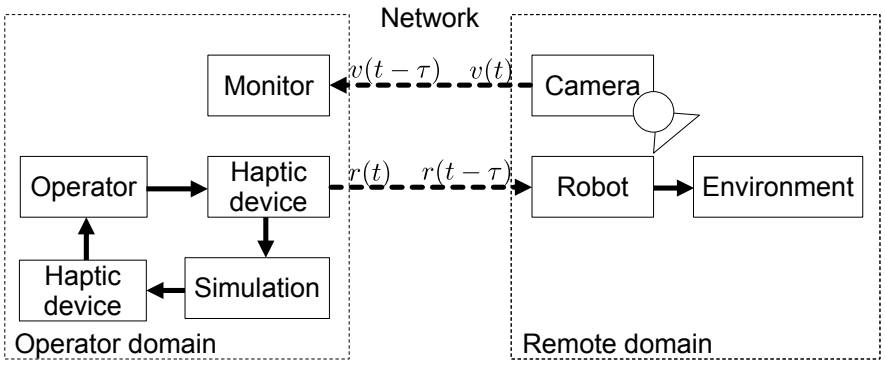


Figure 3.2: Inner workings of HBT with a simulation. The operator performs actions with the haptic device, it sends the actions over the network to the robot, who executes them. The remote environment is simulated on the operator side, it also gets the actions. This simulation generates force feedback, that the operator can experience. Changes in the actual remote environment get monitored by a camera. Who sends a video stream to a monitor over the network, the operator can observe these changes on the monitor.

To reduce the harsh timing requirements imposed on the communication network, the remote environment can be simulated locally on the operator side. This substantially decreases the time required for the operator to receive haptic feedback. The simulated environment must reproduce all relevant elements with which the remote robot interacts in order to generate meaningful haptic responses [28]. This architecture is illustrated in figure 3.2. In this setup, sensors in the remote domain are no longer required for force feedback generation, as all force responses originate from a local simulation. The operator controls a haptic device. It sends the coordinates of the device to the simulation, which contains a virtual model of the remote workspace. Within the simulation, a proxy object moves according to the movement of the haptic device, based on its position in the simulation, force feedback is computed.

To generate natural and convincing haptic feedback, the system must represent force magnitude, force direction, and contact position [7]. Most physics engines only track object positions and detect collisions; they generally do not natively support haptic rendering. However, they can be extended through spring-based interaction models to compute force feedback [35]. In this approach, the proxy object acts as a point constrained by the virtual environment. When it collides with an object, its motion is restricted to the collision point. The proxy can move away from the surface or along its tangent plane, but cannot penetrate the object. This constraint prevents the objects from further colliding, while still allowing free tangential motion.

The operator's physical motion via the haptic device is not directly constrained; instead, a force feedback signal is produced. The haptic force at time t is computed as

$$f_{\text{feedback}}(t) = k_{\text{object}}\big(p_{\text{proxy}}(t) - p_{\text{haptic}}(t)\big),$$

where $p_{\text{proxy}}(t)$ is the proxy's position, $p_{\text{haptic}}(t)$ is the haptic device position, and k_{object} is the spring constant representing the stiffness of the contacted surface. This formulation ensures that the deeper the haptic device moves into the virtual object, the larger the resulting feedback force experienced by the operator. Because all computations occur in the operator domain, this process eliminates both network and remote sensor delays. The resulting m2f can now be expressed as

$$\tau_{\text{m2f}} = \tau_{\text{haptic}} + \tau_{\text{simulation}} + \tau_{\text{force}},$$

where τ_{force} denotes the time required for the simulation to process an input, and compute the corresponding output force. Due to the nearly instant simulated force feedback, the operator can rely primarily on haptic feedback, while the visual feedback may tolerate higher latency. Prior work shows that m2p can be significantly larger, up to approximately 150 ms without severely impairing task performance[20].

3.3. Unpredictable forces

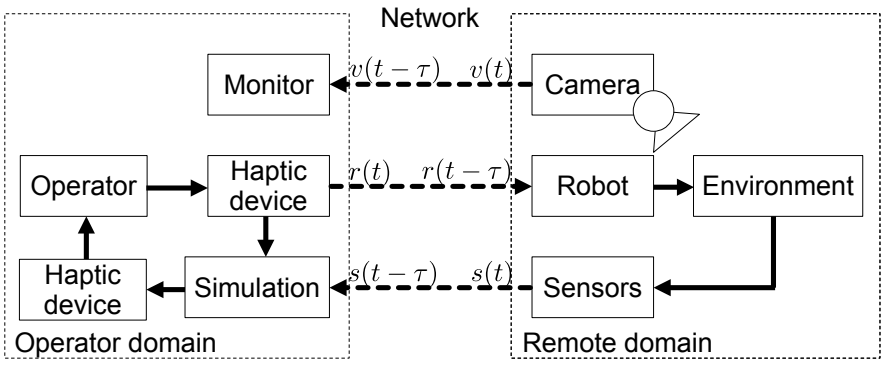


Figure 3.3: Inner workings of HBT with a simulation. The operator performs actions with the haptic device, it sends the actions over the network to the robot, who executes them. The remote environment is simulated on the operator side, it also gets the actions. This simulation generates force feedback, that the operator can experience. The remote environment contains some sensors to detect unexpected object, when it encounters them the data gets sent to the simulation so that it can be updated. Visual changes in the remote environment get monitored by a camera. Who sends a video stream to a monitor over the network, the operator can observe these changes on the monitor.

The simulation-based approach assumes that the virtual environment contains a virtual version of all objects with which the robot may interact. However, this assumption fails when the robot encounters an object that is not visible to the operator and not represented in the simulation. For example, when searching for loose change under a couch, a person knows that the loose change is present but not the precise location of the loose change. Similarly, the robot may make contact with an unseen object whose position is unknown to both the operator due to no direct line of sight, and the simulation model. When the robot moves into an unexpected object, the robot will experience a force from this object. The operator receives no corresponding haptic feedback, because the simulation doesn't have a virtual representation of the object. We refer to such contact as unpredictable forces: forces resulting from contacts that neither the operator nor the simulation anticipates. To address this, this thesis investigates an approach in which the remote environment is still simulated locally, but the robot is equipped with sensors capable of detecting unexpected contacts. When these sensors detect such an object, they

immediately halt robot motion along the direction of contact to prevent damage. At the same time, the sensor data are transmitted to the simulation, which updates the virtual environment and generates the appropriate force feedback. The updated architecture is illustrated in figure 3.3.

The time required for the robot to stop after encountering an unexpected obstacle is given by

$$\tau_{\text{stop}} = \tau_{\text{robot}} + \tau_{\text{sensor}}$$

τ_{m2f} remains the same as when the force feedback is only based on the simulation. When the sensors detect the robot encountering an unexpected object, the information is sent to the simulation. The simulation creates a new virtual object corresponding to the unexpected object, and generates force feedback based on it. Thus, the time from the operator moving, the robot encountering something, and the operator experiencing this is:

$$\tau_{\text{reaction}} = 2\tau_{\text{network}} + \tau_{\text{haptic}} + \tau_{\text{robot}} + \tau_{\text{sensor}} + \tau_{\text{simulation}} + \tau_{\text{force}}.$$

This delay is longer than when only using the simulation, because the system must wait for: two-way network transmission, physical robot interaction dynamics, sensor detection and processing, and simulation updates using the data sent by the robot when it encountered the unexpected object. These additional steps introduce delays that cannot be avoided when handling unpredictable forces. The approach ensures that safety is preserved, and the simulation is dynamically corrected to reflect the actual environment, enabling the operator to receive force feedback even when the environment is only partially known. The delay means that the simulation will receive the update later than the actual collision happening, this can be represented by

$$\tau_{\text{collision}} = \tau_{\text{network}} + \tau_{\text{sensor}} + \tau_{\text{simulation}},$$

With this delay, the simulation will place the object at another location if the haptic device kept moving. We hypothesize that this wrong representation is still sufficient, because the operator doesn't know the exact location of the object.

4

Theoretical basis

This chapter explains the theoretical background needed to understand the rest of this thesis. First, we explain how our system handles unpredictable forces with an adaptive offset. Then we look at how we can apply additional force after encountering an unexpected object.

4.1. Adaptive offset

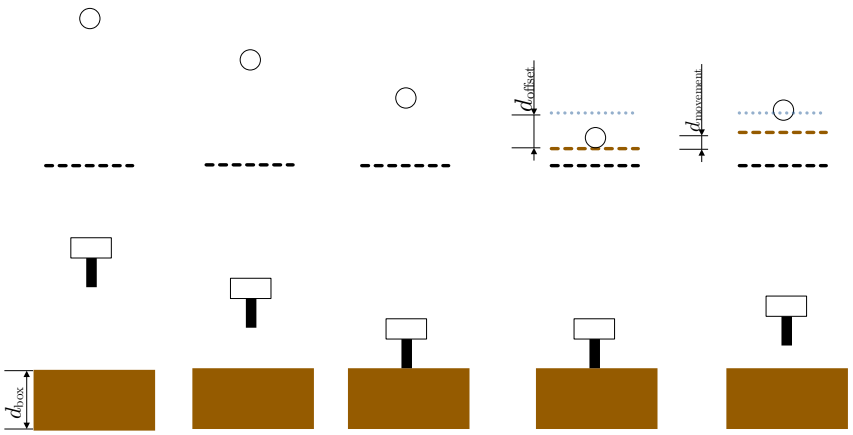


Figure 4.1: Illustration of how the operator and robot interact when the robot encounters an unexpected object. When the sensors on the robot detect the object, they sent a signal to the operator. Due to network delay, there is an offset. This offset is slowly reduced.

When there is a delay in Haptic Bilateral Teleoperation (HBT), the robot inevitably lags behind the operator's motion, as illustrated in figure 4.1. It shows how the operator and robot interact when the robot encounters an unexpected object. The black doted (—) line represents a plane in the simulation, the operator can't move past this plane. Force feedback is generated based on this plane. First, the proxy object in the simulation and the robot are at the same height, when the operator moves down the robot follows with some delay. When the robot encounters an object, it sends its current location to the operator domain, and stops moving. Meanwhile, the operator is still going lower because they haven't received the that the robot encountered an object yet. When the simulation receives that the robot has encountered a box, it moves the plane to the current position of the haptic device, denoted by the brown doted (—) line. This represents the contact plane with a virtual object, the operator can't move past this plane. The simulation also keeps track of the real location of the object indicated by the blue doted (—) line, based on the position of the robot when it encountered the box. Every time the operator moves away from the contact plane, the virtual contact plane also moves a little until the

virtual contact plane is at the same location as the real contact plane.

The robot's position at time t is based on the operator's position at an earlier time:

$$p_{\text{robot}}(t) = p_{\text{operator}}(t - \tau_{\text{m2r}}).$$

where $p_{\text{operator}}(t)$ is the position of the operator at time t. The delay τ_{m2r} creates an offset between the operator's current position and the robot's current position. Consider a scenario in which the operator attempts to perform a task but doesn't have a direct line of sight of where the task is performed. While they are executing the task, they encounter an object that they cannot see. The operator keeps on moving, because they are unaware of the obstruction, meanwhile the robot immediately stops when it encountered the object:

$$p_{z,\text{robot}}(t) = d_{\text{box}},$$

where $p_{z,\text{robot}}(t)$ is the height of the robot, and d_{box} is the height of the unknown item. The network delay causes the operator to not receive this contact until τ_{reaction} later. By the time the signal arrives, the operator and, by extension, its proxy in the simulation have already moved beyond the point where the robot actually made contact. This creates an offset between the real and virtual environments. This corresponds to what Kroep et al [19] called delay-induced position error, which can cause significant disturbance when predicted contact forces are involved, these disturbances persist. To reduce and eventually eliminate this offset, the simulation must determine how far the proxy has moved beyond the true contact point. When the virtual simulation creates a box of its current position when it receives that the real robot encountered an object, the positions of the virtual box, and the real one are misaligned. The spatial offset is therefore:

$$d_{\text{offset}}(t) = |p_{\text{proxy}}(t) - p_{\text{proxy}}(t - \tau_{\text{m2r}} + \tau_{\text{haptic}})|,$$

The offset is based on both the network delay and the speed at which the operator moves the haptic device, which is consistent with Kroep et al. [19], they show that delay-induced errors scale with interaction velocity and become strongly perceivable during contact events. Another approximation that relies on fewer external measurements, such as the network delay and the velocity of the haptic device, uses the robot's recorded contact position:

$$d_{\text{offset}}(t) = |p_{\text{proxy}}(t) + \text{convert}((p_{robot}(t - \tau_{\text{network}}))|,$$

where convert$(\cdot)$ maps robot coordinates into the simulation coordinates. The network delay is present due to the nature of the communication that travels over the network. To preserve the ability of the operator to control the robot, the offset is added to the position sent to the robot:

$$p_{\text{to robot}}(t) = p_{\text{proxy}}(t) + d_{\text{offset}}(t),$$

where $p_{\text{to robot}}$ is the location sent to the robot. This ensures that when the operator moves in the opposite direction of the contact plane, the movement is still transmitted correctly, and this makes the real robot move away from the object.

The offset is reduced over time to make sure the virtual environment eventually aligns with the real environment. This is like the principle behind the Adaptive Offset Framework (AOF) proposed by Kroep et al. [19], which compensates for delay in errors with position by adjusting an offset while keeping the haptic perception smooth and stable. When the operator moves away from the virtual object, the virtual object itself can also be moved in that direction until it aligns with the real world. We define a directional projection:

$$\text{direction}(p, q) = \left(\frac{p \cdot q}{||q||^2}\right) \cdot q,$$

which extracts the movement of vector p along vector q. Using this, we compute how much the virtual box has to move.

$$d_{\text{movement}}(t) = \frac{\text{direction}\big(|(p_{proxy}(t) - p_{\text{proxy}}(t-1)|, d_{\text{offset}}\big)}{2},$$

This shifts the virtual box by half of the operator's movement in the offset direction. The offset is then updated as:

$$d_{\text{offset}}(t) = d_{\text{offset}}(t-1) + d_{\text{movement}}(t).$$

This gradual update behaves analogously to the decay functions in AOF.

