CONTROL AND FORCE SENSING FOR A DIY AT HOME TELEMANIPULATION SETUP

V.D.C. (Victor) Bergsma

BSC ASSIGNMENT

Committee:
dr. ir. D. Dresscher
L.B.L. Lenders, MSc
dr. ing. G. Englebienne

July, 2024

030RaM2024
Robotics and Mechatronics
EEMCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Abstract

In 1948, Raymond C. Goertz designed what is considered to be the first successful bilateral (two-sided) teleoperation system to manipulate radioactive material from behind a leaded wall. Since then, teleoperation has advanced to completely remote systems with very high transparency, simulating a direct connection with the task environment to perform high-precision tasks such as surgery remotely. Avatars.Report aims to educate people about this increasingly adopted technology through video lectures, assignments, and a low-cost DIY kit: the focus of this paper.

Firstly, the quality of the DIY kit's force measurement was to be improved, found to be affected most by physical disturbances, gravity and electromagnetic interference from the motor. Except for intense disturbances, satisfactory attenuation of undesired signals has been achieved with a Least-Mean-Square adaptive filter. This filter matches the noise measured by a 'dummy' load-cell (force sensor) to the noise in the 'active' load-cell's measurement. To prevent distortion, basic noise estimation replaces this filter whenever significant force is applied to the handle.

Using this improved force measurement, two control architectures were to be implemented. While the 4-channel architecture was not completed, the position-measured force architecture significantly increased the transparency over the existing position-computed force architecture; when interacting with a rigid object and springs, the average force and position tracking errors were reduced by 61.7 % and 31.6 % respectively. However, due to deviating and temperature-dependent motor constants, these percentages may vary between DIY kits.

Finally, the load-cells' amplifiers were modified to increase their low sampling rate and thereby improve the system's stability. Due to increased noise in the force measurement that could not be attenuated by the conditional LMS filter, using the modified load-cell amplifiers to obtain an increased sampling rate did not improve the system's transparency.

Acknowledgement

I would like to thank my supervisor, Douwe Dresscher, for his guidance and feedback that helped me to improve my approach to conducting and presenting research. Furthermore, I would like to thank my family for their support throughout the past 10 weeks.

Declarations

During the preparation of this work, the author used the built-in spell checker of Overleaf. After using this tool/service, the author reviewed and edited the content as needed and takes full responsibility for the content of the work.

Contents

1 Introduction

1.1 History, meaning & terminology

More than 70 years ago, Raymond C. Goertz designed a mechanical contraption that enabled a person to manipulate material from a nuclear reactor from behind a leaded wall. This is generally considered to be the first successful bilateral (i.e. two-sided) teleoperation system [1; 2]. Since then, teleoperation has advanced to completely remote systems, communicating wirelessly, which can provide very accurate force replication such that high-precision tasks like surgery can now be performed through teleoperation. Evenmore, by scaling up or down the replicated forces, teleoperation can enable the operator to perform very heavy or very precise (remote) tasks [3].

In Ancient Greek, *tele* means *at a distance*. So, one might deduce that telemanipulation must be manipulation of something from a distance, which overlooks one important aspect: feedback. This "feedback" is very important for being able to successfully complete a remote task through the teleoperation system and can be (a combination of) visual, tactile or kinesthetic. Most people are familiar with visual feedback, for example a live video feed of the remote task, or tactile feedback, like a vibration to indicate an obstacle. However, kinesthethic feedback is much less well-known: feedback which provides sensation of forces in muscles and bones [4].

The main objective of teleoperation is to make the operator feel as though they are directly in contact with the remote task through the teleoperation system. This *feel* largely depends on the *transparency* of the system: the fidelity with which the operator can perceive and control the remote task. In most teleoperation systems, (a selection of) the forces and velocities of both robots are communicated wirelessly, which includes delays. These delays compromise the transparency, so current research focuses on optimising the transparency in presence of communication delays.

While not applicable for this assignment, there is another type of teleoperation: model-mediated teleoperation, where the model of the remote task environment is estimated locally on the human operator's side of the system with the model parameters it receives from the robot in the task environment. This can provide a much higher transparency in case of time delays in the communication channel; the haptic feedback is computed based on the locally estimated model instead of (delayed) communicated forces and velocities [5].

In (older) literature about teleoperation systems (e.g. [1; 6; 7]), the terms "master" and "slave" are used: the "master" refers to the robot with which the human operator interacts and the "slave" refers to the robot that is controlled (remotely) by the operator to perform a task. However, these are historically sensitive terms. Therefore, in recent literature (e.g. [8]) other terms are used to replace them. In this report, the "master" will mostly be referred to as the robot on the human side and the "slave" as the robot in the task environment performing the remote task, both of which typically are robotic arms.

1.2 Applications and challenges

As mentioned, modern teleoperation can enable the operator to perform very heavy or precise remote tasks, which has many use-cases such as space explorations, unmanned underwater-/military vehicles, telesurgery and handling of hazardous materials [9] (for which the first teleoperation system was designed by Goertz). The main challenge for these systems is achieving high transparency with a stable system, particularly when the communication delays are large.

Nonetheless, there are also many applications closer to most people's everyday lives like interacting with a remote expert, caretaker or friend. These applications often use robots that are more similar to the human body: robotic avatars. By definition, an avatar is a physical embodiment of a god descending (*avatāra* in Sanskrit) to earth. Hence, a robotic avatar is the physical embodiment of a human in the form of a robot. This form of teleoperation faces some extra (ethical) challenges like privacy and safety, but also sustainability and energy consumption in case of widespread use.

1.3 Avatars.Report & the DIY kit

Since many people are unfamiliar with teleoperation and kinesthethic feedback, Avatars.Report aims to educate them about this increasingly adopted technology with video lectures, assignments and a DIY kit. This DIY kit (Figure 1.1) consists of two identical setups that are made of 3D-printed and low-cost commercially available parts, and can be used for control and telemanipulation experiments.

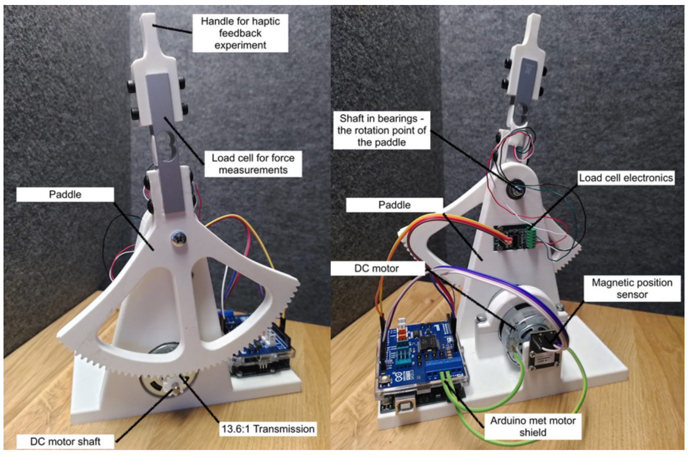


Figure 1.1: Avatars.Report "DIY @ Home setup", front view (left) and rear view (right) [10]. Each DIY kit consists of two of these setups.

1.4 Problem statement

The DIY telemanipulation kit is not quite finished yet; while the position-position and position-computed force architectures have been implemented in a graphical user-friendly interface, the position-measured force and 4-channel architectures have not. Also, the current implementation and calibration of the force sensor (a cost-effective load-cell) does not provide an accurate measurement for all forces in the required range of 0 - 15 N, which is not beneficial for the system's performance.

1.5 Assignment

The thesis involves the integration of a relatively cheap force sensor (load-cell) in the DIY telemanipulation setup from Avatars.Report used to educate people about robotics. Also, the position-measured force and 4-channel control architectures are to be implemented with adjustable gains and time delay on an Arduino with a "Processing" interface: an IDE (integrated development environment) with a more user-friendly graphical interface than the Arduino IDE.

1.6 Research questions

This problem statement and assignment description lead to the following main research question that is to be answered in this thesis:

How can the position-measured force and 4-channel architectures be realised on the Avatars.Report DIY at Home lab setup; what are the critical design aspects and how can these

be addressed using the available hardware and software resources? Furthermore, how can a low-cost force sensor be effectively utilised for force sensing in a teleoperation system?

To aid the process of answering this main research question, multiple underlying research questions have been formulated. First, the integration of the control architectures in the DIY kit is explored by answering:

1. Which control architectures are most commonly used in teleoperation systems and in which settings are they commonly used?
2. How should the performance of a control architecture be evaluated in the context of a cost-effective DIY teleoperation system?
3. How is the position-measured force control architecture constructed and how can it be implemented in a teleoperation system?
4. What are the (dis)advantages of the position-measured force architecture compared to other control architectures in the context of teleoperation?
5. How is the 4-channel control architecture constructed and how can it be implemented in a teleoperation system?
6. What are the (dis)advantages of the 4-channel architecture compared to other control architectures in the context of teleoperation?

Then, the current integration of the force sensor will be improved upon by answering:

7. How can different types of force sensors be used to measure the force in a teleoperation system and which type provides the best performance for a cost-effective DIY teleoperation system?
8. In what ways does the quality of the force measurement have an effect on the transparency and stability of a teleoperation system?
9. What are the shortcomings of the current implementation of the load-cell as a force sensor in the DIY teleoperation system?
10. How can the current implementation of the load-cell be improved to optimise the performance in a cost-effective way?

1.7 Methodology

First of all, relevant literature will be reviewed to gain fundamental knowledge of control architectures and force sensors in teleoperation systems. For this, a block diagram representing a general bilateral teleoperation system [6] will be used and related to the explanations of the architectures provided in Avatars.Report. Furthermore, the current applications and shortcomings of (cost-effective) force sensors in teleoperation systems will be discussed.

Then, this knowledge will be applied to answer some of the underlying research questions and come up with experiments and hypotheses to answer the others, while keeping in mind the limitations of the DIY kit. When these experiments have been conducted, integration of the architectures and force sensor in the DIY kit can start, and experiments to validate this integration should be considered.

Due to hardware limitations of the setup, specifically the load-cell amplifier's sampling rate, and time constraints, the implementation of the 4-channel architecture and Processing interface has not been completed in this thesis. Nonetheless, the required background information and envisioned implementations will be discussed.

After the integration, the performance of the position-measured force architecture and the force sensor will be validated through these experiments. Then, the results will be discussed to determine if the assignment has been completed successfully.

To conclude, the findings will be summarised, the main research questions will be answered and future research directions will be suggested.

1.8 Report outline

In Chapter 2, essential background information for the successful implementation of the control architectures and enhancement of the force measurement is provided and reviewed. In Chapter 3, the prior work to the DIY kit and its limitations are reviewed. Furthermore, several solutions for integrating the control architectures and improving the force measurement are discussed and validated through experiments, considering the identified limitations of the DIY kit. At the end of this chapter, the pseudocode and data-flow diagrams derived from the optimal solutions are presented. In Chapter 4, these solutions (except for the 4-channel architecture and Processing interfaces, as discussed previously) are fully implemented in the DIY kit. Again, experiments are conducted to find the optimal implementation and evaluate its performance. In Chapter 5, the overall performance of the solution to the assignment is evaluated and discussed through several experiments. Finally, Chapter 6 concludes this report by reflecting on the the assignment's completion with the achieved results, and offers several recommendations for further improving the DIY kit's performance.

2 Background

Before formulating any solutions, each part of the assignment should be properly understood. Therefore, this chapter addresses the majority of the underlying research questions outlined in Section 1.6 through reviewing relevant literature and relating this with the explanations provided in the Avatars.Report educational videos.

2.1 Control architectures

2.1.1 Overview of control architectures in teleoperation

In teleoperation, the purpose of the control architecture is to make the forces and velocities of the robot on the human side and robot in the task environment equal, thereby simulating a direct connection between the operator and task. This can be achieved through controlling the positions/velocities and/or forces of the robots with feedback or feed-forward control.

There are many different control architectures, each with different use-cases and complexity levels. In teleoperation, the most common types of control architectures are 2-channel, 3-channel and 4-channel architectures, named after the number of communication channels between the two robots used to communicate the robots' forces and velocities.

First of all, it is useful to already take a look at the general bilateral teleoperation control architecture (Figure 2.1) before diving into these different types of architectures.

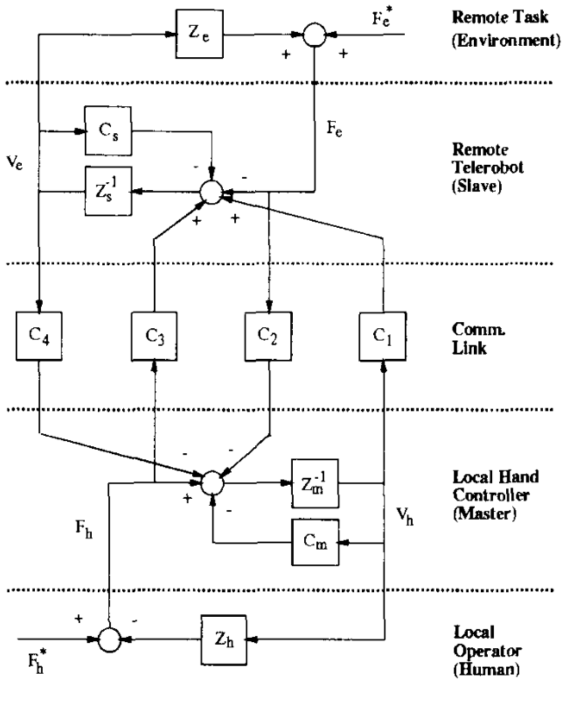


Figure 2.1: General bilateral teleoperation system [6].

This block diagram has only two inputs: the external force that is exerted on the "master" robot by the operator's hand and the external force that is exerted on the "slave" robot by the remote task. Notably, there are no outputs in this diagram; the operator's hand and remote task are included in the diagram as impedances, which convert velocities to forces: $F = Z \cdot V$. However, these impedances are constantly changing depending on the human operator and task environment, making it practically impossible to accurately represent them in a model [6]. To keep with the explanations provided in the Avatars.Report videos, the hand and remote task impedances will be excluded from the architecture diagrams in this thesis. Instead, the input forces for the architecture diagrams will come from the power ports that interact with the operator's hand and the task, explained in Section 2.1.2.

Between the two robots, there are four available communication channels:

- C_1 communicates the "master" robot's velocity to the "slave" robot
- C_2 communicates the "slave" robot's force to the "master" robot
- C_3 communicates the "master" robot's force to the "slave" robot
- C_4 communicates the "slave" robot's velocity to the "master" robot

Furthermore, there are two local (position) controllers: the "slave" controller C_s and the "master" controller C_m. Also, the forces and velocities of the robots are related by their impedances Z_m and Z_s. This will be further explained in Section 2.1.2 as well.

Although not used very often in modern teleoperation systems, 2-channel architectures can provide good transparency with only 2 communication channels, especially in a system that is able to switch between architectures depending on the task environment [11]. The most commonly used two-channel architectures are position-position which uses communication channels C_1 and C_4, and position-force which uses communication channels C_1 and C_2 of Figure 2.1. Evenmore, there is a variation of the position-force architecture which measures the (interaction) force on the robot performing the task instead of computing it in the controller, improving its transparency in case of time delays (Section 2.1.3).

Most often, variations of the general 4-channel architecture (Figure 2.1) are used in modern teleoperation systems. The general architecture can achieve a very high transparency, but requires the models of both robots to be very accurate. Also, the stability of the general 4-channel architecture is not significantly better than that of the 2-channel architectures [6]. Because of these limitations, modified variants of the 4-channel architecture have been designed which are much more suited for implementation in real teleoperation systems.

To maintain its performance in case of uncertain robot models due to varying task/hand impedances or time delays, adaptive control with neural networks can be implemented [12; 13]. More specifically, radial basis function neural networks (RBF NNs) are used because they can estimate most nonlinear continuous functions (the uncertainties in the teleoperation system over time) accurately and relatively quickly [14]. Furthermore, this can be combined with adaptive sliding mode control to handle time-varying delays [12].

To maintain its performance in case of large communication delays, wave-variable control can be implemented in the 4-channel architecture. Now, instead of the power variables (forces and velocities), wave variables are communicated over the four channels, which are a function of these power variables. This passivates the communication channels, guaranteeing stability in case of passive operator and task environments [15]. However, this does typically degrade the transparency compared to the general 4-channel architecture (without time delays) [16].

3-channel architectures can be seen as a middle ground between the 2-channel architectures and the modified 4-channel architectures; they provide better stability than the 2-channel architectures while being less complex than the modified 4-channel architectures. With the use of local force compensators (placed just like C_m and C_s in Figure 2.1), they can even outperform the non-modified 4-channel architecture in terms of transparency with communication delays [17]. There are many types of 3-channel architectures, but the position, force-force (PF-F, using C_2, C_3 and C_4) and position-position, force (P-PF, using C_1, C_3 and C_4) are the most robust. PF-F is most suited for tasks requiring high speed motion without large forces, while P-PF is better suited for tasks requiring high precision [17].

Finally, there are control architectures that differ from the general bilateral architecture (Figure 2.1), like in model-mediated teleoperation where model parameters are communicated instead of forces or velocities (or wave variables). Currently, only half of the originally proposed model-mediated architecture by Hannaford [7] has been succesfully implemented in teleoperation systems: the estimation (and reflection to the operator) of the environment's model/impedance. However, advancements have been made to implement the full model-mediated architecture with bi-directional impedance reflection, such that the operator's model/impedance is also estimated and reflected to the task environment [8].

All of these architectures can be "scaled" with different methods to improve their suitability for certain applications. For example, for industrial teleoperation, the operator needs to perform a task that involves very large forces, requiring that the forces/velocities at the operator's robot are amplified at the robot performing the task. However, the forces exerted on this robot should also be attenuated for the force feedback at the operator's robot to avoid injury. Conversely, in teleoperation systems for microsurgery or micro-assembly the opposite "scaling" is required. In bilateral teleoperation, this is often achieved via power/impedance scaling or position/force scaling and is used in fields such as space explorations, unmanned underwater-/military vehicles, telesurgery and handling of hazardous/explosive materials [9].

For the DIY kit, four control architectures have been chosen: the position-position, position-computed force and position-measured force (2-channel) architectures and the 4-channel architecture. These four architectures form a good theoretical basis for the users' understanding of teleoperation control without overwhelming them with more complex variants designed for non-ideal systems (with time delays and inaccurate models of the robots).

2.1.2 Performance measures of control architectures

In teleoperation, the performance of a control architecture is typically expressed in terms of transparency and stability (or passivity) [6]. Transparency can be seen as the fidelity with which the operator can perceive and control the remote task. Hence, in case of perfect transparency, the operator feels directly connected to the task environment.

Stability is a term that is often used in robotics to indicate the system's robustness. In teleoperation, this typically refers to the robustness in case of communication (time) delays between the two robots: the operator's actions will not be replicated simultaneously in the task environment and the feedback from the task will not be felt immediately by the operator. In other words, the input and output powers of the communication channel are not equal at all times, so time delays can cause energy generation in the otherwise ideally power-continuous channel. In case of large time delays, there can be an uncontrolled increase in energy, leading to instability. This has to be prevented, as any uncontrolled motion ruins the transparency and compromises the safety of everyone near the teleoperation devices. Therefore, if the system does generate energy (it is not passive), it has to be "passivated" to avoid instability in case of large time delays.

Unfortunately, transparency and stability/passivity are conflicting design goals [6] in current teleoperation control architectures. So, they have to be quantified to be able to provide the best trade-off for a certain application.

A typical bilateral teleoperation system can be seen as a two-port model [6], which translates the forces and velocities of the robot that is performing the task to forces and velocities for the robot with which the operator interacts: Figure 2.2. On the right-hand side, F_e, V_e and Z_e describe the task (environment) force, velocity/motion and impedance, which are related by $F_e = Z_e \cdot V_e$. Similarly, on the left-hand side, F_h, V_h and Z_h describe the (operator's) hand's force, velocity/motion and impedance. Z_t describes the impedance that is transmitted (and felt) by the operator.

For perfect transparency, the operator should "feel" the exact task impedance, resulting in Lawrence's transparency condition: $Z_t = Z_e$ [6]. Since $F_h = Z_t \cdot V_h$, this requires equal forces $F_h = F_e$ for equal velocities $V_h = V_e$ and vice versa.

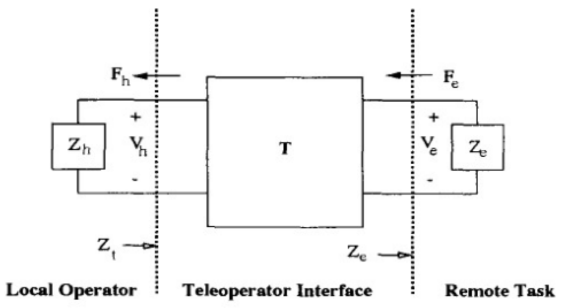


Figure 2.2: General two-port model of a bilateral teleoperation system [6].

However, this transparency condition is not (yet) applicable in the models of the teleoperation system explained in the Avatars.Report videos, where there are three power ports at both robots: the interaction port, controller port and dynamics port. These are indicated in Figure 2.3 as $P_{h,i}$, $P_{h,c}$ and $P_{h,d}$ for the robot on the human side and $P_{r,i}$, $P_{r,c}$ and $P_{r,d}$ for the robot performing the task respectively, to keep with the notation used in Avatars.Report [18]. Each of these power ports represent a co-located[1] force F and velocity $\dot{x}$, the product of which forms the power.

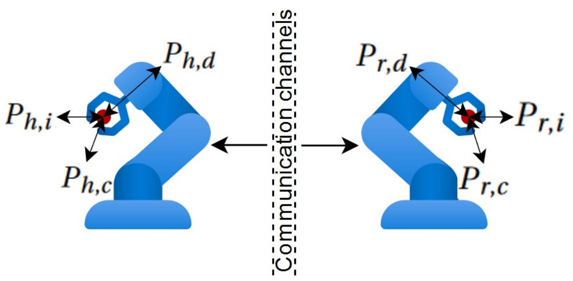


Figure 2.3: Teleoperation system model based on power ports, used in the Avatars.Report videos.

By definition, all three power ports are on the same rigid body or "rigidly connected" (the red dots in Figure 2.3). Since there is only 1 Degree of Freedom (DOF) per robot in the DIY kit, these power ports are one-dimensional. Because they are rigidly connected, the power ports' velocities are equal and their forces are summed at the rigid connection for each robot:

For the robot on the human side:

$$\dot{x}_{h,i} = \dot{x}_{h,c} = \dot{x}_{h,d} \tag{2.1}$$

$$F_{h,i} = F_{h,c} + F_{h,d} \tag{2.2}$$

For the robot in the task environment:

$$\dot{x}_{r,i} = \dot{x}_{r,c} = \dot{x}_{r,d} \tag{2.3}$$

$$F_{r,c} = F_{r,i} + F_{r,d} \tag{2.4}$$

Perfect transparency implies that the operator feels directly connected to the task, so the interaction force and velocity of the robot performing the task should be exactly the same as the interaction force and velocity of the robot with which the operator interacts: equations 2.5 and 2.6. This corresponds with Lawrence's transparency condition and two-port model (Figure 2.2).