4.2. Adaptive offset with applied force

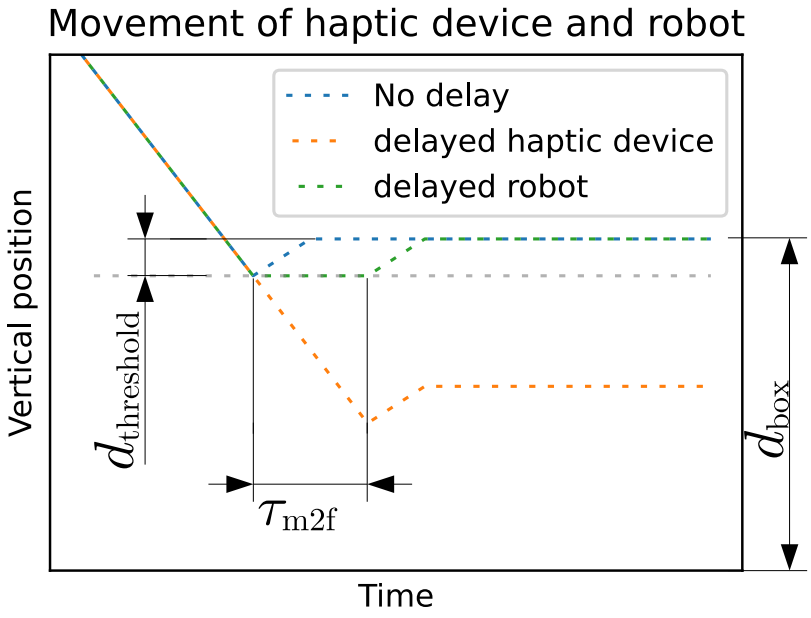


Figure 4.2: Illustration on the effect of delay on the vertical position of the robot and haptic device in relation to each other. In this figure we only consider τ_{m2f}, and none of the other delay.

To enable the operator to deliberately apply force to an encountered object, it is essential that the robot first comes to a controlled stop upon contact. Once this contact is detected and communicated back to the operator, the operator should be able to modulate the force exerted on the object in a predictable and intuitive manner. In practice, this means that after the initial stop, further interaction with the object should be governed by the force applied by the operator to the virtual representation of the environment, rather than by unrestricted motion commands.

A common way to apply force with a robot is to command it to move slightly into an object. When the robot's end effector is modeled as being connected to the environment through a virtual spring, the magnitude of the applied force becomes proportional to the depth. By moving the robot further into the object, a larger force is exerted, while moving it away reduces the applied force. This behavior provides a natural and physically intuitive mechanism for force control.

The force that the operator experiences through the haptic device, denoted by $f_{\text{feedback}}(t)$, can therefore be used directly to determine how much additional displacement should be commanded to the robot in order to apply the desired force. This displacement is given by

$$p_{\text{force}}(t) = \frac{f_{\text{feedback}}(t)}{k_{\text{spring}}},$$

where k_{spring} represents the stiffness of the virtual spring attached to the robot's end effector. This term converts the desired force into a corresponding positional offset. By incorporating this displacement into the command sent to the robot, the total target position becomes

$$p_{\text{to robot}}(t) = p_{\text{proxy}}(t) + d_{\text{offset}}(t) + p_{\text{force}}(t).$$

With this formulation, the operator is able to actively apply force to the object by increasing the force they exert on the virtual object, and the robot responds by moving further into the physical object. Conversely, reducing the applied force causes the robot to retreat, decreasing the interaction force accordingly. In this way, the applied force at the robot is directly proportional to the force feedback perceived by the operator.

In the remote domain, object contact is detected once the robot's sensors register that a predefined distance threshold has been exceeded. At this point, the robot halts further motion in the direction of

contact to prevent damage. Because the robot is modeled as interacting through a virtual spring, this stopping behavior results in an applied force on the object equal to

$$f_{\text{object}} = k_{\text{spring}} * d_{\text{threshold}}.$$

where $d_{\text{threshold}}$ is the distance at which contact is detected. This force represents the baseline force exerted on the object when the robot initially comes to rest.

To ensure that the operator is aware that the robot is applying force to the object, the operator must also experience a corresponding force feedback. This is achieved by slightly shifting the virtual object away from the robot's contact position in the direction opposite to the collision. Doing so introduces a virtual collision that produces force feedback on the haptic device, allowing the operator to feel the interaction. Incorporating this effect modifies the offset term as

$$d_{\text{offset}}(t) = \left| p_{\text{proxy}}(t) + \text{convert}(p_{robot}(t - \tau_{\text{network}})) + \frac{f_{\text{object}}(t)}{k_{\text{object}}} \right|.$$

This adjustment ensures that the operator experiences the same force that the robot applies to the object, despite the presence of communication delay. A simplified representation of the effect of τ_{m2f} on this interaction is illustrated in figure 4.2. Here we only look at vertical movement, and only consider τ_{m2f}, but the same principle applies for all the axis, and when the other delays are also considered. When there is no delay, the operator and robot both follow the dotted blue line (- -). Since there is no delay, the robot perfectly mimics the movements of the operator. First the operator moves down, the robot follows this motion until it encounters an object. The robot sends this to the operator. Now both the robot and the operator are at $d_{\text{box}} - d_{\text{threshold}}$, the operator experiences force feedback corresponding to $d_{\text{threshold}}$. They move up until they no longer feel the feedback, now both the robot and operator are at d_{box}.

How τ_{m2f} effects the movement of the haptic device is shown by the doted orange line (- -). First, the operator still moves down, but they don't immediately receive that the robot encountered an object. So they keep moving down for τ_{m2f}. When the message arrives, the operator receives force feedback corresponding to $d_{\text{threshold}}$, and moves back up. The operator thinks that they are now at the location of the box, but they are lower, this corresponds to their movement speed between the robot encountering an object, and the operator receiving this. The doted green line - -) shows the corresponding movement of the robot. Here the robot goes downward, like when there is no delay, until they encounter an object. They send this information to the operator, and meanwhile they stop moving. When the operator receives this info and moves up, the robot also moves up. This means that the robot remained in place for τ_{m2f} at $d_{\text{box}} - d_{\text{threshold}}$. While the operator is lower than the actual box, the robot is precisely at the location of the box.

A similar effect can be observed when the operator chooses to actively apply force to the object, as illustrated in figure 4.3. We again use the simplified view, but everything also applies when there is more delay and more axis of movement. Here, the blue doted line (- -) again represents both the operator and the robot when there is no delay. The operator and robot both simultaneous move downwards until the robot encounters an object at $d_{\text{box}} - d_{\text{threshold}}$ when the operator feels this force feedback, they decide to move down further to exert more force. The robot mimics this motion, after a while the operator moves back up again until they no longer feel force feedback, now both the robot and operator are at d_{box}. The doted orange line (- -) shows what happens to the vertical position of the operator when τ_{m2f} is present. The operator again moves downward, until they receive from the robot that the robot has encountered an object. This is τ_{m2f} later than when there is no delay. After receiving the corresponding force feedback, the operator decides to move further downwards, after a while they move back up again. The operator thinks that they are on top of the box, but they are not. They are a bit lower, this corresponds to their movement speed between the robot encountering an object, and the operator receiving this. The doted green line (- -) shows how the robot moves when τ_{m2f} is present. They also move downwards with the operator, until they encounter the object. They hang at this position $d_{\text{box}} - d_{\text{threshold}}$ for τ_{m2f}. When the operator receives that the robot encountered an object and moves downward, the robot also moves downwards, but they move as if the haptic device was at the same position as the robot, so the offset is mitigated. When the haptic device moves back up, the robot follows, until the robot reaches d_{box}.

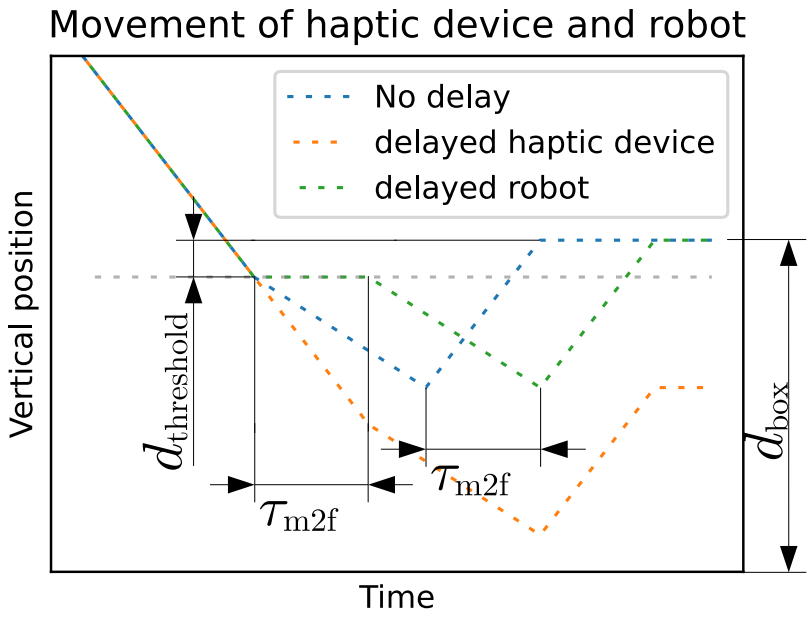


Figure 4.3: Illustration on the effect of delay on the vertical position of the robot and haptic device in relation to each other. In this figure we only consider τ_{m2f}, and none of the other delay.

5

Implementation

In this chapter, we explain how the system was built. First, we examine how communication between the operator side and the robot side was established using Tailscale. This was done to facilitate easier connectivity between the different parts of the system, even when they are on different networks. After that, we describe our attempts to implement live video feedback from the remote environment, and the problems we encountered in reducing the delay. Lastly, we discuss the physical prototypes that were made to detect unexpected contact, and how the design evolved over multiple iterations.

5.1. Communication

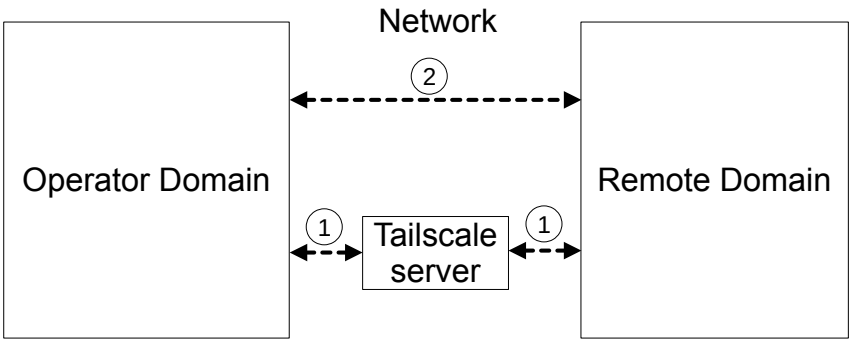


Figure 5.1: Explanation of our use of tailscale, (1) both the operator and the remote domain connect to the tailscale server. They are now under the same tailnet, (2) this allows them to communicate with each other without the tailscale serve

Firstly, we tried to improve the setup of the project, by making it easier for the different components to communicate with each other. This to reduce some of the hassle during the setup of the teleportation system. The operator domain and the remote domain need to constantly communicate with each other, they send coordinates, sensor readings, and video streams. Establishing direct and stable communication channels between geographically distributed machines is not trivial. Many real-world networks impose restrictions such as dynamic IP addressing, NAT traversal, or firewalls that block inbound connections. This makes the deployment of the system more difficult, since we want the system to be able to work with the operator located in any part of the world, and the robot in any part of the

world. The main thing we want to eliminate are the unpredictable IP addresses and opening ports for communication for incoming connections. To do this we use Tailscale, a mesh VPN technology, as a component of our network layer. Tailscale provides all device connected to it with a unique and stable virtual IP address, independent of the underlying physical network. All components that must communicate whether running on separate physical machines or co-located on a single workstation join the same Tailscale network. Once connected, they appear to exist on a unified private subnet, enabling seamless communication even across restrictive or heterogeneous network environments. Tailscale can be configured to allow communication between certain ports of devices connected to it.

When a device connects to the Tailscale coordination server, it is assigned a Tailscale IP address that other devices connected to the server can use to communicate with the device. After this initial discovery, Tailscale tries to establish a direct connection between the participating devices so that they don't need the server. As illustrated in figure 5.1. This makes sure that the coordination server is used only for setup, and state updates. The server is not used to relay application data under normal circumstances. This means that the system does not incur additional latency from the VPN layer, which is important for our applications, because in teleoperation we want minimal communication delay between the components to ensure the quality of the interaction and the confidence of the end user. Tailscale also has a MagicDNS feature, it assigns a human-readable hostname to every connected service. This means that these names can be used instead of IP addresses, further reducing the technical knowledge required to setup the project. Not relying on fixed IP addresses has other benefits, since the IP addresses can change during development. This eliminates the need for manual reconfiguration when components are moved or replaced.

We wanted the architecture to be flexibile, all services should be able to run on multiple distributed machines, all concurrently on a single machine, or a mix of these 2. To achieve this, each service is deployed as one or multiple Docker containers. This containerization ensures different hardware is not really a factor in the deployment. And it makes sure that each service behaves consistently regardless of its host machine. To use these Docker containers in the Tailscale network, we introduce an additional container that communicates with the Tailscale server for each device we want to connect to the network. The other containers on the device route their traffic through this node. This allows the containers to join the Tailscale network without needing a full Tailscale installation, and they don't need to be modified, so they can still run independent. Running all services on a single machine introduces additional challenges with respect to networking. Docker containers on the same host can fail to communicate correctly if placed on conflicting or overlapping network interfaces. We therefore configure each container to use a different network interface, ensuring clear separation and that traffic routing functions correctly in both local and distributed deployments. This allows us to test the setup on a single machine while preserving compatibility with the remote deployment scenario.

To further refine the testing environment, we deploy a Headscale server. That is the self-hosted variant of the Tailscale server. It took longer than expected to run the headscale server and all the other containers on the same machine. Headscale enables us to manage connection keys, and routing policies without depending on the external infrastructure. This is great for research, where reproducibility and independence from proprietary cloud services are desirable. By connecting both local and remote containers to the same Headscale server, we ensure that minimal configuration changes are required when transitioning from multi-machine setups to single-machine simulations or to a mix of the two. Headscale does not enable UDP communication by default, but our teleoperation setup requires UDP for low-latency data exchange. This functionality needs to be explicitly enabled. For this to work, all connections to the Headscale server itself need to be authenticated over HTTPS. With the containerization, limited setup, and the ability to self-host, we created a setup that is flexible, repeatable, and easily testable even on a single device. This setup allows the system to function reliably across diverse network conditions while providing developers with a controlled and reproducible environment for testing and experimentation.