For perfect transparency:

$$\dot{x}_{h,i} = \dot{x}_{r,i} \tag{2.5}$$

$$F_{h,i} = F_{r,i} \tag{2.6}$$

For a good energetic connection:

$$\dot{x}_{h,c} = \dot{x}_{r,c} \tag{2.7}$$

$$F_{h,c} = F_{r,c} \tag{2.8}$$

As mentioned before, the controller of the teleoperation system should translate the forces and velocities of the robot on the human side to the forces and velocities for the robot in the task

[1]The velocity and force are at the same location, in the same orientation, and related to the same effect, making them power-conjugated [18].

environment. As follows from equations 2.1 and 2.3, the velocities of the controller power ports at both robots should be equal (Equation 2.7) to make the interaction velocities of both robots equal ($\dot{x}_{h,i} = \dot{x}_{r,i}$). Moreover, the forces of the controller power ports at both robots should be equal (Equation 2.8). When these two requirements are met, there is a good "energetic connection" between the robots. However, the required equal forces $F_{h,c} = F_{r,c}$ in the controller power ports pose a problem for achieving perfecting transparency (using equations 2.2 and 2.4):

$$F_{h,i} = F_{h,c} + F_{h,d} = F_{r,c} + F_{h,d} = F_{r,d} + F_{r,i} + F_{h,d} \neq F_{r,i} \tag{2.9}$$

So, perfect transparency ($F_{h,i} = F_{r,i}$) and a good energetic connection can not be achieved unless the dynamics of both robots cancel each other, which is practically impossible.

To quantify the stability of a system, phase and gain margins are typically used in robotics. When the gain of a system is larger than unity, it will start oscillating and diverging with positive feedback, so negative feedback is often used to prevent oscillation and ensure stability. However, when the phase shift is 180 degrees, this negative feedback becomes positive feedback, causing instability. The gain margin therefore tells how far below unity the gain of the closed loop (i.e. with feedback) system is, and the phase margin indicates how far below 180 degrees the system's phase shift is. Therefore, if the gain or phase margin is negative, the system becomes unstable. Although the gain margin should be confirmed to be below unity, the phase margin is the most important stability margin since the main cause of instability in teleoperation is the phase shift due to communication delays. The phase and gain margins can be retrieved from bode plots, but Nyquist plots are used more often because they show the phase and gain margin in a single plot. Also the number of counterclockwise encirclements of the -1 point on the real axis can tell if the closed-loop system is stable [19].

Next to these mathematical performance measures, there is another important performance measure, namely the experiences and sense of embodiment of the operator that uses the teleoperation system. The sense of embodiment can be divided into three categories: the sense of ownership, agency and self-location [8].

The sense of ownership indicates to what extent the operator can attribute the actions of the remote robot/avatar to their own, while the sense of agency indicates how much control the operator feels they have over the remote robot. Finally, the sense of self-location specifies the operator's awareness of the robot's place in the remote environment [8].

Evenmore, the operator's task performance and experiences of using the device (e.g. comfort and usability) are useful to record, although they do depend on the aforementioned "senses".

2.1.3 Position-measured force architecture

There are two commonly used types of the position-force control architecture, which are both present in the DIY kit: position-*computed* force control and position-*measured* force control. While the position-computed force architecture has already been implemented in the DIY kit, it is useful to understand that architecture first to see the benefits of the position-measured force architecture. First of all, some assumptions will be made to obtain a basic understanding of both types of the position-force control architecture. Then, the position-measured force architecture will be further explained with the general bilateral teleoperation system of Figure 2.1.

Most systems can be roughly approximated by a mass, so can a robotic arm: one can apply a force to the arm, which will result in a velocity of the arm. However, one can not directly "apply" a velocity to the mass, so the other condition for an energetic connection (Equation 2.7) can not be met without some sort of controller. Assuming that the velocities of both controller power ports are equal if the positions are equal ($\dot{x}_{r,c} = \dot{x}_{h,c}$ if $x_{r,c} = x_{h,c}$), the controller can be modelled as an ideal spring, which controls the mass' velocity with the applied force: Figure 2.4 [20].

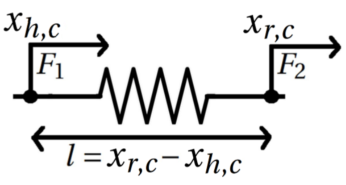


Figure 2.4: The controller as a spring between the controller power ports.

The goal of the controller is to bring the position of the controller power port on the human side to the position of the controller power port on the task side (and vice versa) as fast as possible. A spring also wants to return to its rest length l_0 as fast as possible, but this is limited by the spring constant K: $F_1 = K \cdot (l - l_0) = -F_2$. By setting $l_0 = 0$ for the controller and replacing F_1 and F_2 by $F_{h,c}$ and $-F_{r,c}$ (they act in opposite directions) respectively, Equation 2.10 is obtained for the position-computed force architecture, resulting in the architecture diagram of Figure 2.5.

$$F_{h,c} = F_{r,c} = K \cdot l = K \cdot (x_{r,c} - x_{h,c}) \tag{2.10}$$

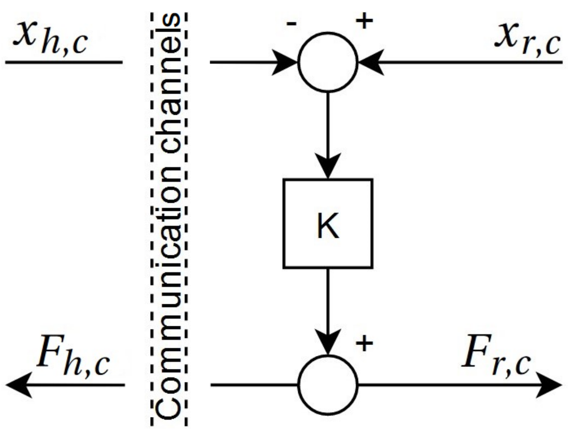


Figure 2.5: Position-computed force control architecture.

This architecture has an inherent problem: for a rigid connection, where the controller power ports can not move with respect to eachother, the spring constant needs to be infinite, which is practically impossible. Furthermore, in case of time delays in the communication channel, energy is generated in the system. When K is increased for a more rigid connection, this generated energy in case of time delays increases as well; a larger spring constant increases the energy in the system, so when a delay occurs in the communication channel, the energy that is generated during this delay is higher. Therefore, the spring constant is limited by the required stability, meaning $x_{h,c}$ will not always be equal to $x_{r,c}$. With the previously made assumption, this also means that $\dot{x}_{h,c}$ will not always be equal to $\dot{x}_{r,c}$, so a good energetic connection (Equation 2.7) is not achieved. Also, regarding transparency, both robots' dynamics still play a role: from Figure 2.5 and Equation 2.4, $F_{h,c} = F_{r,c} = F_{r,i} + F_{r,d}$, which in combination with Equation 2.2 results in $F_{h,i} = F_{h,c} + F_{h,d} = F_{r,i} + F_{r,d} + F_{h,d}$. As mentioned before, perfect transparency requires $F_{h,i} = F_{r,i}$, which is not the case for this architecture due to the robots' dynamics $F_{r,d}$ and $F_{h,d}$.

The position-measured force architecture attempts to solve this problem by measuring the interaction force at the robot in the task environment and using this as the force for the controller power port at the human side directly, as seen in Figure 2.6 [20]. By doing this, the dynamics of the robot performing the task are completely "masked", improving transparency [21]; now, $F_{h,c} = F_{r,i}$, so Equation 2.9 becomes $F_{h,i} = F_{h,c} + F_{h,d} = F_{r,i} + F_{h,d}$. While $F_{h,i}$ is still not equal to

$F_{r,i}$, it is a lot closer since the dynamics of the robot in the task environment $F_{r,d}$ are removed from the equation, so the system has a higher transparency.

Another benefit of this architecture is the removal of the spring output to the communication channel. This makes the spring local (in the task environment) since its output is no longer transmitted over the communication channel. Consequently, the aforementioned energy generation due to time delays is no longer dependent on the spring constant, so the spring constant can be higher than for the position-computed force architecture. This higher spring constant (K) improves the transparency as positions of the controller power ports on both robots ($x_{h,c}$ and $x_{r,c}$) are matched faster (see Equation 2.10), making the connection more rigid. Also, using the same assumption as before, this means that $\dot{x}_{h,c}$ and $\dot{x}_{r,c}$ are (close to) equal more often, improving the energetic connection (Equation 2.7).

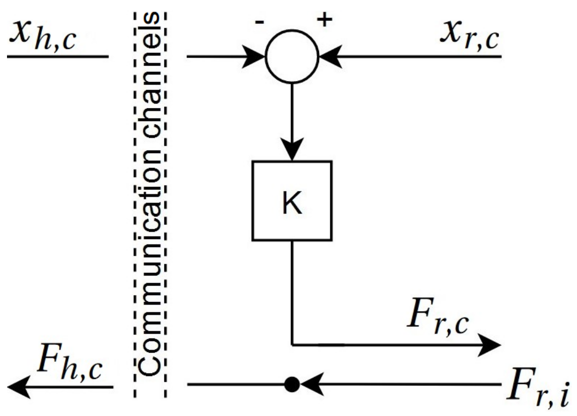


Figure 2.6: Position-measured force control architecture.

Relating this to the general bilateral teleoperation architecture of Figure 2.1, only communication channels C_1 and C_2 are used. By zooming in on the middle three parts of the general architecture (thus excluding the external hand and environment impedances), the position-computed force architecture can be replicated by setting $C_s = K$ and $C_m = 0$. The impedances Z_m and Z_s representing the robots' dynamics are not present in Figure 2.5 since this architecture only includes the controller (between the robots' controller power ports). In the general architecture these impedances should be seen as $Z_{h,d}$ and $Z_{r,d}$, relating the forces and velocities of the robots' dynamics as: $\dot{x}_{h,d} = Z_{h,d}^{-1} F_{h,d}$ and $\dot{x}_{r,d} = Z_{r,d}^{-1} F_{r,d}$. This will be explained in more detail in Section 2.1.4 with Figure 2.7. The relation with the position-measured force architecture is very similar, but here the communication channel C_2 uses the force (measured) after the environment impedance Z_e ($F_{r,i}$ in Figure 2.7), thereby deviating from the general bilateral teleoperation architecture.

While this control architecture is not widely used in teleoperation systems, it is used in some due to its accurate force feedback when the "slave" robot is in contact with something in the environment. For example, it has been used in a cable-driven industrial teleoperation robotic arm meant to reduce the friction of a traditional gear-driven robotic arm [22]. More often, this control architecture is used in combination with the position-position architecture; since the position-position architecture provides a lower tracking error in free space motion, some teleoperation systems switch between the position-position architecture for free space motion and the position-measured force architecture for constrained motion [22; 11].

Comparison with other control architectures

As mentioned before, the position-measured force architecture has two main benefits over the position-computed force architecture: higher transparency due to the masked robot dynamics and a better energetic connection due to the independence of the system's energy generation on time delays. However, when comparing it to the other common 2-channel architecture, position-position control, choosing one over the other is more challenging. The aforementioned better force feedback for constrained motion [22] is certainly a benefit in most teleoperation systems, but the position-position architecture generally provides a lower tracking error in free space motion [22; 23] and higher phase (stability) margin [6]. Nonetheless, if the position-position control is too aggressive in free space motion (i.e. very fast tracking), the operator can experience a "sluggish" feel of the task at the "master" robot due to a large effective inertia, in which case the position-measured force architecture would provide better force feedback [6].