5.2. Live low latency video

After simplifying the setup needed for communication between the operator domain and the remote domain, we decided to focus to enabling live video feedback from the remote environment to the operator. The live video is necessary to get visual feedback from the remote environment. This visual feedback

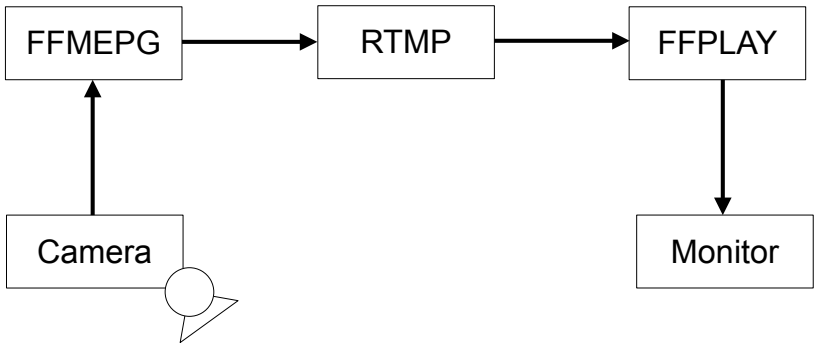


Figure 5.2: First attempt of low latency video streaming, here a FFMEPG container sends the video signal from the camera to a RTMP server running in a different container, a container running FFPLAY streams this data to the monitor

is essential for teleoperation. We tried to create a setup that allows the operator to receive live video with minimal Robot to pixel (r2p) delay.

The first implementation used a three-container setup. One container ran FFMPEG, a command-line tool used to capture and transmit video from the camera. The captured stream was sent to an RTMP server running in a second container. The third container ran FFPLAY, which subscribed to the RTMP stream and displayed the video on the operator's monitor. We chose to use docker containers for the same reasons as stated in the previous section, and because it allowed for easy integration with the tailscale setup. An overview of this architecture is shown in figure 5.2.

The FFMPEG container was geared to for ultra–low-latency video transmission. The pipeline used the bare minimum of processing and buffering, to minimize delay. To further reduce delay, all unnecessary encoding steps were avoided. The resolution of the transmitted video was deliberately kept low this to reduce the number of packets required per frame. This in then reduced serialization and transmission delays. These choices aimed to reduce the time between an event occurring in the remote environment, the camera capturing that event, and the resulting video frame being transmitted to the RTMP server.

The RTMP server acted as a middle man between the video producer and the consumer. Its role was to gather the incoming video stream from the FFMPEG container, and distribute it to one or more FFPLAY containers. The RTMP server allowed the video capture and video display components to start independently. It also enabled multiple FFPLAY containers to subscribe to the same live stream simultaneously.

On the operator side, FFPLAY was geared to minimize the delay in displaying the stream. Internal buffers were disabled to prevent additional inherent delay. Despite all the added optimizations, the τ_{m2p} remained substantial. The results of the latency measurements for this setup are shown and analyzed in section 7.1.

This setup failed to achieve a mean τ_{m2p} below 150 ms, causing us to explore an alternative approach. For the purposes of executing the user study, it was feasible to capture video and display it on a single computer. This allowed us to remove both the FFMPEG container and the RTMP server from the pipeline, further reducing delay. FFPLAY was used directly to capture video from the camera and display it on the monitor By eliminating intermediate components, this configuration reduced overhead, and thus reduces τ_{r2p}.

The resulting delay was still higher than desired, even with the simplified approach. Measurements

showed a Robot to pixel (r2p) delay τ_{r2p} of about 80 ms, which is too large. The system can thus not be considered usable for low-latency teleportation. The remaining delay can be attributed to several unavoidable factors. The camera operated at 30 fps, it also performed internal image processing using two to three frames before outputting a usable image. The refresh rate of the display also added a delay before a captured frame became visible on the monitor.

Due to these limitations and the time constraints on the project, it was not possible to further reduce the τ_{m2p} to a usable level. Because of this, the user study was conducted without live video feedback. Instead, participants were able to directly observe the robot with their own eyes. This decision ensured that the study focused on haptic interaction and control behavior without introducing confounding effects from excessive visual delay.

5.3. algorithm

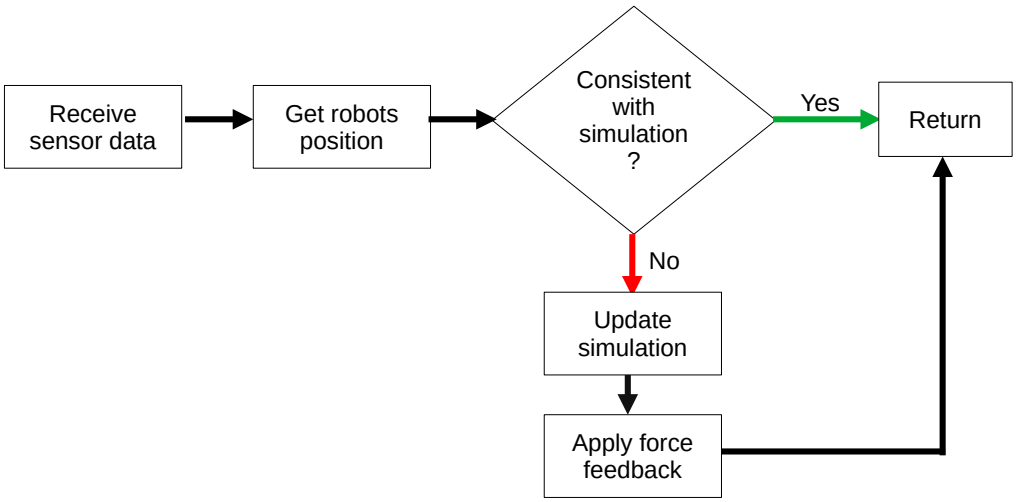


Figure 5.3: Flowchart that illustrates how our algorithm in the operator domain handles unexpected forces from the remote domain.

This section describes the design of the algorithms used in our system. Before these changes, communication between the operator domain and the remote domain was already in place, and the robotic arm followed the movement of the haptic device. When the operator moved into an object and received force feedback, this movement into the object was not sent to the robot. Because we want the robot to also be able to apply force to an object, we modified the control logic. The movement of the haptic device into an object is now transmitted to the remote domain as well, scaled by a spring constant.

How the simulation in the operator domain is updated when sensor data from the remote domain is received is shown in figure 5.3. The algorithm starts when the operator domain receives data from a sensor located in the remote domain. Using this sensor data together with information from the robot, the contact point is determined. The algorithm then checks whether this contact is already possible within the current simulation. If the contact is possible, the algorithm returns without making changes. If the contact is not possible, the simulation is updated.

The simulation update is done by inserting a virtual object. This object is first placed at the current position of the haptic device and then moved in the opposite direction of the detected force, over a distance equal to the detection threshold. This ensures that the force feedback calculated by the simulation corresponds to the force experienced by the robot. While updating the simulation, the algorithm also computes the offset between the newly created virtual contact point and the actual contact location of the robot. This offset is stored and later used by a separate algorithm. After the simulation update, force feedback is calculated and applied, and the function returns.

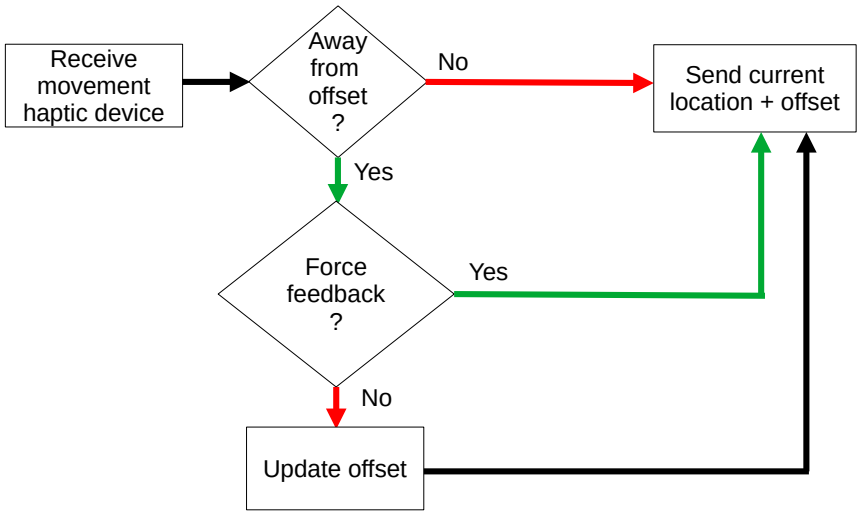


Figure 5.4: Flowchart that illustrates how our algorithm implements an adaptive offset

The algorithm that handles this offset is shown in figure 5.4. This algorithm is executed every time the haptic device registers movement. First, it checks whether the movement is in the direction of the offset. If the operator does not move away from the offset, the current position of the haptic device plus the offset is sent to the remote domain. This position already includes the force feedback component discussed earlier.

If the operator does move away from the offset, the algorithm checks whether force feedback is still present. This prevents the offset from being updated while the operator is still in contact with an object. If force feedback is present, the current haptic position plus the offset is again sent to the remote domain. If there is no force feedback, the offset is reduced by subtracting half of the operator's movement away from the offset from the offset. The updated haptic position plus the new offset is then sent to the remote domain. This allows the robot to continue moving while an offset exists between the operator domain and the remote domain.

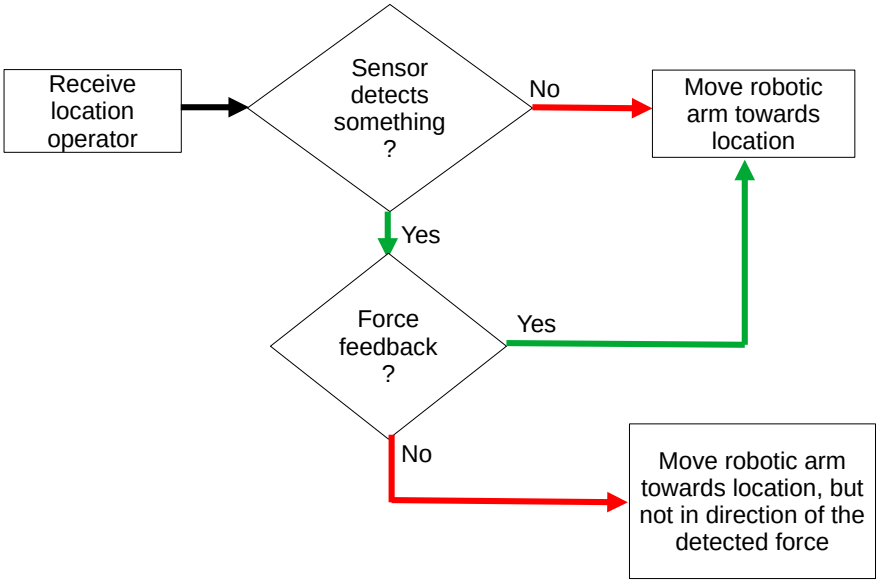


Figure 5.5: Flowchart that illustrates how the robot handles its sensors encountering unexpected forces

Finally, in the remote domain, there is an algorithm that ensures unexpected objects are not damaged. This algorithm runs every time the remote domain receives a new target position from the operator domain. First, it checks whether any contact is detected by the sensors. If no contact is detected, the robot moves toward the target position as normal. If contact is detected, the algorithm checks whether the operator already receives force feedback. This is used to determine whether the operator is intentionally applying force or has not yet noticed the contact.

If force feedback is present, the robot is allowed to move toward the target position. If no force feedback is present, the robot still moves toward the target position, but motion in the direction of the detected force is blocked. This allows the operator to move away from the object while preventing further penetration.

5.4. Physical prototypes

Next, we designed an attachment that can be mounted on a robot arm. The goal of the attachment is to detect when the robot arm encounters an object without breaking the arm or the attachment. We used a hollow cylindrical structure, which can be attached to the robotic arm. Inside the cylinder is another cylinder that can move up and down. Inside the outer cylinder, there is a spring, which ensures that the location of the inner cylinder remains the same when no forces are applied to it. The inner cylinder's movement is limited to one degree of freedom, relative to the outer shell.

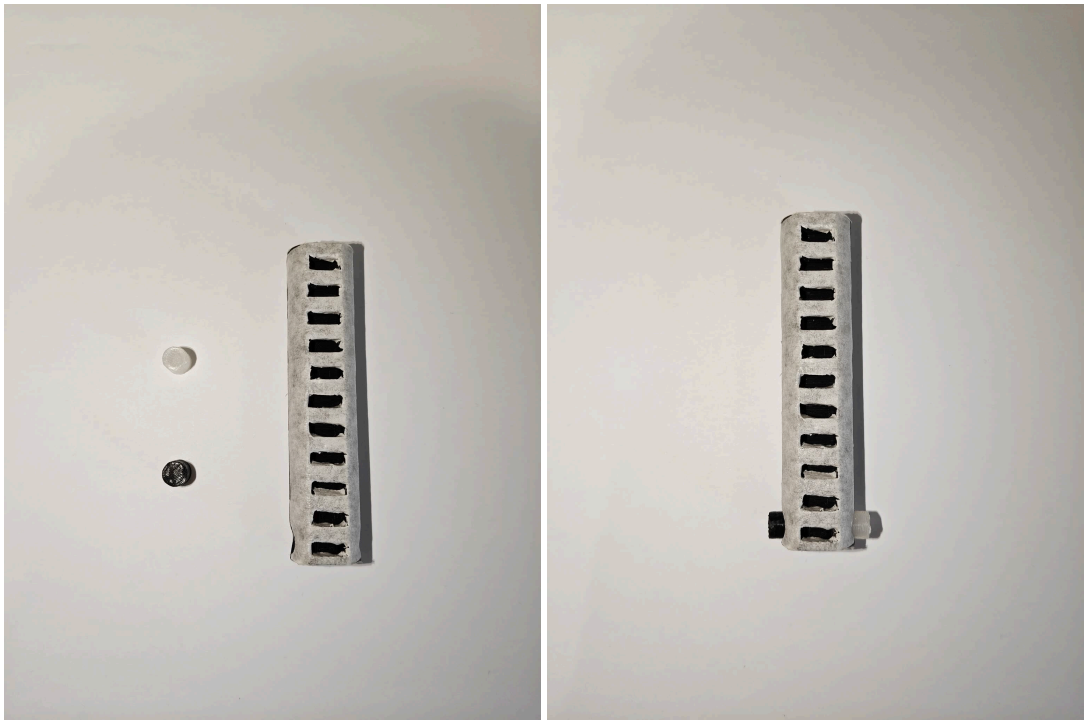
The inner cylinder compresses the spring when the robotic arm encounters an obstacle. When the robotic arm moves away, the spring ensures that the inner cylinder returns to its original position. The displacement is measured, which allows us to deduce where the encountered object is. The outer cylinder serves as the housing, while the inner cylinder is a moving element. We made multiple versions of a prototype, and they all used 3D printing. This allowed for rapid iterations at a low cost. Using 3D printing also allowed us to easily customize the designs in a CAD software.