Compared with 3-channel architectures, the transparency of (a combination of) the position-measured force and position-position architectures can be similar, but the stability is typically worse [11].

The comparison between the position-measured force and 4-channel architectures is made in Section 2.1.4.

2.1.4 4-channel architecture

As stated previously, there are many different variants of the 4-channel architecture that are all tailored for different applications requiring different levels of transparency and stability. However, most of them stem from the general 4-channel architecture proposed by D.A. Lawrence [6], using all 4 communication channels of Figure 2.1.

When omitting the local position controllers C_m and C_s (mainly used for damping in [6]) and external forces F_h^* and F_e^*, the task and operator environments (i.e. the robots and the external impedances) are described by the block diagrams of Figure 2.7, in which the intermediate forces and velocities in the terminology from Section 2.1.2 have been added in blue. From these block diagrams, one can obtain:

For the robot on the human side:

$$\dot{x}_{h,i} = \dot{x}_{h,c} = \dot{x}_{h,d} \tag{2.11}$$

$$F_{h,d} = -F_{h,c} - F_{h,i} \tag{2.12}$$

For the robot in the task environment:

$$\dot{x}_{r,i} = \dot{x}_{r,c} = \dot{x}_{r,d} \tag{2.13}$$

$$F_{r,d} = F_{r,c} - F_{r,i} \tag{2.14}$$

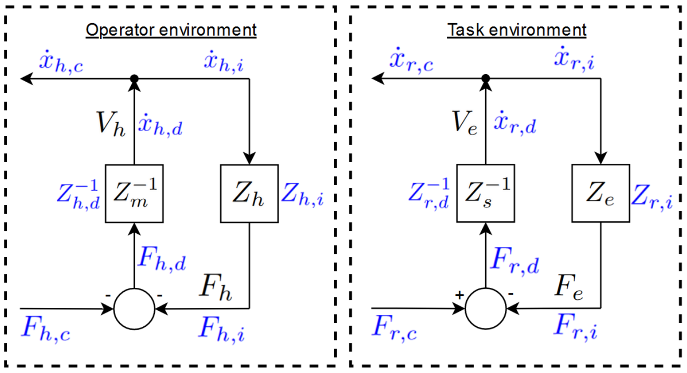


Figure 2.7: Block diagrams showing the robots' dynamics including the operator's and task environment's impedances, obtained from Figure 2.1 by omitting the local position controllers and external forces.

Equations 2.11, 2.13 and 2.14 match the ones based on the Avatars.Report model (equations 2.1, 2.3 and 2.4), but Equation 2.12 does not. This is caused by omitting (i.e. setting to 0) F_h^* in Figure 2.1, which makes $F_h = F_{h,i}$ negative before entering the summation block where it gets added to $-F_{h,c}$, thereby effectively subtracting $F_{h,i}$ from $-F_{h,c}$ instead of adding them to create $F_{h,d}$ like in Equation 2.2. If the external forces F_h^* and F_e^* are not omitted, Figure 2.7 results in Figure 2.8. Here, the sign change of $F_{h,i}$ when F_h^* is equal to zero can be seen clearly.

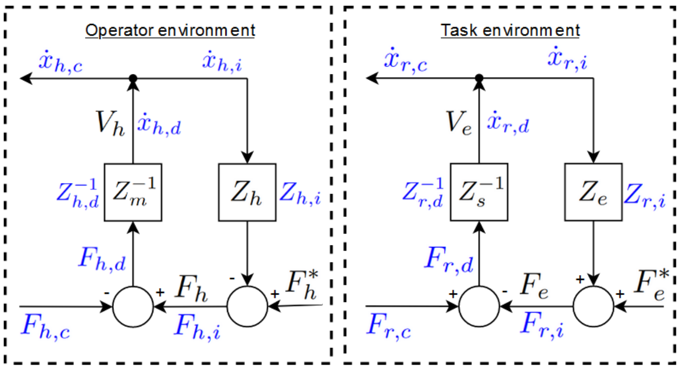


Figure 2.8: Block diagrams showing the robots' dynamics including the operator's and task environment's impedances, obtained from Figure 2.1 by omitting the local position controllers.

When looking at the bond graphs of the teleoperation system (Figure 2.9), it can be concluded that equations 2.1 to 2.4 are indeed derived without considering the (external) impedances Z_h and Z_e. In a bond graph, a 1-junction presents a connections where the flows (in this case velocities) are equal and the sum of efforts (in this case forces) is zero. In a 0-junction, the opposite holds: the sum of flows is zero and the efforts are equal. The arrows indicate the direction of the powers throughout the system [24]. Equations 2.1 and 2.2 are modelled in the red box, while equations 2.3 and 2.4 are modelled in the green box. Because a power always has to enter an impedance (which is not a power source) and the human interaction power $F_{h,i}$ is an input on the human side, the directions of $F_{h,i}$ and power into Z_h are opposite. Since the sum of efforts (forces) has to be zero at this 1-junction, the force at Z_h has to be $-F_{h,i}$, which explains the aforementioned inconsistency between the Avatars.Report model and general bilateral teleoperation architecture. As the bond graph on the task side outputs the (robot) interaction force $F_{r,i}$, the force at Z_e is equal to $F_{r,i}$. Therefore, this inconsistency does not affect the task side.

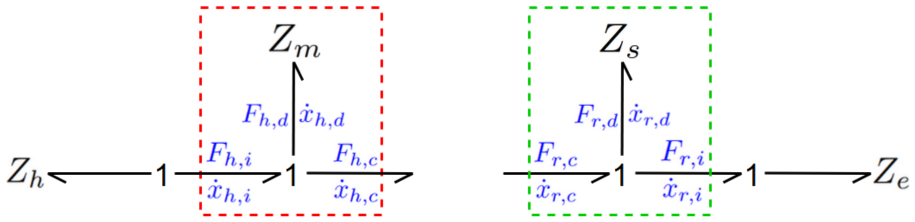


Figure 2.9: Bond graphs of the robots' dynamics with the operator's and task environment's impedances.

However, as mentioned before, the impedances of the operator's hand and task environment are difficult or impossible to determine, so they are excluded from the 4-channel architecture diagram. Hence, the diagram starts and ends at the robot interaction power ports, equivalent to zooming in on the middle 3 parts (the controllers and communication channels) of Figure 2.1.

The 4-channel architecture uses the principle of feed-forward control to match the forces and velocities of the interaction power ports of both robots. Feed-forward control uses no (feedback) loops, so it can be seen as an open-loop gain which depends on the variables and parameters present in its transfer function.

By placing feed-forward controllers (as impedance) before each force/velocity is communicated, the 4-channel architecture block diagram of Figure 2.10 is obtained. Here Z_{C_1} is equal to C_1 in Figure 2.1 and similarly $Z_{C_2} = C_2$, $Z_{C_3} = C_3$, $Z_{C_4} = C_4$.

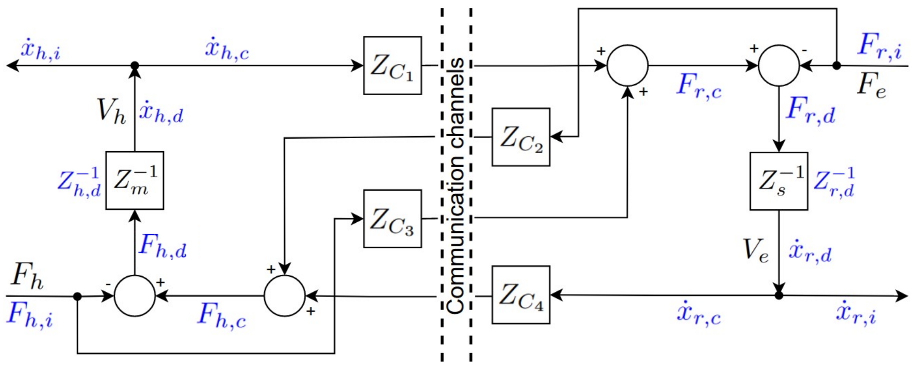


Figure 2.10: 4-channel architecture of Figure 2.1 adapted to correspond with the Avatars.Report teleoperation models.

As per Equation 2.5, for perfect transparency $\dot{x}_{h,i}$ should be equal to $\dot{x}_{r,i}$. From Figure 2.10, it can be derived that (keeping in mind that $\dot{x}_{h,i} = \dot{x}_{h,c} = \dot{x}_{h,d}$ and $\dot{x}_{r,i} = \dot{x}_{r,c} = \dot{x}_{r,d}$):

$$\dot{x}_{r,i} = \frac{1}{Z_{r,d}}(F_{r,c} - F_{r,i}) = \frac{1}{Z_{r,d}}(Z_{C_1}\dot{x}_{h,i} + Z_{C_3}F_{h,i} - F_{r,i}) \tag{2.15}$$

If this should be equal to $\dot{x}_{h,i}$, then: $Z_{C_1} = Z_{r,d}$, $Z_{C_3} = 1$ and $F_{h,i} = F_{r,i}$. The last condition is also the other requirement for perfect transparency: Equation 2.6. From Figure 2.10, it can also be derived that:

$$F_{h,i} = F_{h,c} - F_{h,d} = Z_{C_4}\dot{x}_{r,i} + Z_{C_2}F_{r,i} - Z_{h,d}\dot{x}_{h,i} \tag{2.16}$$

If this should satisfy $F_{h,i} = F_{r,i}$, then: $Z_{C_4} = Z_{h,d}$, $Z_{C_2} = 1$ and $\dot{x}_{h,i} = \dot{x}_{r,i}$. Hence, the perfect transparency conditions are both met if one of them is met. For this to be the case, $Z_{r,d}$ and $Z_{h,d}$ should be known since Z_{C_1} should be equal to $Z_{r,d}$ and Z_{C_4} should be equal to $Z_{h,d}$. The dynamics of both robots can be approximated, but they are almost never perfectly accurate, meaning absolutely perfect transparency is unfeasible in physical teleoperation systems using the 4-channel architecture.

Therefore, this general 4-channel architecture is rarely used in real teleoperation systems. Nevertheless, modified variants of the 4-channel architecture are widely used, as discussed in Section 2.1.1.

Comparison with other control architectures

In an ideal scenario, without communication delays and with models/impedances perfectly representing both robots' behaviours, the 4-channel architecture is one of the best architectures to use for optimal transparency. Compared the position-position, position,force-force and modified (wave-variables) 4-channel architectures, the general 4-channel architecture provides the best transparency for both free motion and constrained/contact tasks [25; 26].

Nevertheless, this is not a realistic scenario, which is why alternative (modified) architectures have been designed. When there are communication delays, 3-channel architectures (like position,force-force) and modified 4-channel architectures generally provide a higher transparency than the general 4-channel architecture due to their higher stability margins [25]. This adheres to Lawrence's statement that stability and transparency are conflicting design goals.

While theoretically, 2-channel architectures like position-position and position-(measured or computed) force can achieve high transparencies, they have to use infinitely large control gains or impedances to do so, which is not feasible in physical systems [6]. The 4-channel architecture and its modified variants can achieve very high transparency with physically reasonable control laws [6] (as derived from equations 2.15 and 2.16), which is a major benefit for application in physical teleoperation systems.

Regarding stability, the 4-channel architecture performs better than most 2-channel architectures. This difference is a consequence of not considering the "dynamics of the interconnected system" [6] in the position-position and position-computed force architectures: the teleoperation robots are interconnected in a feedback loop via the communicated forces/velocities (depending on the architecture). However, for the position-measured force architecture, these closed-loop dynamics are circumvented by measuring the interaction force at the task robot directly, which contributes to a higher stability than its computed force counterpart.

2.1.5 PWM and torque control

To control DC motors with digital (micro)controllers, Pulse Width Modulation (PWM) is commonly used because of its efficiency and speed in creating an analog variable voltage at a variable frequency [27]. As the name suggests, this voltage is created by varying the width of an analog pulse, which has a certain fixed amplitude. As seen in Figure 2.11, when one varies this pulse width, the average analog voltage changes proportionally to the pulse width. The width of the pulse is defined as the duty cycle, ranging from 0% to 100% and calculated with respect to a single PWM period. At a duty cycle of 0%, the average analog voltage is 0 V and at a duty cycle of 100%, the average analog voltage is equal to the pulse amplitude. Hence, by varying the duty cycle, any average voltage between 0 V and the pulse amplitude can be created. Because the (micro)controller only has to switch between 0 V and the pulse amplitude using internal semiconductor switches (transistors), the power conversion efficiency is very high. However, a DC motor does not simply use the average voltage of this PWM signal, so the PWM frequency should be as high as possible to ensure smooth motion.

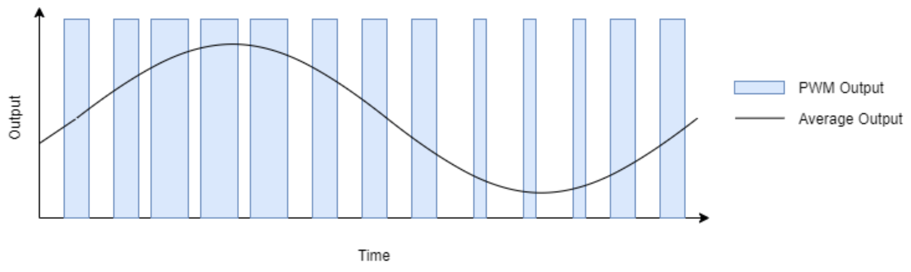


Figure 2.11: Pulse Width Modulation to create any voltage between 0 V and the pulse amplitude [28].

As stated, PWM controls the voltage over the motor, which is not equal to its torque. Since the position-measured force and 4-channel architectures control the robots' forces ($F_{r,c}$ in Figure 2.6, and $F_{r,c}$ and $F_{h,c}$ in Figure 2.10) which originate from the motor's torque, a way of controlling or estimating this torque is required.

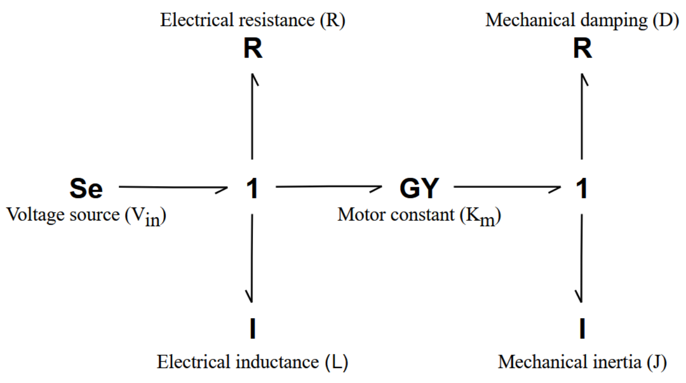


Figure 2.12: Bond graph representation of a DC motor.

A motor can be seen as a gyrator that converts a power in the electrical domain to a power in the mechanical domain, and can be modelled as the bond graph in Figure 2.12, where there is an internal resistance and inductance in the electrical domain and an internal inertia and damper in the mechanical domain. As explained before, 1-junctions indicate that the flows (currents i and angular velocities φ') are equal and the efforts (voltages V and torques T) are summed. The gyrator then converts the flows in one domain to efforts in the other domain and vice versa. This leads to the following constitutive relations in Laplace domain[2] [29]:

Electrical domain:

$$V_m = V_{in} - V_R - V_L \tag{2.17}$$

$$i_m = i_R = i_L = i \tag{2.18}$$

$$i_L = \frac{1}{Ls} V_L \tag{2.19}$$

$$i_R = \frac{V_R}{R} \tag{2.20}$$

Gyrator:

$$V_m = K_m \varphi'_m \tag{2.21}$$

$$T_m = K_m i_m \tag{2.22}$$

Mechanical domain:

$$T_m = T_J + T_D \tag{2.23}$$

$$\varphi'_m = \varphi'_J = \varphi'_D = \varphi' \tag{2.24}$$

$$\varphi'_J = \frac{1}{Js} T_J \tag{2.25}$$

$$\varphi'_D = \frac{T_D}{D} \tag{2.26}$$

These constitutive relations can be combined into one differential equation as follows [29]:

$$V_{in} = i_R R + Ls i_L + K_m \varphi'_m \tag{2.27}$$

$$i = \frac{V_{in} - K_m \varphi'_m}{R + Ls} \tag{2.28}$$

$$Js\varphi'_J + D\varphi'_D = K_m i_m \tag{2.29}$$

$$Js\varphi' + D\varphi' = K_m \frac{V_{in} - K_m \varphi'_m}{R + Ls} \tag{2.30}$$

$$Js\varphi' + D\varphi' + K_m^2 \frac{\frac{1}{R}}{1 + \frac{L}{R}s} \varphi' = K_m V_{in} \frac{\frac{1}{R}}{1 + \frac{L}{R}s} \tag{2.31}$$

$\frac{\frac{1}{R}}{1+\frac{L}{R}s}$ can be simplified as it represents a (transfer function of a) low-pass filter. Using the Fourier transform ($s = j\omega = j2\pi f$), the transfer function in frequency domain is $\frac{\frac{1}{R}}{1+\frac{L}{R}j2\pi f}$. So, for frequencies f much lower than $\frac{R}{2\pi L}$ (the cut-off frequency): $\frac{\frac{1}{R}}{1+\frac{2\pi L}{R}fj} \approx \frac{\frac{1}{R}}{1+0j} = \frac{1}{R}$.

With this simplification, Equation 2.31 becomes:

$$Js\varphi' + (D + \frac{K_m^2}{R})\varphi' = K_m \frac{V_{in}}{R} \tag{2.32}$$

The first part of this equation can be seen as the torque output T of the motor ($\frac{K_m^2}{R}$ is the effect of the electrical resistance in the mechanical domain). Furthermore, V_{in} can be seen as $\frac{\text{duty cycle}}{255} V_{\text{supply}}$ since the maximum duty cycle on an Arduino is 255 and the PWM signal is low-pass filtered to an analog value by the aforementioned $\frac{\frac{1}{R}}{1+\frac{L}{R}s}$. Consequently, the motor's torque output for a certain duty cycle can be approximated with the motor constant K_m and electrical resistance R as per Equation 2.33, where $\frac{\frac{\text{duty cycle}}{255} V_{\text{supply}}}{R}$ is the current through the motor.

$$T = K_m \frac{\frac{\text{duty cycle}}{255} V_{\text{supply}}}{R} \quad [\text{Nm/A}] \tag{2.33}$$

2.2 Force sensing

2.2.1 Overview of (cost-effective) force sensors in teleoperation

There are three main fields where teleoperation is (starting to be) applied: medical applications, industrial applications and social applications [3]. While some applications use quite

[2]To go from time domain to Laplace domain: $\frac{d}{dt} \rightarrow s$ & $\int dt \rightarrow \frac{1}{s}$

expensive force sensors that can measure forces and torques in multiple DOFs, they are often fundamentally similar to cheaper variants measuring forces in one DOF. Therefore, the cheaper force sensors will be discussed first.

A strain gauge is a widely used and cost-effective sensor to measure force or torque [30]. When a force or torque is applied, the resistive wire in the gauge deforms, thereby changing its resistance. Most strain gauges are made of metal wires arranged in a grid and placed between two insulating layers. Nonetheless, strain gauges can also be made of semiconductor material (e.g. silicon) to increase the linearity of their change in resistance with respect to the applied force [30]. However, silicon strain gauges (also called piezoresistive sensors) are much more sensitive to temperature changes compared to metal strain gauges. Both types of strain gauges only exhibit very small changes in their resistance (and thus voltage), so they are often placed in a Wheatstone bridge to get more noticeable changes in voltage when a force is applied [30].

There are many different types of Wheatstone bridge configurations, varying in the amount of active strain gauges (measuring the relevant force) and their placement. These configurations can greatly increase the accuracy of measuring the small changes in the strain gauges' resistances, but do not compensate for the non-linearity of their resistance versus strain curve (Figure 3.15), so load-cells often come calibrated from the factory [31].

In a Wheatstone bridge (Figure 2.13), four strain gauges are placed in a diamond shape and the supply voltage is applied at the top and bottom corners of the diamond. Similarly, the output voltage is measured between the left and right corners of the diamond.

A full Wheatstone bridge, where all four strain gauges are active, will provide the largest output voltage (difference) for a certain deformation of the strain gauges. This output voltage is halved for the half Wheatstone bridge, where two strain gauges are active, and halved again for the quarter Wheatstone bridge, where only one strain gauge is active [31].

Arranging the strain gauges correctly in the bridge is important, since the output voltage is measured between two voltage dividers (Figure 2.13):

$$V_{out} = \frac{R_2}{R_1 + R_2} V_{supply} - \frac{R_3}{R_3 + R_4} V_{supply} \tag{2.34}$$

If $R_1 = R_2 = R_3 = R_4$ when no force is applied, $V_{out} = 0$. So, for the largest possible output voltage, R_1 and R_3 should increase and R_2 and R_4 should decrease when a force is applied (or vice versa), indicated with arrows in Figure 2.13a.

Another cost-effective force sensor is a piezoelectric sensor. When a force is applied on a piezoelectric material, it changes the electric charge on its surface: the piezoelectric effect [33]. Consequently, the voltage over the material changes, with which the deformation and applied force can be calculated. These sensors typically deform much less than strain gauges and have a better dynamic response (varying forces), but can not handle sustained static forces and are sensitive to changes in temperature and humidity [30].

Capacitive sensors are also cost-effective, very sensitive to changes in the applied force and less susceptible to temperature changes. However, since it measures the change in capacitance as the gap between materials increases or decreases, the presence of parasitic capacitances in or around the capacitive sensor are very detrimental to the accuracy of the measurement [30].

Finally, one can use fiber-optic grating (FBG) sensors to measure force, although they are generally a bit more expensive than the aforementioned sensors. The principle behind these sensors will not be discussed as it is quite complicated, but the main idea is that a force causes a change and shift of the wavelength of the light through a fiber-optic cable, which is used to measure the force. These sensors have many advantages: they are not susceptible to electromagnetic interference and temperature changes and they are very accurate and relatively small. Non-

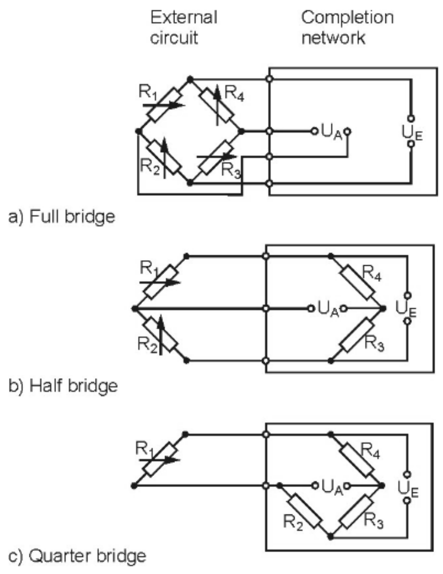


Figure 2.13: Typical Wheatstone bridge configurations of strain gauges [32].

etheless, they can typically not measure forces in as many dimensions as the aforementioned sensors and are slightly more difficult to integrate in teleoperation systems. [30].