5.4.1. Version 1: initial

(a) (b)

Figure 5.6: Attachment Version 1: (a) front view and (b) side view.

The goal of the first design was to make ourselves more familiar with computer-aided design (CAD) software. We wanted to create something that could easily be mounted onto the robotic arm. And it



(a) (b)

Figure 5.7: Mechanism to connect the inner cylinder. All the separate parts (a), and the parts assembled (b)

should have some moving parts, such that there is a component as part of the attachment that can move freely in only one degree of freedom without breaking the attachment.

The first prototype satisfied these requirements, the attachment could be attached to the robot arm. Its geometry allowed it to be produced using 3D printing techniques. The attachment has an inner cylinder that can freely move around vertically, without it breaking or falling out. There is a way to incorporate a spring into the design. This demonstrated to us that the idea of the attachment is possible, and the design can be used as a base to develop further iterations. However, there is no way to actually measure if the arm encountered an object, so the design is still limited.

However, the prototype also exhibited several limitations. Most notably, it lacked a mechanism for measuring the forces acting on the moving cylinder, which was essential for the intended functionality. Furthermore, the overall design was relatively long, making it less compact and potentially less practical for integration with different robotic configurations. To address this issue, the next iteration should have some kind of sensing mechanism that can detect when the robot encounters an object.

The initial prototype design is shown in figure 5.6. The front view can be found in figure 5.6a and the side view in figure 5.6b. There are 2 vertical slots along the side of the attachment, as can be seen in figure 5.6b. This allows the inner cylinder to attach to the outer cylinder with some pins, which can be seen in figure 5.7a. The figure also shows where the pins attach to the inner cylinder. The slots allow for vertical movement, but limit all other kinds of movement. Everything together can be seen in figure 5.7b. We use this attachment mechanism in all further iterations.

5.4.2. Version 2: linear encoder

The second design iteration focused on actually measuring when the inner cylinder moves. Several sensors were considered, including linear potentiometers, optical encoders, and magnetic encoders. After comparing the different options, we selected an optical encoder as the most suitable solution. It is a cost-effective alternative to magnetic encoders and, unlike a linear potentiometer, it is not limited by a small measurement range. The front view of the design is shown in figure 5.8a, and the side view in figure 5.8b.

(a) (b)

Figure 5.8: Attachment Version 2: (a) front view and (b) side view.

This iteration was also designed to be more compact, while still leaving enough space to integrate the encoder into the attachment. However, there were still some problems. First, the transparent plastic used in the 3D-printed prototype caused issues with the optical encoder, making the light-based measurements unreliable. Additionally, the setup did not have a way to detect the direction the inner cylinder was moving. This meant that we could detect a change, but not if we where close to an object, or moving away from it.

With these issues in mind, we decided on two main improvements for the next iterations. First, the attachment should be fabricated using opaque material to avoid interference with the optical measurements. Second, a direction detection mechanism should be added, so that both the displacement and the direction of motion can be measured reliably.

5.4.3. Version 3: 2 encoders

For the third iteration, we decided to focus on detecting the direction in which the inner cylinder moves. For this, we used two optical encoders and the A-B phase principle. Here, the idea is that both sensors measure the movement of the inner cylinder based on engraved slots. However, these slots are placed at an offset of a quarter of the slot size. With this offset, the encoders measure the same movement at different intervals, based on which encoder measured the change first, we can determine the direction of the movement, either upwards or downwards.

To obtain better encoder readings, we decided to add slits in front of the encoders. We measured multiple sizes of slits by moving the encoder over the inner cylinder in a dark environment, with the encoder looking through the slit being tested. The results of this can be seen in figure 5.10. The moving speed was not consistent between measurements, which is why the measurements don't seem to be in phase. We looked for the smallest slit size that had an acceptable difference between the measured highs and lows. The smallest slit size allowed for the smallest resolution, and we decided to use slits with a size of 1 mm. The final configuration is shown in figure 5.9b.

With this change, the attachment could measure the direction of the movement, and how for the inner cylinder moved. There are still some improvements that can be made. This 3D structure used the

(a) (b)

Figure 5.9: Attachment Version 3: (a) front view and (b) side view.

same amount of unnecessary filament, reducing this would reduce the print time of the attachment. In this design, the encoders were mounted at an offset, this meant that it could only be used with one size of slits. Because we wanted more possibilities, we decide to make inner cylinders slots have an offset instead. The front and back views of the inner cylinder can be seen in figure 5.11a and figure 5.11b, respectively.

5.4.4. Version 4: more efficient

The fourth design iteration focused on using less material during 3D printing, and also making the encoder placement more flexible. In the previous design, the encoders were physically offset, which meant that the cylinder slot spacing was basically fixed. In this iteration, both encoders are placed at the same height. This removes that constraint, so the attachment can now work with cylinders that have different indent sizes. That makes the design more modular, and it is easier to change the measurement resolution by just swapping the cylinder. The front of this design is shown in figure 5.12a, and the back in figure 5.12b.

To still make the two-encoder setup work, the indents on the back side of the cylinder were designed to be staggered by one quarter of the slot pitch. This way, we still get the required phase shift between the two encoder signals, and we can keep reliable direction detection. The CAD model of the staggered cylinder with 4 mm gaps is shown in figure 5.13a and figure 5.13b. Another improvement is that the encoders can now be mounted on the sides of the attachment using screws, which makes it easier to adjust or replace them.

To increase the contrast between a gap and no gap, painter's tape was placed on the cylinders, as shown in figure 5.14. We also made cylinders with different step sizes. The version with 4 mm gaps is shown in figure 5.14a, the 3 mm version in figure 5.14b, and the 2 mm version in figure 5.14c.

Overall, this iteration reduced unnecessary filament use and made sure different kinds of inner cylinders could be used. One limitation is that the prototype is still larger than it needs to be compared to the size of the springs inside. A more compact version would print faster and use less filament.

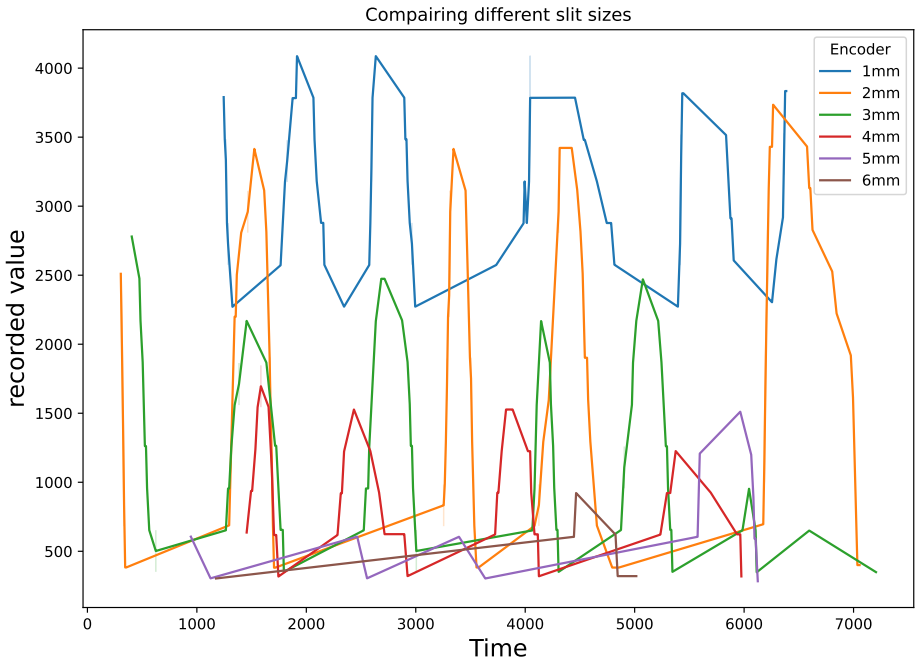
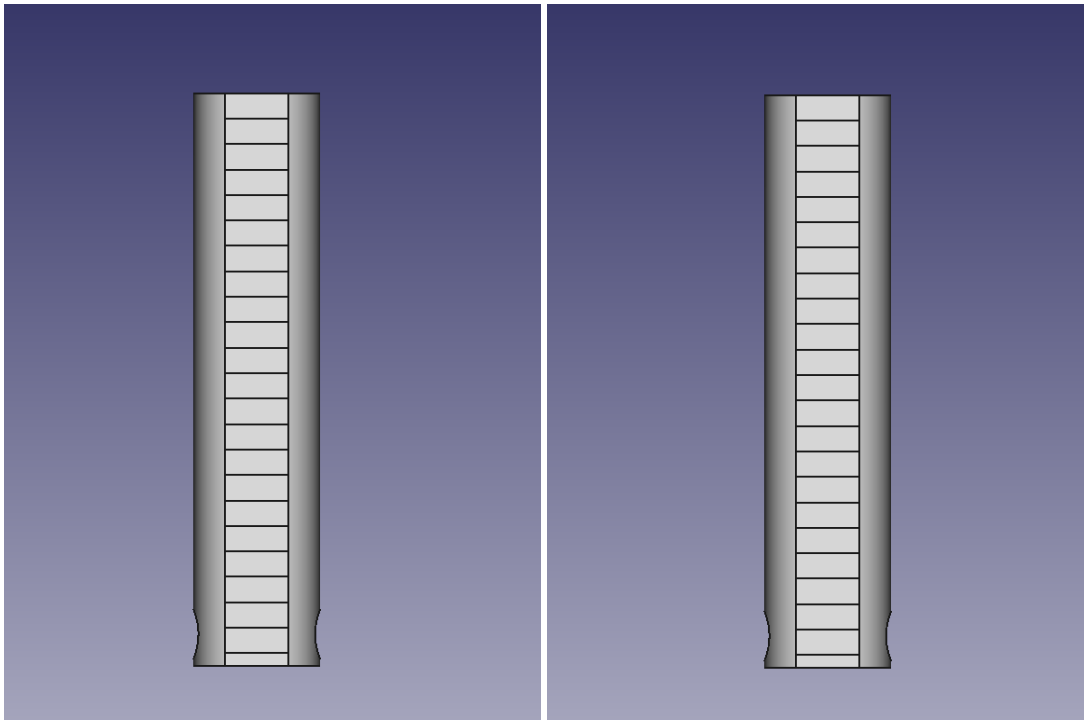


Figure 5.10: Comparing the effectiveness of different kinds of slits, by running them over a cylinder with 4 mm gaps, and painters tape for extra contrast. The ticks are an indication on the amount of time past on the micro controller, and not how fast a difference is measured.

5.4.5. Version 5: smaller

The fifth design iteration focused on making the attachment more compact, to better match the size of the internal spring. This was done to reduce print time and to ensure that the spring fits better. The updated attachment is shorter than the previous version and includes a recess for the spring, in both the attachment housing and the inner cylinder. This recess allows for better alignment of the spring and reduced unwanted movement. The front view can be found in figure 5.15a, the side view in figure 5.15b. The rest of the cylinder is the same as the previous design.

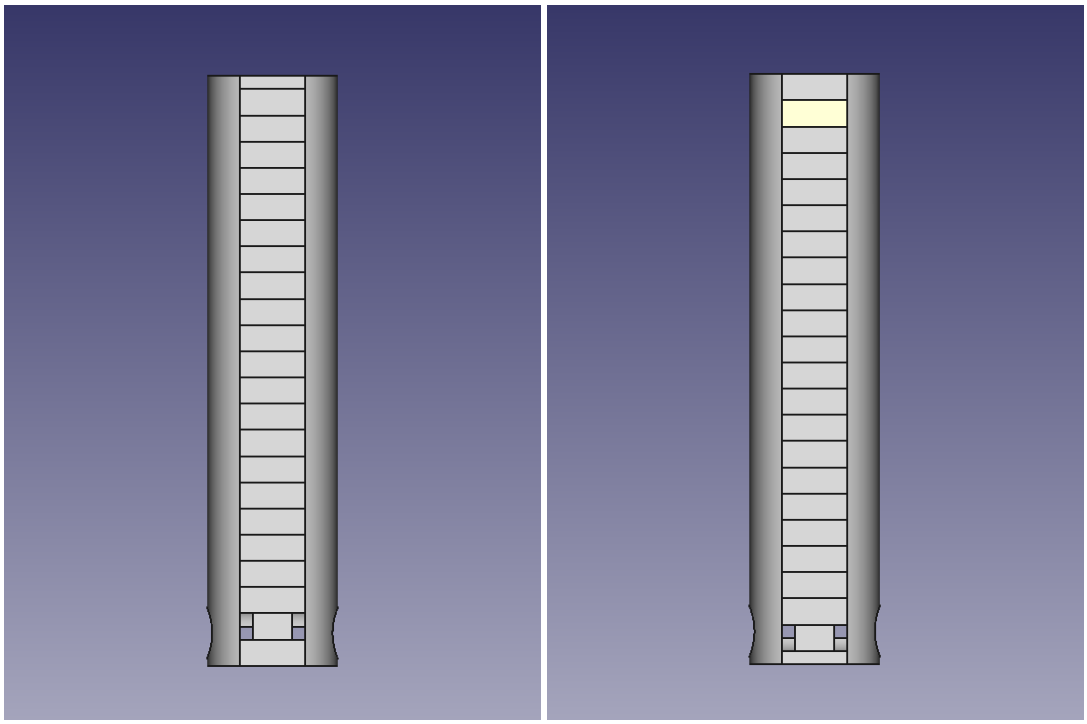


(a) (b)

Figure 5.11: Inner cylinder not staggered cad was used for clarity, (a) front, (b) back

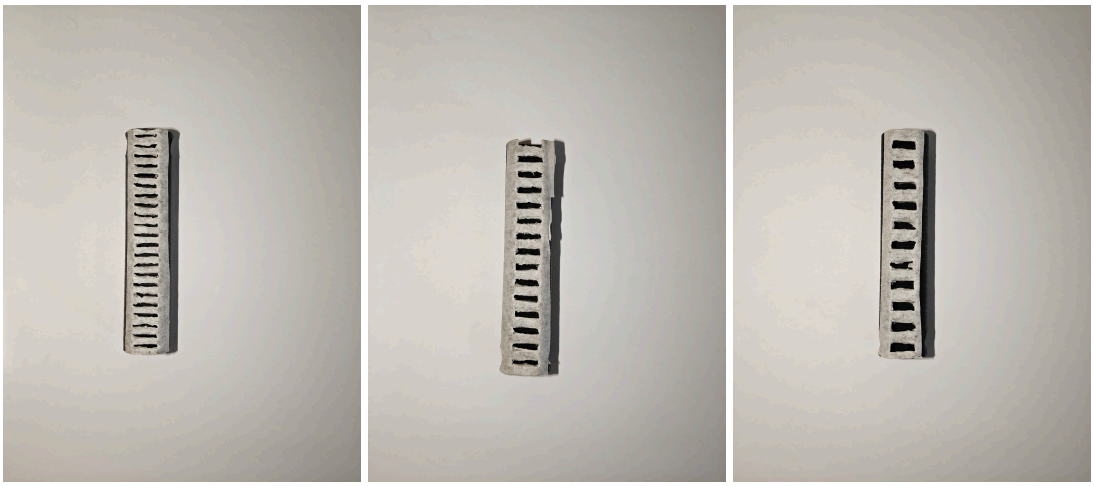
(a) (b)

Figure 5.12: Attachment Version 4: (a) front view and (b) side view



(a) (b)

Figure 5.13: Inner cylinder staggered: (a) front view and (b) side view



(a) (b) (c)

Figure 5.14: Inner cylinders for version 4, with different sizes of slots and painters tape.(a) 4 mm, (b) 3 mm, (c) 2 mm

(a) (b)

Figure 5.15: Attachment Version 4: (a) front view and (b) side view

6

Experimentation

This chapter describes how the experiments and the user study were carried out. First, we explain how delays were added to the system and how we verified that these delays were measured correctly. We also explain the different scenarios that were tested and how the scores were collected. After that, we describe how the objective latency measurements were performed. Lastly, we go over the setup used for the user study and how participants interacted with the system.

6.1. Hardware setup

We used a Novint Falcon as the haptic device. The Novint Falcon is a low-cost commercial haptic device that allows for precise measurement of its position. It can also generate force feedback in three cardinal directions. The device communicates with a computer over USB. We chose the Novint Falcon because it is easy to use, available for this project, and there is an already working code base. With some minor modifications, other haptic devices could also be used. They do need to report their position in Cartesian coordinates and be able to render force feedback.

The robotic arm we used in this work is a UFACTORY Lite 6. It can easily be mounted on a table. The robot is controlled via commands sent over Ethernet. It also has an easy mounting system so we can easily attach different kinds of end-effectors. We needed this to mount our custom sensor we developed for this research. The existing code base combines the Lite 6 with the Novint Falcon. This reduced development time and allowed us to focus on other aspects of the project.

The custom attachment uses an ESP32-C3 Super Mini microcontroller. We chose this microcontroller because it is inexpensive and fast enough to process sensor data in real time. It also includes built-in analog-to-digital conversion. This makes it possible to use analog sensors without additional hardware, the encoders required this. Communication between the microcontroller and the controlling device is done over USB through a serial interface. To detect when the encoder makes contact with objects, we used an optical A-B encoder. It uses two Otronic TCRT5000 sensor modules. They are used because they are cheap, and easy to obtain. They also provide analog outputs, which lets us use the attachment in different lightning conditions.

The haptic device, the microcontroller, and the robotic arm were all connected to a single laptop running Ubuntu 24.04.3 LTS. The laptop has an Intel i7 processor and 16 GB of RAM. The system is split into three components: the operator domain, the remote domain, and the network. Each domain runs in its own Docker container. Since everything runs on the same machine, communication between the domains goes through localhost, which acts as the network. We used this setup because it makes it easy to inject an extra delay into the communication. The containers also simplify deployment and make the setup easier to reproduce on other machines. An overview of the setup is shown in figure 6.1.

The operator domain reads the position of the Novint Falcon and generates force feedback. It runs a Bullet physics simulation at 1000 Hz, so force feedback is updated with a maximum delay of about 1 ms. The operator domain communicates with the remote domain over localhost. Messages from operator

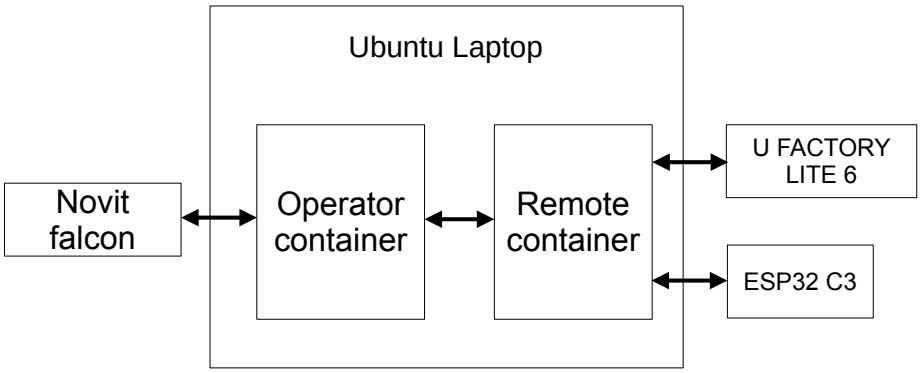


Figure 6.1: This illustration shows the containerized setup, both the operator domain and the remote domain run on the same Ubuntu laptop, but in different docker containers. The haptic device communicates with the operator container, and the ESP32 C3 and the robot communicate with the remote container. Both containers communicate over local host.

to remote mainly contain the target coordinates for the robot, and a flag indicating if the operator has acknowledged that the robot hit an object. When the operator domain receives a message from the remote side that an object was detected, it creates a virtual box at the current position of the haptic device. The acknowledgement flag is then set, so the remote side knows the operator received the collision information. Since the user study only used one object type, the height of the virtual object was hardcoded, instead of converting it from the measured robot contact location.

The remote domain contains the robot controller. It communicates with the UFACTORY Lite 6 over Ethernet, and it communicates with the ESP32 microcontroller over USB. Robot motion commands are calculated at 250 Hz, based on the target coordinates received from the operator domain. When the optical sensors reach the detection threshold, the microcontroller reports that contact occurred. From that moment on, all downward motion of the robot is stopped until the operator side confirms that the collision has been acknowledged. The remote side also sends the robot's current position and the sensor data back to the operator domain, where it is processed for the simulation and force feedback.

Even though everything was executed on one laptop during the experiments, the Docker setup still allows the system to run across multiple machines. To support this, an optional Tailscale container can be launched to connect the domains over different devices. Due to time and resource constraints, live video feedback that satisfies the required τ_{m2p} latency was not implemented in this work.

6.2. Scenarios

To evaluate our approach, we conducted a user study. In this user study, participants controlled a remote robot via a haptic device and provided subjective ratings of their experience. The goal of this study was to test whether the system functioned correctly under different delays and visibility conditions, and how these factors influenced the operator's perception of spatial awareness and control. Teleoperation performance is defined by the human user's experience, which is subjective, so we used subjective assessments for this evaluation and not objective measures.

During the user study, participants were asked to control the robot for multiple trials, each with different conditions. During each trial, the participant was instructed to operate the haptic device, and thus the robot. This to explore the remote environment. They were also asked to rate their experience in making contact with a target object. After each trial, the participant was asked to rate how the interaction felt. To ensure the ordering of the trials had no effect on the recorded user experience, a base case was

inserted between each trial, with no added delay.

The influence of changes in τ_{m2r} were tested throughout the user study. This delay varied across trials and was unknown to the participants. The approach was chosen to observe how human perception of teleoperation quality changes, specifically when the responsiveness gets worse. The tested delay values were kept within a range that allowed participants to continue controlling the robot, and high enough that differences were expected to be noticeable.

The participants were served four different cases, alongside the different additional delays. These cases reflect all combinations of two factors: whether the participant could clearly see the robot and the object in the environment, and whether the object existed in the simulation used for computing force feedback. This covers a large amount of real-world situations, including the base case where the operator can see everything and the simulation contains all relevant objects in the remote domain, when the simulation doesn't line up with the real world, how the operator interacts with an environment when their line of sight is obstructed, and when there are unexpected forces.

In the **first scenario**, participants had a clear line of sight of both the robot and the object, and the object was also inside the simulation. This scenario served as the baseline condition, because both the visual and haptic feedback aligned with each other. We expected that users would experience the interaction as natural and predictable when the delay was minimal. As the delay increased, differences between user motion and how the robot responded would gradually become noticeable. When this delay reaches a certain threshold, these differences would become too big to ignore, so the system becomes unusable.

In the **second scenario**, the simulation included the object, and the participant's line of sight was partially obstructed. The participant could still see the general movement of the robot, but could not clearly observe the detailed interaction with the object. In this case, we expected that the lack of visual feedback would force participants to rely more heavily on haptic feedback, because the simulation still contained the correct object, the feedback remained consistent, and participants were likely to interpret the behavior of the robot as normal even if some delay was present. For this reason, we expected that increasing the delay would more gradually make the user experience worse than in the baseline scenario.

The **third scenario** inverted the previous case, by having an imperfect simulation, but the participant had a clear line of sight. This would confuse the operator because, based on the visual feedback, they expected haptic feedback, but this was not immediately present because of the imperfect simulation. Increasing the delay caused the robot to move later, so it also encountered the object later, which made it so the operator experienced this feedback later. We expect that with little or no delay, the interaction would feel like there was nothing wrong with the simulation. However, once the delay increases, the inconsistency would grow much more pronounced. The participant might see the robot reach the object, but the force feedback would arrive noticeably later. We expected this scenario to feel unnatural more quickly than any other condition, and that even moderate delays would be experienced as jarring or confusing.

For the **fourth** and final **scenario** we combined limited visibility with an incomplete simulation. The participant has no clear line of sight, and the object was not present in the simulation. Because the participant did not have visual feedback for the expected interaction, we hypothesized that the absence of the object in the simulation would be less noticeable. As in the second scenario, participants would rely more on the haptic feedback that arrived when the robot eventually made contact with the object. For this reason, we expected the perceived degradation due to delay resembling that of the second scenario rather than the first or third scenarios. Small to moderate delays might not strongly affect the user experience, since the participant would not anticipate precise timing of the contact in the first place.

Together, the four scenarios allowed us to investigate how users perceive teleoperation under mismatched visual and haptic feedback, and how this user experience shifts as delay increases. By comparing ratings across scenarios, we aimed to identify whether it is possible to use this approach when there are unpredictable forces present. This by seeing if it preforms better or worse that the other cases. We also aim to find out how much delay is too much delay to still provide a decent user experience.

6.3. Evaluation

Before conducting the user study, we first verified that any added delay via software is measured correctly. The additional delay was added to all UDP communication from the operator domain to the remote domain. This is to only introduce the delay once and not in both directions. The measurements can be seen in section 7.1. The measured delay matched the configured delay closely, which confirms that setting the delay acts as expected. The user study was done with 13 participants, they are all students or ex-students at the TU-Delft. The group contained participants from multiple faculties.

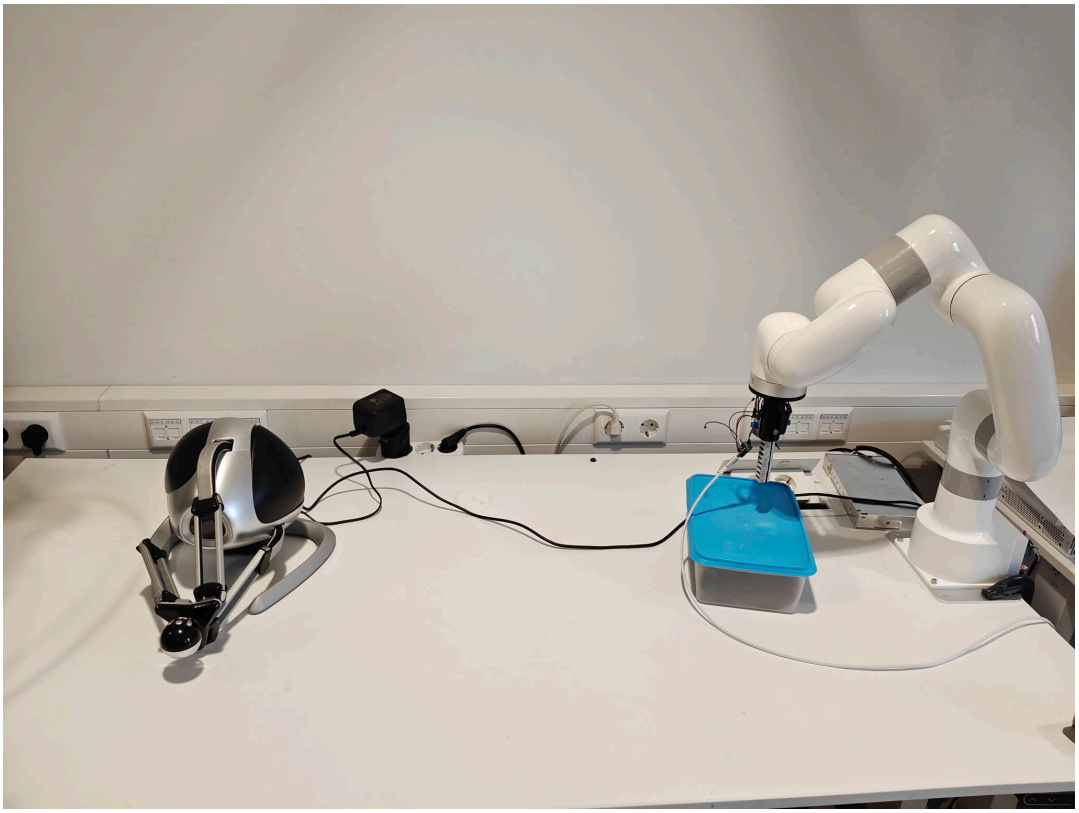


Figure 6.2: Picture of the setup of the user study, with the Novit Flacon, the UFACTORY Lite 6, and the box.