In the medical field, the main application for teleoperation is (tele)surgery. It can enable surgeons to operate on a patient from all over the world, and it might even enhance their precision by scaling down their movements in the robot that is performing the surgery. FBG sensors are becoming the norm for telesurgery, since they are biocompatible and sterilizable [30]. Furthermore, their high accuracy and immunity to electromagnetic interference are very beneficial in high-risk teleoperation systems such as telesurgery. Nonetheless, multiple DOF force/torque (F/T) sensors consisting of silicon strain gauges are still widely used, like in the da Vinci surgical robots used for research purposes. As of 2018, more than 6 million procedures had already been performed with 5000 of these da Vinci telesurgery systems [34].

In most industrial applications, like space and underwater explorations, F/T sensors consisting of resistive strain gauges are used [35]. In applications like microassembly and micromanipulation, atomic force microscopes (AFMs) are the norm, which can achieve force resolutions in terms of nanonewtons [36]. While this can not be achieved by using only cost-effective force sensors, AFMs do sometimes use piezoelectric or piezoresistive sensors for part of the measurement process [36]. However, AFMs do have disadvantages like temporary instability at a certain point in the measurement, leading to increased use of more complex MEMS (microelectromechanical systems) as force sensors in micromanipulation [37].

In social applications, there are not many implementations of force sensors yet; the majority of these teleoperation systems currently only use visual feedback, equivalent to a video call.

Notably, sometimes force sensors are completely omitted in teleoperation systems. Instead, the force is estimated via visual deformation of the material [36] or the change in currents driving the robot [38].

2.2.2 Importance of an accurate force measurement in teleoperation

First of all, it should be mentioned that the requirement of an accurate force measurement depends on the control architecture. For the position-position and position-computed force (Equation 2.10) architectures, no force measurements are required since the only inputs are the positions of the controller power ports of both robots. For the position-measured force architecture, one force measurement is required: the force at the interaction power port of the robot in the task environment. Finally, for the 4-channel architecture two force measurements are required: the forces at both robots' interaction power ports as seen in Figure 2.10. There are also modified versions of the 4-channel architecture that use force estimation [39], but this often leads to a higher overall (computational) complexity.

Even in case of sensor and encoder noise, control architectures that use force sensor information (e.g. position-force, position,force-force and 4-channel architectures) generally outperform architectures that do not (e.g. position-position) in terms of dynamic torque tracking and free motion tracking [25]. Evenmore, in case the robots' dynamics are not identical, force sensors are almost a requirement for good kinesthetic coupling [23] since approximating their dynamics will be even more difficult.

There is little literature evaluating the importance of the accuracy of the force measurement (i.e. transparency and stability for varying levels of accuracy) for the discussed control architectures. Hence, this importance is to be hypothesised and evaluated through experiments.

2.2.3 Improving the quality of the force measurement

An accurate force measurement over the whole measurement range is dependent on two main factors: sensor noise and the linearity of the sensor's output with respect to the applied force. Since most force sensors are calibrated (and linearised) in the factory, minimising sensor noise is the most effective way of improving the measurement's quality.

To minimise the sensor noise, a filter can be used. A Wiener filter is commonly chosen since it minimises the Mean Squared Error (MSE) between its output and the desired output, but this filter requires the signal of interest and noise to be stationary processes (the mean and variance do not change over time) [31]. Since the sensor noise can originate from all kinds of varying mechanical and electrical noise sources, like physical disturbances of the DIY setup, mechanical vibrations from nearby electromechanical machines (e.g. a computer fan) or electromagnetic interference [31], the sensor noise is likely not stationary, rendering a Wiener filter and any other filter with static coefficients (e.g. low-pass, high-pass, band-pass or band-stop) inadequate. Furthermore, the desired signal (the applied force on the DIY setup's handle) depends on the actions of the operator or the task environment, which are non-deterministic: their future values can not be predicted with certainty.

There are filters that can automatically adapt their coefficients when the characteristics of the desired signal or noise signal change, such that its performance (e.g. MSE) is always optimised [31]. These filters are called adaptive filters and can use different types of recursive algorithms to update their filter coefficients for the optimal performance, like the Least-Mean-Square (LMS) or the Recursive Least-Squares (RLS) algorithms [40].

The differences between algorithms lie in the performance requirements for the filter. For example, if a low computational complexity (or high power efficiency) is important, the LMS algorithm is a good choice [41]. This algorithm uses a filter with an M amount of taps (= the filter order + 1), of which the filter coefficients $\mathbf{w}(n)$ for the first iteration ($n = 0$) are typically equal to zero. In each iteration, the error $e(n)$ is calculated between the filter output $y(n)$ and the desired output $d(n)$. This error is used together with the convergence factor μ and a vector $\mathbf{x}(n)$ consisting of (delayed) versions of the input signal $x(n)$ to calculate the new filter coefficients that are used in the next iteration: $\mathbf{w}(n+1)$. The convergence factor determines how fast the optimal filter coefficients $\mathbf{w}_{opt}(n)$(achieving the lowest MSE) are reached and should be chosen carefully; a too large convergence factor will overshoot the optimal coefficients, while a too small convergence factor will be needlessly slow to reach the optimal coefficients. In practice,

μ is typically between $\frac{0.01}{MP_x}$ and $\frac{0.1}{MP_x}$, where P_x is the power of the input signal [42].

$$\mathbf{w}(n) = \begin{bmatrix} w_0(n) \\ w_1(n) \\ \vdots \\ w_{M-1}(n) \end{bmatrix} \qquad \mathbf{x}(n) = \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-M+1) \end{bmatrix} \qquad \begin{aligned} y(n) &= \mathbf{w}^T(n)\mathbf{x}(n) \\ e(n) &= d(n) - y(n) \\ \mathbf{w}(n+1) &= \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n) \end{aligned} \tag{2.35}$$

However, for the best performance, the RLS algorithm is a better choice as it adapts itself faster to changing signal/noise characteristics and provides better filtration (attenuation) results [41]. For this algorithm, the filter output $y(n)$ and error $e(n)$ are calculated in the same way (Equation 2.35). However, the filter coefficients are calculated differently: $\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{k}(n)e(n)$ [41]. Since it is a recursive algorithm, it calculates the current filter coefficients with the previous ones, instead of calculating the next coefficients with the current ones. $\mathbf{k}(n)$ is the (Kalman) gain vector and is calculated as Equation 2.36 [43]. λ is the weighting or "forgetting" factor, greater than 0 and smaller than or equal to 1 (typically between 0.95 and 0.995), and defines the influence of previous error samples on the current filter coefficients. Consequently, this determines the rate of convergence to the optimal coefficients and the stability of the filter coefficients [43].

$$\mathbf{k}(n) = \frac{\mathbf{R}^{-1}(n-1)\mathbf{w}(n)}{\lambda + \mathbf{w}^T(n)\mathbf{R}^{-1}(n-1)\mathbf{w}(n)} \qquad \mathbf{R}(n) = \sum_{i=0}^{n} \lambda^{n-i}\mathbf{x}(i)\mathbf{x}^T(i) \tag{2.36}$$

Because of its high computational complexity, there are modified variants of the RLS algorithm with reduced complexity. For example, the Lattice RLS (LRLS) algorithm uses a priori estimation errors and error feedback to reduce the amount of computations [31].

Regardless of the choice of algorithm, the adaptive filter is to be implemented as a noise canceller: Figure 2.14. An adaptive filter tries to replicate a desired signal with its output by filtering its input signal. However, this will only work if the input signal and desired signal are correlated in some way [40], which is taken advantage of by using the adaptive filter as a noise canceller. By giving the filter (an approximation of) the noise as input, it can only replicate the noise in the corrupted signal, as long as the input noise and clean signal are uncorrelated. So theoretically, the error between the filter's output and the corrupted signal is the clean signal, which can be used as the system's output [31]. Nonetheless, there will always be some correlation between the filter's input noise and the clean signal, so the system output will never perfectly replicate the clean signal.

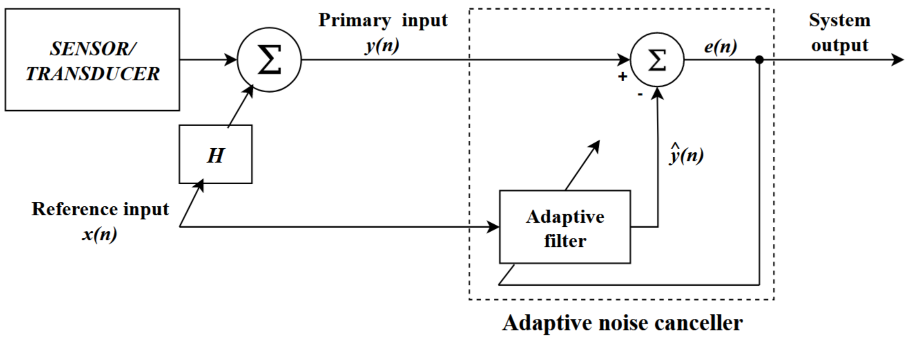


Figure 2.14: Implementation of the adaptive filter as noise canceller, adapted from [31]. $x(n)$ indicates the noise and transfer function H indicates that this noise is uncorrelated with the clean signal but correlated in some way with the corrupted signal $y(n)$.

3 Analysis

As mentioned in Section 1.8, this chapter discusses the limitations of the current iteration of the DIY kit and potential solutions to the assignment with the discussed background information and several (validation) experiments. Section 3.1 considers the DIY kit's limitations and evaluates potential solutions regarding the implementation of the control architectures. Meanwhile, Section 3.2 discusses the shortcomings and limitations of the current force measurement and proposes potential improvements for this aspect of the DIY kit.

As a result, this chapter addresses underlying research questions 2, 3, 5, 7, 9 and 10 of Section 1.6 in the context of the DIY kit.

3.1 Control architectures on Arduino

3.1.1 Limitations of the physical teleoperation setup

Last year, Frank Bosman designed, fabricated and tested the hardware for (one setup) of the DIY kit [44]. This physical setup (excluding the microcontroller) consists of 5 main parts: a motor, motor driver, transmission, position sensor (measuring the angle of the motor shaft) and force sensor. Frank chose an RS-775 (brushed) DC motor [45], which is driven with an Arduino Motor Shield (Rev3) [46], based on an L298 motor driver.

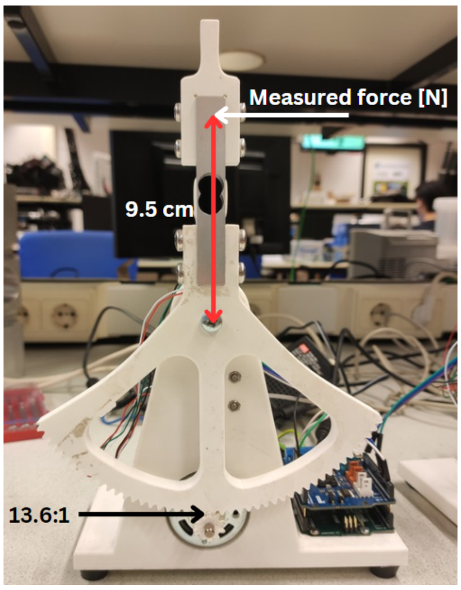


Figure 3.1: Lever length and transmission ratio of the DIY setup to relate the motor torque with the force measured by the load-cell.