Before the user study started, participants were asked to read the explanation in appendix A. Their answers to questions asked during the user study were recorded using the table in appendix B. During the study, participants were seated at a desk with the full setup in front of them. The setup consisted of the robot arm, a physical box, and the Novint Falcon haptic device. It can be seen in figure 6.2. First, the participants were asked to get familiar with the system. They could freely move the robot around while they got familiar with the setup. The box was present in both the real world and in the simulation while the users got familiar with the system. They were encouraged to interact with the box and see how the force feedback felt.

Once the participant felt comfortable, the evaluation phase started. Participants were asked to rate their experience on a scale from 1 to 10, with 1 being the worst and 10 the best. Each time we asked them for a score, we also asked how confident they would feel using this setup in a critical application. The evaluation started with a general score for controlling the robot. After that, the participant was asked to interact with the box. First they approached the box slowly and rated the experience. Then they approached it faster and rated it again. Both slow and fast approaches could be repeated multiple times, so the participant could give a score they were confident about. This part tested the *first scenario*, where the box is visible and also present in the simulation.

After that, the participant could ask removed the box from the simulation. The physical box stayed in place, but now the simulation no longer contained it. The participant repeated the same steps again: approach the box slowly, rate it, then approach faster and rate it. This part tested the *third scenario*,

where the object exists in the real world but not in the simulation.

During these trials, an extra delay was added in the background. For each set of interactions, a delay was randomly chosen between 0 ms and 140 ms, using steps of 20 ms. This gives eight possible delay values. The delay was added to the communication between the operator domain and the remote domain, and the participant did not know which delay was active. After finishing and scoring a delay condition, the participant was given a baseline trial with no added delay. This was done to reset their frame of reference. After that, a new random delay was chosen. This continued until the participant completed all eight delay values.

When all conditions with full visibility were done, a screen was placed in front of the participant to restrict their line of sight. In this configuration the participant could still see the top part of the robot, but they could not see the box or the end effector anymore. The participant again gave a general score for controlling the robot, after which they approached the box slowly and scored the experience. Then they approached it faster and scored again. This tested the *second scenario*, where the box is in the simulation but not visible to the participant.

Finally, the *fourth scenario* was tested by removing the box from the simulation while keeping the screen in place. The participant again did slow and fast approaches and rated the experience. The same randomized delay procedure was used as before, including a baseline trial between each delay value. This was repeated until all delay conditions were tested once.

We included both slow and fast approaches because we expect moving faster makes the effects of delay worse. It increases the offset between the operator side and the robot side, and this is especially noticeable when contact happens. By comparing scores across these conditions, we could see how delay, line of sight, and simulation accuracy influence how the teleoperation system feels for the user.

7

Results

This chapter presents the results of the experiments and the user study. First, we show the objective latency measurements of the system. After that, we go over the results from the user study and how the participants rated their experience under different delays and scenarios. Finally, we summarize the main observations that can be taken from these results.

7.1. Objective results

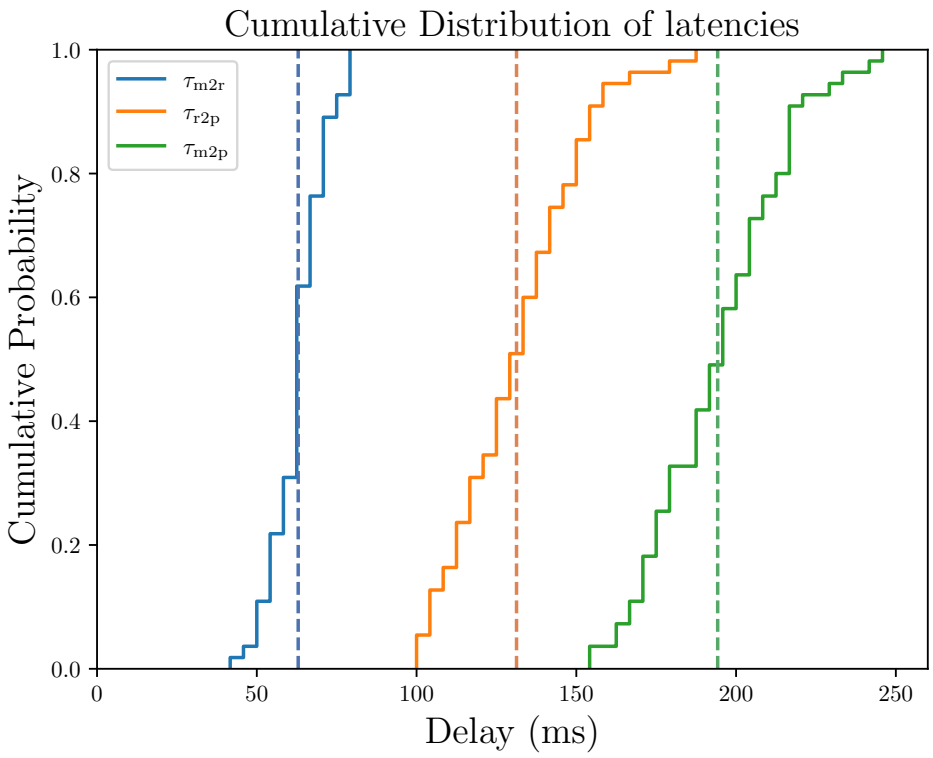


Figure 7.1: This graph contains the cumulative distribution of measured delay from a sample size of 50 measurements. The blue line (—) shows the measured τ_{m2r} this has a mean of 62.95 ms, the orange line (—) shows the measured τ_{r2p} with a mean of 131.29 ms, and lastly the green line (—) shows the measured τ_{m2p} with a mean of 194.24 ms.

First we measured τ_{m2r}, τ_{r2p}, and τ_{m2p}. To do this, we recorded the setup using a high-speed camera operating at 240 fps. This gives a timing resolution of 4.17 ms. The camera filmed the haptic device, the robot, and the monitor at the same time. We then hit the haptic device and counted the number of

frames until the robot started moving. After that, we counted how many frames it took until the robot movement was visible on the monitor. We repeated this 50 times. The results are shown in figure 7.1.

The blue curve (—) shows the cumulative distribution of the measured Motion to robot (m2r) delay τ_{m2r} over 50 samples, with a mean value of 62.95 ms. The orange curve (—) corresponds to the Robot to pixel (r2p) delay τ_{r2p}, with a mean of 131.29 ms. Finally, the green curve (—) shows the full Motion to pixel (m2p) delay τ_{m2p}, with a mean of 194.24 ms.

The measured τ_{m2r} was higher than we expected. This might be caused by some underlying issue in the code, the setup, or the devices we used. The τ_{r2p} was also higher than expected. Since both delays were larger, their sum τ_{m2p} became too large as well. It even crosses the usability boundary suggested in [20]. Because of this, the setup was not usable for our user study. We therefore decided to not use a camera feed displayed on a monitor, and instead let participants directly observe the robot with their own eyes.

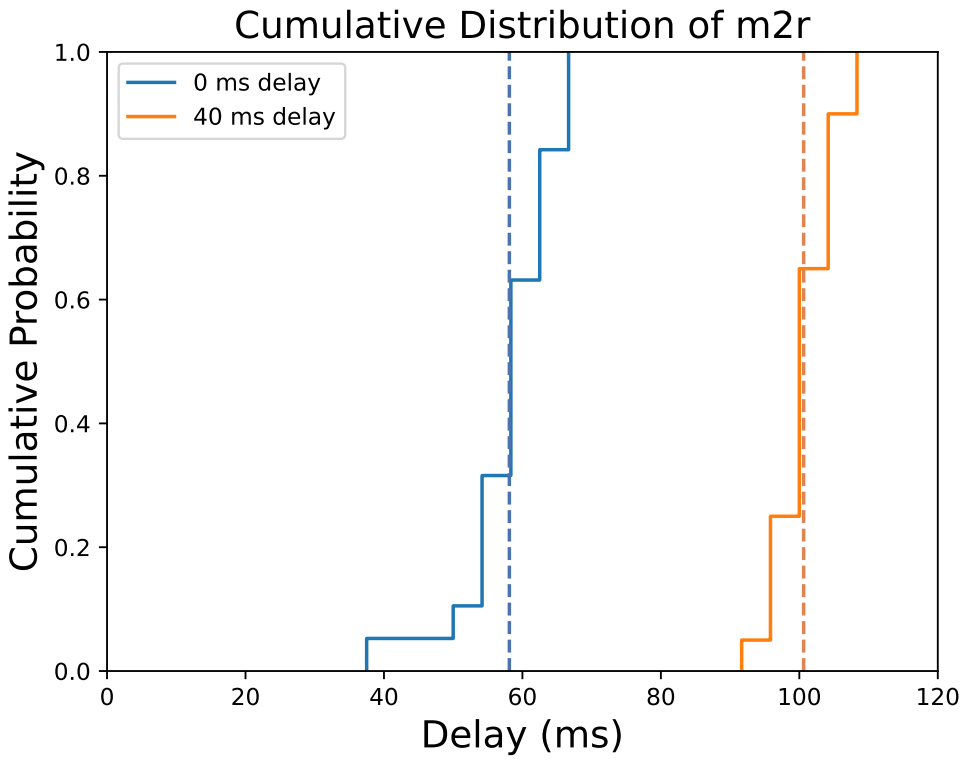


Figure 7.2: We measured τ_{m2r}, to confirm that the additional delay added via software is also measured

We also checked if adding additional delay to the communication between the haptic device and the robot, corresponds to the same additional delay in the measurements, the results of this can be seen in figure 7.2. These results were gathered from 20 samples each, they used the same 240 fps high speed camera, and the same method of generating motion. This measured delay is in line with the set delay, it is not exactly the same due to the small sample size.

7.2. Subjective results

In this section, we cover the results obtained from our user study. We asked the participants how comfortable they were with using the technology for a critical application under the current conditions, and rate this from one to ten. One being the worst, and ten the best. The results can be seen in figure 7.3. Here, the blue line (—) represents the score given to the *first scenario*. The doted blue line (- -) represents the *second scenario*. The orange line (—) represents the *third scenario*, and the doted orange line (- -) represents the score given to the *fourth scenario*.

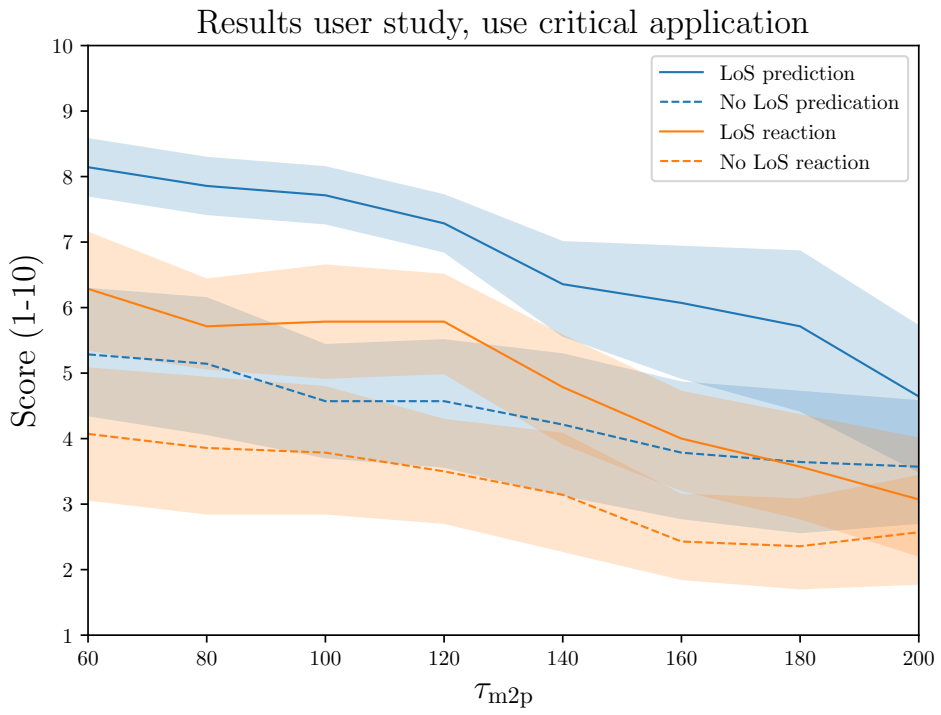


Figure 7.3: Testing how comfortable a participant is to use the technology in a critical task, to test this random delays were added from a range between 0 ms and 140 ms in increments of 20 ms. The delay is added from the operator domain to the remote domain, in one direction. The operator was tasked with making contact with a box, and apply force to it, both when they have a clear line of sight represented by the full lines, and the dotted lines represent when they did not have a direct line of sight

From this we can see that participants felt more comfortable when they had a direct line of sight. There is also a large difference in the given confidence level when the participants had a clear line of sight, based on if the force feedback was based on prediction or reaction. When there is no direct line of sight, this difference is smaller.

We also asked the participants to grade the general effectiveness of the provided visual feedback based on how the robot moves around in relation to what the operator does, the results can be seen in figure 7.4. Here, the blue line (—) represents the participant's general assessment of the effectiveness of the visual feedback when they have a clear line of sight. The doted blue line (- -) shows the score when they don't have a direct line of sight with the object.

Overall, the general performance of the visual feedback degrades as the additional delay increases. The participants graded the visual feedback a bit worse when they couldn't see the whole robot. This is in line with the previous results, where the participants indicate that they are felt less confidant when there was no direct line of sight.

In another part of the user study, the participants were tasked with making contact with a box, and applying force onto this box while having a clear line of sight, they were asked to rate the quality of the force feedback. The results of this can be seen in figure 7.5. Here, the blue line (—) represents the score given to the force feedback when the participant approached the box slowly, and the box was present in the simulation. The orange line (—) represents the same interaction, but the user moves faster. Together they encapsulate the *first scenario*, where the operator has a clear line of sight, and the object encountered is in the simulation. The scores for slow and fast approaches remain approximately the same.

In the same graph figure 7.5, the doted blue line (- -) represents the score given to the force feedback when the participant had a clear line of sight, and they approached the object slowly, but the object was not in the simulation. The doted orange line (- -) represents the same, but with a faster approach.

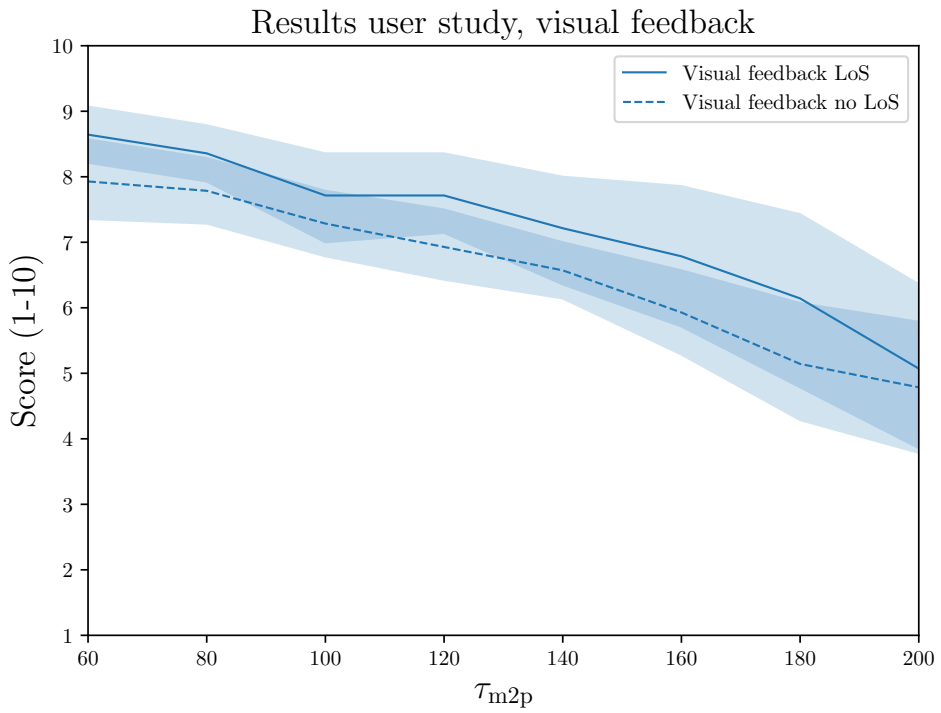


Figure 7.4: Testing the effectiveness of visual feedback, to test this random delays were added from a range between 0 ms and 140 ms in increments of 20 ms. The delay is added from the operator domain to the remote domain, in one direction.

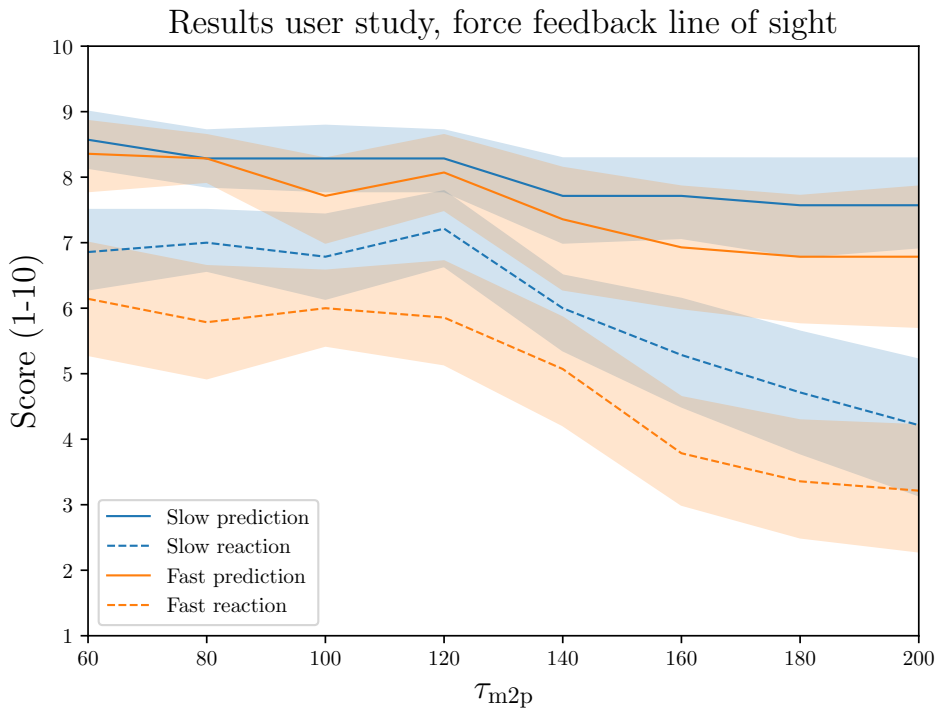


Figure 7.5: Testing the effectiveness of the provided force feedback, to test this random delays were added from a range between 0 ms and 140 ms in increments of 20 ms. The delay is added from the operator domain to the remote domain, in one direction. The operator was tasked with making contact with a box, and apply force to it, they had a clear line of sight on the box.

These two lines represent the *third scenario*, where the operator has a clear line of sight, but the object is not in the simulation. Here, the faster interaction gets a worse score than the slower interaction. We think this is because the offset is based on the speed of the haptic device. They both score significantly less than their counterparts based on a simulation. After τ_{m2p} 120 ms there is a steep decline in the scores given to both fast and slow approaches, but the difference between them stays about the same.

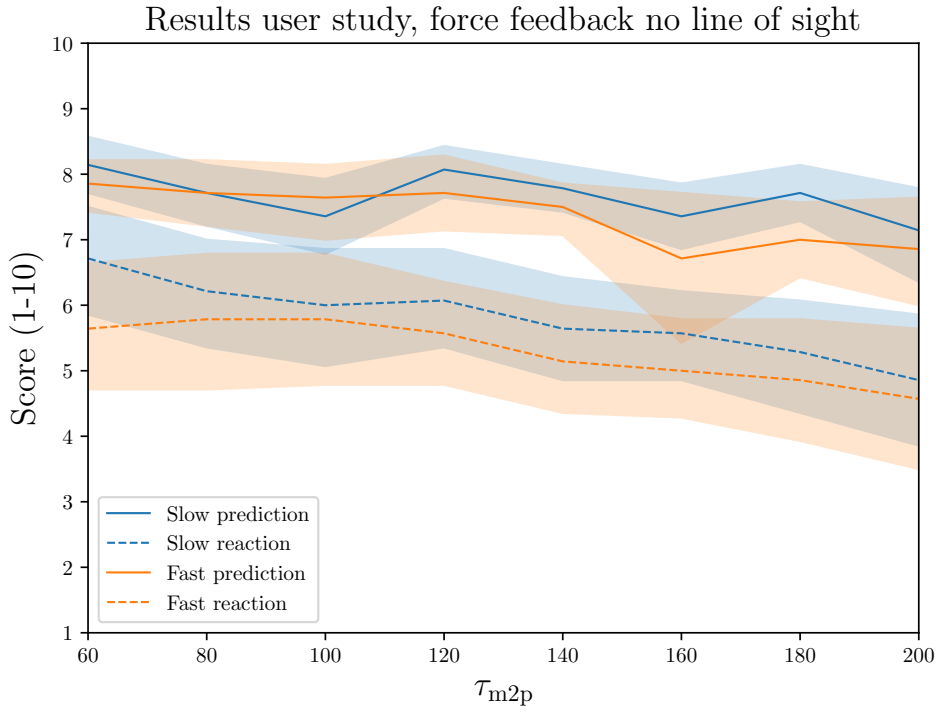


Figure 7.6: Testing the effectiveness of the provided force feedback, to test this random delays were added from a range between 0 ms and 140 ms in increments of 20 ms. The delay is added from the operator domain to the remote domain, in one direction. The operator was tasked with making contact with a box, and apply force to it, they had no clear line of sight of the box.

The results from when the participant had no direct line of sight can be seen in figure 7.6, here the blue line (—) represents the participant slowly making contact with the box, while it is in the simulation, and they have no clear line of sight. The orange line (—) shows what happens when the operator approaches the box faster. Together they represent the *second scenario*, where the operator has no direct line of sight, and the simulation contains the right object. Here, the scores given to the fast, and slow interaction are about the same.

In the same graph figure 7.6, the doted blue line (- -) represent the score the participants gave to the force feedback when they slowly approached an object not in the simulation, while they have no clear line of sight. The doted orange line (- -) represents the same, but with a faster approach. Both of them can be used to evaluate the *fourth scenario*, where the operator has no clear line of sight, and they approach and make contact with an object that is not in the simulation. Here, the scores given to the slow and fast interaction also don't really deviate. But when comparing to the case where the user had a clear line of sight, their score is significantly better. These scores are also only a bit worse than when using prediction.

In our user study, we looked at both fast and slow approaches. We expected that the fast approaches preformed significantly worse than slow approaches when τ_{m2p} got higher. This because we believed that the higher velocities would have a greater impact with higher delays. When looking at the result, however, we can see that the difference between fast and slow approaches remain constant within each scenario. The difference is the highest in scenario 3 where the operator has a direct line of sight, and the force feedback is based on reaction. But across all the scenario's there is a difference, the difference

doesn't seem to get larger when the τ_{m2p} gets larger. This answers sub question three: "How does the movement speed of the operator effect the performance of a Haptic Bilateral Teleoperation (HBT) system". We believe the deviation between our expectation and the results could be due to the smaller sample size, or due to a specific way we asked the questions during the user study.

Inference 1: While the fast approach consistently scores a little bit worse than the slow, with direct line of sight there is a larger difference than no direct line of sight, the difference is not influenced by latency

We also tested how reaction based force feedback preformed when the operator has a direct line of sight and when they do not. Our expectation was that when there was a direct line of sight, the reaction based force feedback would preform worse than when there was no direct line of sight. This because when the force feedback is based on reaction, the real world and virtual offset becomes larger with larger τ_{m2p}. When there is direct line of sight, the operator can see this discrepancy, but when there is not, the operator cannot see it. We can see from the results that our expectation was correct, reaction based force feedback preformed better when there was no direct lien of sight, specially when τ_{m2p} increases. There is an addition observation that can be made. when there is direct line of sight the performance reaction based force feedback quickly drops off after a τ_{m2p} of 120 ms. When there is no direct line of sight, this dropoff doesn't happen. This answers part of our first sub question: "What are the effects of having no direct line of sight on the area of operation?". We can see from the results that it negatively influences the confidence level of the operator, but for the perceived quality of force feedback it has a positive effect.

Inference 2: Reaction based force feedback degrades at a significant rate when there is a direct line of sight, and a τ_{m2p} larger than 120 ms, The same doesn't happen when there is no direct line of sight, here the score given is much more stable.

Inference 3: The requirements on τ_{m2p} with a direct line of sight for reliable force feedback based on reaction are harsher than those required for visual feedback."

As a sort of sanity check, we also tested the performance of prediction based feedback. Here, we expect the force feedback to get the same score across all the different τ_{m2p} tested. We also expect there to be almost no difference between the fast and slow approaches. We also expect the difference between reaction based force feedback and prediction based force feedback when there is no direct line of sight to remain constant between the different τ_{m2p}. From the results, we can see that the score given to prediction based feedback remains relatively stable, but it gets a slightly worse when τ_{m2p} gets higher. We believe this is because of the degrading quality of visual feedback, and not because of the prediction itself. There is a slight difference between the fast approach and the slow approaches, this is likely caused by the greater velocity making every interaction a bit worse. The difference between reaction and prediction remained constant when there was no direct line of sight, leading us to believe that another reaction implementation that preforms closer to how prediction preforms could replace it. This answers our second sub question: "Can reaction based force feedback be used when prediction is not possible?", an experienced user can use reaction based force feedback when there is no direct line of sight.

Inference 4: When direct line of sight is not possible and there is no reliable simulation, then reaction based force feedback can be used.

Inference 5: The requirements on, τ_{m2p} when there is no direct line of sight, are less harsh for reaction based force feedback than what they are for visual feedback.

Bringing it all together, we expected that an operator could use reaction instead of prediction when there is no direct line of sight. From these results we can conclude that this is possible, but a better implementation for reaction based force feedback should be implemented to make it better. This answers our research question: "How can a long distance Haptic Bilateral Teleoperation operator handle unpredictable forces?"

7.3. Discussion

To answer our research question, we had a clear technical goal, provide an environment where an operator can use HBT to control a robotic arm, and receive force feedback based on prediction and

reaction. How we achieved this was not linear. We learned several lessons from this project. The biggest takeaway is that it is easy for infrastructure work to consume a lot of time, especially if it is not bounded early on. At the start we spent a lot of time getting Tailscale and Headscal to work with our local Docker setup. While it created an environment that was easier to use, and able to be tested in multiple environments, but it costed a lot of time. At the time early on the project, our daily supervisor became a father, this understandably lead to less direct supervision, which caused us to spend longer on the Tailscale Headscale path than necessary. This taught us that it is important to define when a solution is good enough to be used, instead of trying to make everything perfect.

Another takeaway is that taking user studies required more time than we anticipated, which in part lead to the relatively small user sample size. The technical aspect is one part, but once it works, there are still a lot of things that need to be tweaked before the experiments can be conducted. Also when gathering the data it is important to ask the right questions, such that the results gathered are usable, and not contaminated by noise or other factors. This taught us that it is important to think about the user study beforehand, and make a clear plan of what needs to be done. This all leads to another thing we learned, that is to keep track on the main goal, and not lose ourselves in the smaller details that in the end weren't important to conduct the research.

8

Conclusion

8.1. Conclusion

Long distance teleoperation has a wide range of applications. Long range Haptic Bilateral Teleoperation (HBT) in particular can be used for surgery, training, remote work, and disaster relief. One of the biggest challenges in long distance HBT is communication delay. When force feedback is based directly on sensors in the remote environment, large delays make the system unstable and hard to use. To address this, previous work has used simulations of the remote environment on the operator side to generate force feedback locally, while providing live video of the real environment. This allowed systems to remain usable with a τ_{m2p} below 150 ms. Other approaches even used simulation for visual feedback. However, these methods fail when the robot encounters unpredictable forces, meaning objects that are not present in the simulation and not visible to the operator. The system then does not know how to react. This leads to our research question: “How can a long distance Haptic Bilateral Teleoperation operator handle unpredictable forces?”