Frank stated that this motor can provide a torque of 6.3765 Ncm. The maximum force used by a healthy person in daily tasks is 34.8 ± 1.6 N [47], but without using the thumb this is closer to 15 N. Since achieving high torques is very demanding of the whole physical setup and requires expensive motors, he aimed for a force (F) of 7 N experienced by the human operator at the handle. As seen in Figure 3.1, the transmission gear ratio (g) is 13.6 and the lever length (l) is 9.5 cm. Since the load-cell is calibrated with force applied at the height of its upper screw hole (in this thesis), this lever length does not include the handle. This leads to the required motor torque (T) of Equation 3.1. So, the motor should be able to provide sufficient force feedback.

$$T = \frac{F \cdot l}{g} = \frac{7 \cdot 9.5}{13.6} = 4.89 \; [Ncm] \tag{3.1}$$

Nevertheless, he mentioned that the motor presented cogging behaviour: in permanent magnet motors, there is an interaction between the teeth in the iron core and the poles of the permanent magnet. The iron teeth closest to the magnet's poles are magnetised, resulting in preferential positions of the core with respect to the magnet [48]. This means that there is a small (cogging) torque acting on the core to keep the teeth close to these preferential positions, which is especially noticeable in case of low speeds where the core does not have enough moment of inertia to overcome this cogging torque.

The (dual H-bridge) motor driver was chosen because of its direct compatibility with the Arduino Uno and easy implementation by the user. While it can control the speed and direction of the motors with the applied voltage, it can not control the torque directly as this is related to the current.

For the transmission between the handle (and paddle) and the motor in Figure 1.1, he opted for double helical gears that should be in constant contact with eachother. Hence, this transmission should not present any real limitations in terms of backlash. The transmission ratio was chosen to be as high as possible (which was 13.6, limited by the 3D printer using PLA) to minimise the motor's torque requirement; as per Equation 3.1, the higher the transmission ratio (g), the lower the required motor torque will be.

The AS5048B magnetic position sensor [49] that is used should not present any limitations either; Frank measured a precision of 0.023 degrees and an accuracy of 0.012 degrees. However, the paddle and handle of both setups have to be pointing in the same direction (e.g. straight upwards) when the system is powered on, as the position value is not stored when the system is turned off.

Finally, Frank stated that the force sensor presented a non-linear error and sensor noise. These limitations and their effects on the teleoperation performance will be discussed in Section 3.2.

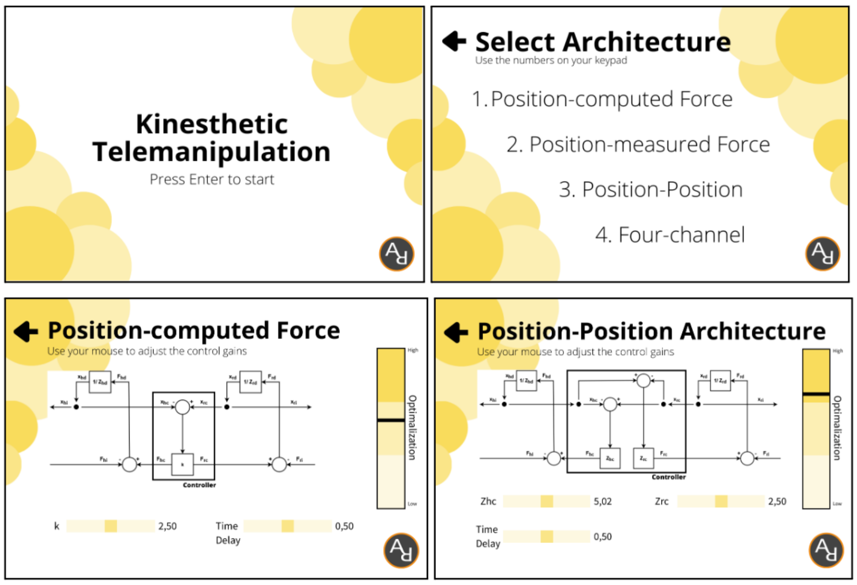
3.1.2 Limitations of the Arduino Uno and user interface

Simultaneous with Frank's assignment, Famke van den Boom designed and tested the software for the DIY setup [50]. For the teleoperation system's microcontroller, she chose an Arduino Uno (Rev3) instead of an ESP32 (both widely used microcontrollers) because the Arduino offers a slightly better user experience in the Arduino and Processing IDEs. This choice is supported by the Arduino's recognised brand and the fact that the wireless communication functionality of the ESP32 is not required for the DIY setup. Consequently, the DIY setup is limited by the microcontroller in 3 aspects: the flash memory (where the code is stored), the clock speed and the amount of I/O (input/output) pins. The Arduino Uno has a flash memory of 32 KB, clock speed of 16 MHz and 14 digital I/O pins including 6 capable of providing a PWM (Pulse Width Modulation, explained in Section 2.1.5) output, and 6 analog input pins [51].

Famke built her own test setup, where both "robots" (motors with flywheels) were controlled with one microcontroller. The encoders are best connected to the interrupt pins of the Arduino to avoid checking the motor angles in each loop iteration, slowing down the code. However, since both encoders need two interrupt pins (with the Encoder.h library [52]) while the Arduino Uno only has two interrupt pins, she had to resort to a different microcontroller: an Arduino Mega with four interrupt pins. With the Arduino Mega, a maximum loop frequency of 1000 Hz could be reached for both the position-computed force and position-position architectures. When the loop frequency is not reached, the Arduino's built-in LED will blink.

The current iteration of the teleoperation setup and revised code uses two of Frank's setups, so two Arduino Uno's with wired communication (through the I2C protocol). Therefore, the number of iteration pins does not pose a problem anymore. She also experienced some inconsistencies in the system's behaviour for higher control values, but she suspects this is caused by a position-dependent friction component on the human side of her test setup, which differs from the one used in this thesis.

Lastly, she designed a user interface and implemented this in the Processing IDE, which communicates with the Arduino via serial communication. As seen in Figure 3.2, the user interface contains sliders to set the control gains, impedances and/or time delay. Furthermore, a feedback bar indicates how optimal the set configuration is with respect to the verified optimal parameters. Only when these parameters are changed, messages with their values are sent to the Arduino. Hence, the addition of the user interface does not present any notable limitations.


Kinesthetic Telemanipulation
Press Enter to start

Select Architecture
Use the numbers on your keypad
1. Position-computed Force
2. Position-measured Force
3. Position-Position
4. Four-channel

Position-computed Force
Use your mouse to adjust the control gains
k 2,50
Time Delay 0,50

Position-Position Architecture
Use your mouse to adjust the control gains
Zhc 5,02
Zrc 2,50
Time Delay 0,50

Figure 3.2: Overview of all user interface screens in Processing [50].

3.1.3 Effects on the implementation & performance of the control architectures

Most of the aforementioned limitations have a relatively minor effect on the performance of the DIY teleoperation setup, considering its cost-effectiveness and goal. Nonetheless, cogging of the motor will reduce the transparency of the setups when it is moving at low speeds, where position and force errors are more likely due to the preferential positions of the motor core and resulting cogging torque: Figure 3.3.

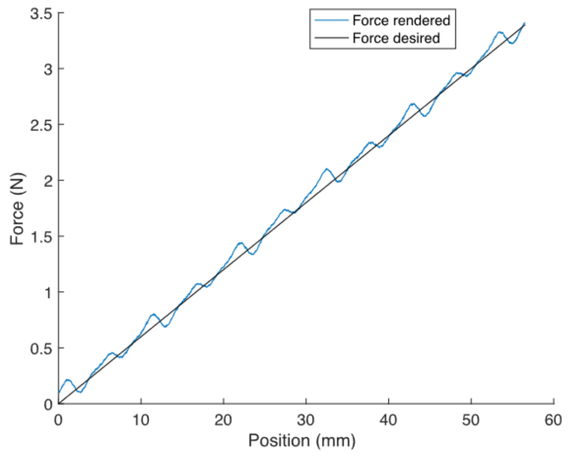


Figure 3.3: Effect of cogging torque while rendering a spring of stiffness k = 0.06 N/mm [48].

Furthermore, the loop frequency should be as high as possible to reduce the latency and optimise the transparency between both setups. If the loop frequency is too low, the user might notice intervals between the updates of the force or position/velocity variables, which ruins the smoothness of the motion of the setups.

While not, the lack of direct torque control might be an issue in the implementation of the control architectures; currently, only the velocity/position of the motor cores (and consequently the handle) in the setups can be controlled, while the position-measured force (and position-computed force) architecture controls the force of the robot on the task side directly.

Fortunately, there are ways to control the torque (and consequently force at the handle) indirectly with the existing setup. If one can measure or approximate the current that is drawn by the motors, the voltage (PWM) of the motors can be adjusted accordingly to achieve a certain current (torque). There is current sensing functionality in the motor shield, so depending on its accuracy a feedback loop can be implemented to control the torque.

Alternatively, the motor torque (and consequently force at the handle) can be estimated based on the PWM duty cycle, motor constant and motor's electrical resistance, as discussed in Section 2.1.5. However, determining the motor constant and electrical resistance is time-consuming and best repeated for each motor to get accurate torque estimates, so if the current sensing functionality of the motor shield suffices, the aforementioned method will be used.

3.1.4 Performance measures for the implementation in the DIY kit

As discussed in Section 2.1.2, teleoperation performance is most often expressed in terms of stability and transparency. Evenmore, it is important to look at the operator's experiences and sense of embodiment when using the teleoperation system and determine whether they completed the task successfully.

In most of the discussed literature (e.g. [11; 17]), the transparency of the teleoperation system is first analysed theoretically using frequency responses, the "hybrid matrix" that represents Lawrence's two-port system Figure 2.2 as per Equation 3.2, and the transparency condition ($Z_t = Z_e$) [6].

$$\begin{bmatrix} F_h(s) \\ V_h(s) \end{bmatrix} = \begin{bmatrix} H_{11}(s) & H_{12}(s) \\ H_{21}(s) & H_{22}(s) \end{bmatrix} \begin{bmatrix} V_e \\ -F_e \end{bmatrix} \tag{3.2}$$

Then, the stability is analysed by evaluating the stability margins with Nyquist plots (Section 2.1.2) of each (sub)loop in the control architecture. With this theoretical analysis, the optimal system parameters and the maximum time delay before instability can be determined.

This theoretical analysis is out of the scope of this thesis, so the transparency and stability of the system will be determined through measurements. In most literature, the transparency and stability are evaluated through experiments by tracking the position and force of both robots under varying time delay. According to the discussed transparency conditions (equations 2.5 and 2.6), the forces and velocities of both robots (i.e. motors in case of the 1 DOF DIY setups) should be the same for perfect transparency.

Hence, the transparency of the DIY setups can be evaluated with the error between the forces and positions of both motors, and the stability/passivity can be evaluated by observing this error for increasing communication (time) delays between the setups.

3.1.5 Implementation in Processing

As mentioned in Section 3.1.2, the current iteration of the Processing interface only sends messages to the Arduino when a parameter (e.g. control gain) is changed. Additionally, the majority of user interface screens have already been created, aside from the screens for position-measured force and 4-channel architectures. Hence, two user interface screens are to be made that match the aesthetic and functionality of the existing Processing user interface screens (Figure 3.2).

However, in the current iteration of the DIY kit with two Arduino Uno's, the Processing user interface has not been implemented yet. Since the setup now uses two seperate microcontrollers instead of one, the variables (e.g. k, $Z_{h,c}$ and $Z_{r,c}$ in Figure 3.2) that can be changed in the user interface, running on one of the microcontrollers, should be communicated between both microcontrollers. Since this communication will increase the workload for both microcontrollers and thereby affect the maximum loop frequency, it should be as minimal as possible.