In this thesis, we proposed a solution that still uses a local simulation for predictive force feedback, but also adjusts this simulation using sensors in the remote environment. To test this idea, we designed a 3D printed attachment that can be mounted on a robotic arm to detect when the robot encounters an object. The sensor data is sent back to the operator side and used to update the simulation, allowing the operator to feel forces from objects that were not originally modeled. We evaluated this approach with a user study involving 13 participants. The results showed that operators strongly preferred having a direct line of sight and a correct simulation. Predictive force feedback performed best in these conditions. However, when the line of sight was blocked, reaction based force feedback using sensor updates still allowed operators to perform tasks with reasonable confidence. The quality of the force feedback didn’t significantly degrade when τ_{m2p} get larger. This means that the requirements on τ_{m2p} for force feedback is less strict than the requirement for visual feedback. When the operator could clearly see the robot, this reaction based approach quickly became unusable as τ_{m2p} went over 120 ms.

Overall, the results suggest that combining predictive force feedback with remote sensing is a viable way to handle unpredictable forces in long distance HBT. While a perfect simulation remains the ideal case, sensor-based updates can provide a practical fallback when the environment is only partially known.

8.2. Future work

To improve this research, specifically looking into handling unpredictable forces in HBT, you should conduct a larger and more elaborate user study. The current study has some useful insights, but the amount of participants was limited. Increasing this number would make the results more statistically reliable. There are also more scenarios that can be tested.

In this thesis, we focused the user study on making contact with a rigid object in a relatively simple

setup. Further research should include objects made of different materials, such as soft materials, or bouncy materials. The shape of the object could also have more variation, like boxes with different heights, or spherical object. This would show if the approach is still viable when the expected contact is different.

We also only tested for unpredictable forces in one known direction, but in the real world they can come from anywhere. So further research should look into other directions, or maybe even multiple directions at the same time. This would make the offset and eliminating it more complex.

There is a big scenario we did not explore in our user study, that is when there is an object inside the simulation, but there is no corresponding object in the real world. This could lead to a weird experience, especially when the operator has a direct line of sight.

Future work should also include actual visual feedback that satisfies the $\tau_{\mathsf{m2p}} < 150\ \mathrm{ms}$ requirement. A larger user study could also deduce specific values for the maximal acceptable τ_{m2p} for giving usable force feedback in each scenario, and see how this lines up with the related work.

References

[1] R.J. Anderson and M.W. Spong. “Bilateral control of teleoperators with time delay”. In: *Proceedings of the 1988 IEEE International Conference on Systems, Man, and Cybernetics*. Vol. 1. 1988, pp. 131–138. DOI: 10.1109/ICSMC.1988.754257.

[2] A. Aziminejad et al. “Stability and performance in delayed bilateral teleoperation: Theory and experiments”. In: *Control Engineering Practice* 16.11 (2008), pp. 1329–1343. ISSN: 0967-0661. DOI: https://doi.org/10.1016/j.conengprac.2008.03.005. URL: https://www.sciencedirect.com/science/article/pii/S0967066108000397.

[3] Taran Batty et al. “A Transparent Teleoperated Robotic Surgical System with Predictive Haptic Feedback and Force Modelling”. In: *Sensors* 22.24 (2022). ISSN: 1424-8220. DOI: 10.3390/s22249770. URL: https://www.mdpi.com/1424-8220/22/24/9770.

[4] David G. Black, Dragan Andjelic, and Septimiu E. Salcudean. “Evaluation of Communication and Human Response Latency for (Human) Teleoperation”. In: *IEEE Transactions on Medical Robotics and Bionics* 6.1 (2024), pp. 53–63. DOI: 10.1109/TMRB.2024.3349612.

[5] Francis Boabang et al. “A Framework for Predicting Haptic Feedback in Needle Insertion in 5G Remote Robotic Surgery”. In: *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*. 2020, pp. 1–6. DOI: 10.1109/CCNC46108.2020.9045432.

[6] Francheska B. Chioson et al. “Implementation of a Bilateral Teleoperation Haptic Feedback Controls to Robotic Minimally Invasive Surgery”. In: *2020 IEEE 12th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*. 2020, pp. 1–5. DOI: 10.1109/HNICEM51456.2020.9400009.

[7] J.E. Colgate and J.M. Brown. “Factors affecting the Z-Width of a haptic display”. In: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. 1994, 3205–3210 vol.4. DOI: 10.1109/ROBOT.1994.351077.

[8] Kourosh Darvish et al. “Teleoperation of Humanoid Robots: A Survey”. In: *IEEE Transactions on Robotics* 39.3 (2023), pp. 1706–1727. DOI: 10.1109/TRO.2023.3236952.

[9] Yaru Deng et al. “A Review of Bilateral Teleoperation Control Strategies with Soft Environment”. In: *2021 6th IEEE International Conference on Advanced Robotics and Mechatronics (ICARM)* (2021), pp. 459–464. URL: https://api.semanticscholar.org/CorpusID:237521221.

[10] J. E. Domínguez-Vidal and Alberto Sanfeliu. “Improving Human-Robot Interaction Effectiveness in Human-Robot Collaborative Object Transportation Using Force Prediction”. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2023, pp. 7839–7845. DOI: 10.1109/IROS55552.2023.10342517.

[11] Wen Fan et al. “Digital Twin-Driven Mixed Reality Framework for Immersive Teleoperation With Haptic Rendering”. In: *IEEE Robotics and Automation Letters* 8.12 (2023), pp. 8494–8501. DOI: 10.1109/LRA.2023.3325784.

[12] Gregory M. Fayard. “Fatal Work Injuries Involving Natural Disasters, 1992–2006”. In: *Disaster Medicine and Public Health Preparedness* 3.4 (2009), pp. 201–209. DOI: 10.1097/DMP.0b013e3181b65895.

[13] Navid Feizi et al. “Robotics and AI for Teleoperation, Tele-Assessment, and Tele-Training for Surgery in the Era of COVID-19: Existing Challenges, and Future Vision”. In: *Frontiers in Robotics and AI* 8 (Apr. 2021). DOI: 10.3389/frobt.2021.610677.

[14] William R. Ferrell and Thomas B. Sheridan. “Supervisory control of remote manipulation”. In: *IEEE Spectrum* 4.10 (1967), pp. 81–88. DOI: 10.1109/MSPEC.1967.5217126.

[15] Glebys Gonzalez et al. “DESERTS: DElay-tolerant SEmi-autonomous Robot Teleoperation for Surgery”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 12693–12700. DOI: 10.1109/ICRA48506.2021.9561399.

[16] B. Hannaford and Jee-Hwan Ryu. “Time-domain passivity control of haptic interfaces”. In: *IEEE Transactions on Robotics and Automation* 18.1 (2002), pp. 1–10. DOI: 10.1109/70.988969.

[17] Armand Jordana et al. “Force Feedback Model-Predictive Control via Online Estimation”. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. 2024, pp. 11503–11509. DOI: 10.1109/ICRA57147.2024.10611156.

[18] Sébastien Kleff et al. “Introducing Force Feedback in Model Predictive Control”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 13379–13385. DOI: 10.1109/IROS47612.2022.9982003.

[19] H. J. C. Kroep, V. Gokhale, and R. Venkatesha Prasad. “Blind Spots of Objective Measures: Exploiting Imperceivable Errors for Immersive Tactile Internet”. In: *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*. 2022, pp. 01–10. DOI: 10.1109/ICCPS54341.2022.00011.

[20] Herman Kroep et al. “Breaking the Latency Barrier: Practical Haptic Bilateral Teleoperation over 5G”. In: *Proceedings of the ACM/IEEE 16th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2025)*. 2025, pp. 1–11.

[21] Arnaud Lelevé, Troy McDaniel, and Carlos Rossa. “Haptic Training Simulation”. In: *Frontiers in Virtual Reality* 1 (2020). DOI: 10.3389/frvir.2020.00003.

[22] S Lichiardopol. “A survey on teleoperation”. In: (2007).

[23] Johanna Miller et al. “Impact of haptic feedback on applied intracorporeal forces using a novel surgical robotic system—a randomized cross-over study with novices in an experimental setup”. In: *Surgical Endoscopy* 35.7 (2020), pp. 3554–3563. DOI: 10.1007/s00464-020-07818-8.

[24] Syeda Nadiah Fatima Nahri, Shengzhi Du, and Barend Jacobus Van Wyk. “A Review on Haptic Bilateral Teleoperation Systems”. In: *Journal of Intelligent & Robotic Systems* 104.1 (Dec. 2021), p. 13. ISSN: 1573-0409. DOI: 10.1007/s10846-021-01523-x. URL: https://doi.org/10.1007/s10846-021-01523-x.

[25] Loïc Niederhauser et al. “A Predictive Model for Tactile Force Estimation Using Audio-Tactile Data”. In: *IEEE Robotics and Automation Letters* 9.2 (2024), pp. 1596–1603. DOI: 10.1109/LRA.2023.3340614.

[26] Rajni V. Patel, S. Farokh Atashzar, and Mahdi Tavakoli. “Haptic Feedback and Force-Based Teleoperation in Surgical Robotics”. In: *Proceedings of the IEEE* 110.7 (2022), pp. 1012–1027. DOI: 10.1109/JPROC.2022.3180052.

[27] Karan Rangarajan, Heather Davis, and Philip H. Pucher. “Systematic Review of Virtual Haptics in Surgical Simulation: A Valid Educational Tool?” In: *Journal of Surgical Education* 77.2 (2020), pp. 337–347. ISSN: 1931-7204. DOI: https://doi.org/10.1016/j.jsurg.2019.09.006. URL: https://www.sciencedirect.com/science/article/pii/S1931720419306609.

[28] K. Salisbury et al. “Haptic rendering: programming touch interaction with virtual objects”. In: *Proceedings of the 1995 Symposium on Interactive 3D Graphics*. I3D ’95. Monterey, California, USA: Association for Computing Machinery, 1995, pp. 123–130. ISBN: 0897917367. DOI: 10.1145/199404.199426. URL: https://doi-org.tudelft.idm.oclc.org/10.1145/199404.199426.

[29] M. Salvato et al. “Predicting Hand-Object Interaction for Improved Haptic Feedback in Mixed Reality”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 3851–3857. DOI: 10.1109/LRA.2022.3148458.

[30] Loulwa M. Al□Saud. “The utility of haptic simulation in early restorative dental training: A scoping review”. In: *Journal of Dental Education* 85.5 (Dec. 2020), pp. 704–721. DOI: 10.1002/jdd.12518.

[31] Thomas B Sheridan. *Telerobotics, automation, and human supervisory control*. MIT press, 1992.

[32] D. Wang, K. Ohnishi, and W. Xu. “Novel Emerging Sensing, Actuation, and Control Techniques for Haptic Interaction and Teleoperation”. In: *IEEE Transactions on Industrial Electronics* 67.1 (2020), pp. 624–626. DOI: 10.1109/TIE.2019.2927784.

[33] Hitoshi Watanabe et al. “Toward the optimal control of haptic communication -Introduction of force prediction to collaborative work-”. In: *2021 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*. 2021, pp. 1–2. DOI: 10.1109/ICCE-TW52618.2021.9602982.

[34] Yang Ye et al. “Brain functional connectivity under teleoperation latency: a fNIRS study”. In: *Frontiers in Neuroscience* 18 (2024). DOI: 10.3389/fnins.2024.1416719.

[35] C.B. Zilles and J.K. Salisbury. “A constraint-based god-object method for haptic display”. In: *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*. Vol. 3. 1995, 146–151 vol.3. DOI: 10.1109/IROS.1995.525876.

A

Explanation user study

The goal of the experiment is to figure out the perceived quality of the feedback provided to the operator, where we look separately at visual and force feedback. Additionally we are interested in the perceived quality of the setup if used to perform a delicate task. We are also interested in observing the perceived effect of visual obstruction on performing a task.

In this study, you will control a robotic arm using a haptic device. A box is positioned beneath the robotic arm, and you can apply force to this box. In an ideal case, the robot feels like an extension of your hand, where you have full control over manipulating both the trajectory of the robot, and the amount of force it applies to its environment.

We are comparing two methods to obtaining force feedback. For every configuration, both methods will be examined. You will then repeat the same interaction steps (slow and fast) and rate them again also indicate how confident you would feel using the system in a critical application. Each delay condition is tested once. After seven trials, all delay conditions will have been completed. A visual screen will then be placed between you and the system, and you will be asked to repeat the same procedure under this condition.

Procedure

You will first be given time to familiarize yourself with the robotic system and the haptic device. After this, the trials will begin. In each trial, a random delay between 0 and 140 milliseconds will be applied, in steps of 20 milliseconds.

For each trial, you will:

- Observe the visual feedback and rate it on a scale from 1 to 10.
- Interact with the box by first applying force slowly, and then more quickly.
- During this phase, the box is present in the simulation (the prediction condition).
- Rate both interaction speeds separately.
- Indicate how confident you would feel using this system in a critical application, such as surgery.

Next, the box will be removed from the simulation (the reaction condition). You will then repeat the same interaction steps (slow and fast) and rate them again also indicate how confident you would feel using the system in a critical application. Each delay condition is tested once. After seven trials, all delay conditions will have been completed. A visual screen will then be placed between you and the system, and you will be asked to repeat the same procedure under this condition.

B

table user study

Line of sight								
	Trail 1	Trail 2	Trail 3	Trail 4	Trail 5	Trail 6	Trail 7	Trail 8
Visual performance								
Slow force feedback prediction								
Fast force feedback prediction								
Use in a critical application prediction								
Slow force feedback reaction								
Fast force feedback reaction								
Use in a critical application reaction								
No line of sight								
	Trail 1	Trail 2	Trail 3	Trail 4	Trail 5	Trail 6	Trail 7	Trail 8
Visual performance								
Slow force feedback prediction								
Fast force feedback prediction								
Use in a critical application prediction								
Slow force feedback reaction								
Fast force feedback reaction								
Use in a critical application reaction								

Table B.1: Table used to gather results from the user study