As for the communication between the Processing user interface and one of the Arduino's, the protocol that Famke created is used: when a variable is changed in the user interface, a message of 6 bytes will be sent in the form of Figure 3.4. The header (a letter) indicates the type of message: "a" indicates architecture selection, "b" indicates time delay selection and the remaining letters are used to indicate selection of other variables in the user interface screens. The data field then indicates the new value or architecture of the header (rounded to three decimals) and the end field contains a semicolon to indicate the end of the message.

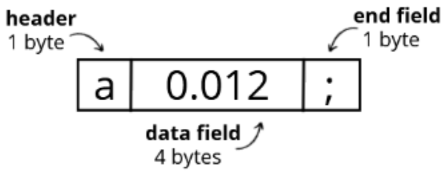


Figure 3.4: Overview of the messages' fields (and their sizes) that are communicated in "string" format between the Arduino and Processing interface [50].

3.1.6 Experiments

First of all, as discussed in Section 3.1.3, the feasibility of directly controlling the motors' torques by measuring their current draw with the Arduino Motor Shields is to be investigated. If this is not feasible, other ways of controlling the force at each setup are to be explored, like estimating the motor's torque for any duty cycle (Section 2.1.5).

The Arduino website [46] states that the motor shield's current sensor is calibrated to output 3.3 V for a (maximum) current draw of 2 A. This is achieved with an internal current sensing resistor and an op-amp for each of the two available motor channels (resistors R1 and R2 and LMV358MMX op-amps IC4A and IC4B in the schematic of the motor shield [53]). These components should not present any non-linear behaviour, as long as the op-amps supply voltage range is not exceeded (0 - 5 V [54]) by the output voltage, which it is not. Finally, there are tutorials that provide accurate measurements (e.g. [55; 56]). Nonetheless, this ought to be verified with experiment 3.1, where the measured currents are compared to the measurements of an external current sensor for various scenarios, listed in Table 3.1.

As an alternative to controlling the torque with the current measured by the motor shield, experiment 3.2 calculates the (motor) constants and the motor's electrical series resistance to estimate the motor's torque for any duty cycle. First, a constant is determined for both setups that approximates the force (f) measured by the load-cell for a current (c) provided to the motors by an external power supply:

$$K_{f,c} = \frac{\text{Force measured by the load-cell [N]}}{\text{Current provided by the power supply [A]}} \tag{3.3}$$

The measurements with this external power supply are performed for voltages up to 2 V since the (specified) current limit of the motor shield (2 A) would then be exceeded by one of the setups. Also, the current limit of the power supply (2.5 A) is reached at slightly higher voltages.

Next, the motors are driven with the motor shields via PWM for different temperatures of the setups to evaluate the temperature-dependence of the motor-constant: "cold" indicates that the setup has not been used for at least the past hour (and is therefore only measured once for each duty cycle), while "warm" indicates the setup has been used with the position-computed force architecture for 5 minutes. "Hot" indicates that the motor was driven with maximum duty cycle for 3 times in a row while holding still the paddle, such that it draws the maximum current the motor shield can supply. From these measurements, a single constant that approximates the force (f) measured by the load-cell for a PWM duty cycle (d) provided by the motor shield(s) is determined for both setups:

$$K_{f,d} = \frac{\text{Force measured by the load-cell [N]}}{\text{PWM duty cycle (0-255) provided by the motor shield}} \tag{3.4}$$

As a ratio, duty cycle is unitless, so $K_{f,d}$ is expressed in newtons (at the load-cell). Finally, this constant $K_{f,d}$ is validated by approximating the constant $K_{f,c}$ with a current measurement at maximum duty cycle and comparing it to the measured $K_{f,c}$. Also, while not used in this thesis, the motor constant is estimated: K_m (Equation 2.33).

Table 3.1: Overview of the experiments conducted to control or estimate the motor's torques.

Experiment	Objective	Tested configurations
3.1	Determining the accuracy of the current sensor in the Arduino motor shield	Motor disconnected from the paddle, able to spin freely. Duty cycles: 0, 50, 100, 150, 200, 255 (0-100 % on Arduino)
		Motor disconnected from the paddle, providing various levels of resistance to the motor shaft's rotation. Excluding the duty cycles where the shaft is not rotating. Duty cycles: 150, 200, 255
3.2	Determining the (motor) constants (K_m, $K_{f,c}$ and $K_{f,d}$) and motors' electrical series resistances	Driving the motors with an external power supply from 0 to 2.0 V. Setup temperature: warm
		Driving the motors with the motor shields. Setup temperature: cold, warm, hot

Experiment 3.1: Determining the accuracy of the current sensor in the Arduino motor shield

Method: Since the motors are controlled with a PWM signal, the current will vary with at least the same frequency as the PWM signal, which is 31372.55 Hz. The external current sensor has to be able to accurately measure and communicate the current at this frequency. Hence, a ACS712 20A (Hall-effect) current sensor module is used which has a bandwidth of 80 kHz, maximum error of 1.5% and an internal resistance of only 1.2 mΩ, so it will not notably reduce the voltage over the motor. This sensor is placed between the motor shield and positive terminal of the motor and connected to an analog input of the Arduino; it communicates the measured current as a voltage with a sensitivity of 100 mV/A, biased around half the supply voltage [57] (so, biased around 2.5 V). The Arduino's Analog-to-Digital Converter (ADC) has a resolution of 10 bits, leading to 2^{10} = 1024 levels to represent voltages between 0 and 5 V. With this knowledge, the current through the motor can be calculated as per Equation 3.5.

$$I_{\text{motor, ACS712}} = \frac{2.5 - V_{\text{ACS712}} \cdot \frac{5.0}{1024}}{0.100} \ [A] \tag{3.5}$$

$$I_{\text{motor, motor shield}} = \frac{V_{\text{motor shield}} \cdot \frac{5.0}{1024}}{1.65} \ [A] \tag{3.6}$$

The current measured by the motor shield is also communicated as a voltage to one of the Arduino's analog inputs. However, this voltage is not biased around 2.5 V and the sensitivity is 1.65 V/A [46], leading to a different equation for the current through the motor: Equation 3.6. These currents are communicated with the computer via serial communication together with the time (in seconds). To receive and save this data, a Python script is used, inspired by [58].

To provide various levels of resistance to the motor shaft's rotation, the motor gear is gripped (by hand) with various levels of force, thereby slowing down the motor shaft and increasing the current draw of the motors.

Results: From Figure 3.5, it can be seen that for low duty cycles (0 - 100), where the motor is not rotating due to its Coulomb friction[1] (among other things, discussed in the next experiment), the ACS712's sensor noise is higher than the current that the motor is drawing. From the bottom-left and bottom-centre plots, it is more difficult to see clear differences because of the rapidly switching currents due to PWM. It is noteworthy that the measured current draw exceeds the specified maximum of 2 A [46]. Finally, the bottom-right plot shows that the current measurements at maximum duty cycle (a constant 12 V supply voltage) when the motor starts spinning (0 to 0.1 s) are very similar.

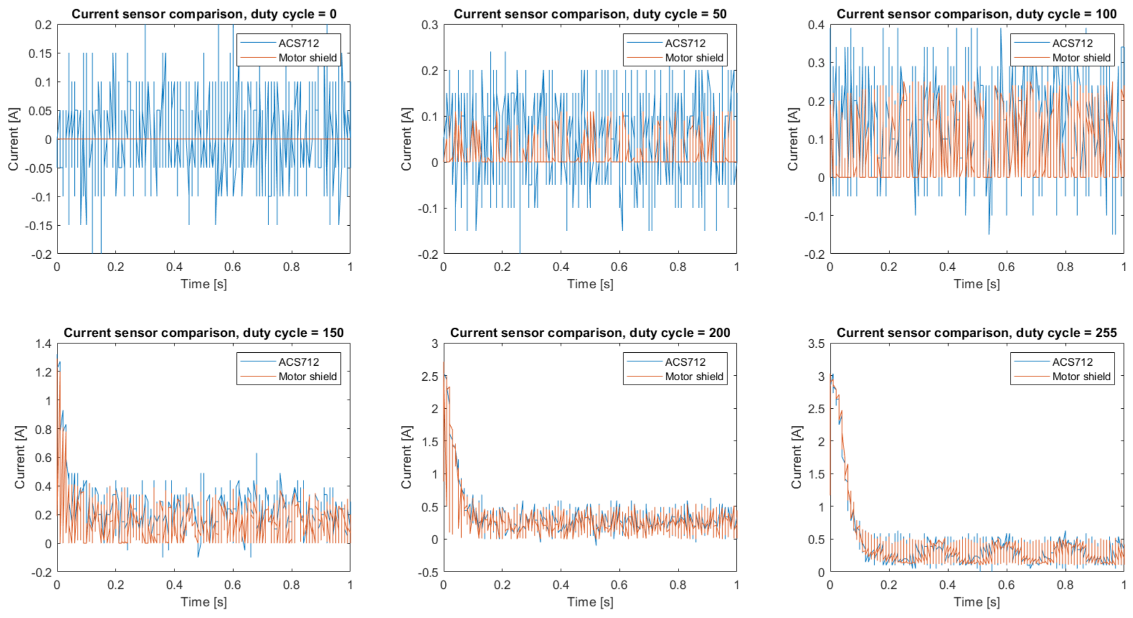


Figure 3.5: Current sensor comparison for varying PWM duty cycles without disturbances.

With Figure 3.6, the main difference between the current sensors can be seen more clearly: especially at large currents, the motor shield's current measurement extends further down to 0 A when the PWM signal is low (as explained in Section 2.1.5, a PWM signal switches between "high" and "low" states to create a certain average DC voltage). At maximum duty cycle (255), this difference is no longer present.

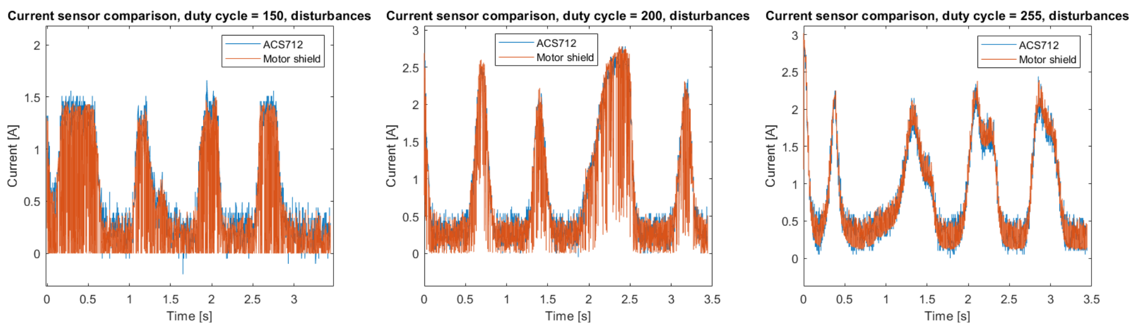


Figure 3.6: Current sensor comparison for varying PWM duty cycles with disturbances.

[1]Friction that is independent of the rotational velocity and present in the direction opposite to the velocity [59].

By low-pass filtering these results using a second order Butterworth filter with a cut-off frequency of 5 Hz (Figure 3.7) and plotting their upper envelopes (Figure 3.8), the difference between the sensors' measurements is much more visible; at duty cycles of 150 and 200 (where the motor shaft is rotating, but not at full speed), the ACS712 current sensor measures much higher currents than the motor shield when the motor shaft's rotation is resisted. Again, this difference is not present at maximum duty cycle. While the upper envelopes of the current sensor's measurements in Figure 3.8 are slightly higher than the upper envelopes of the motor shield's measurements for duty cycles of 150 and 200, the difference is significantly smaller.

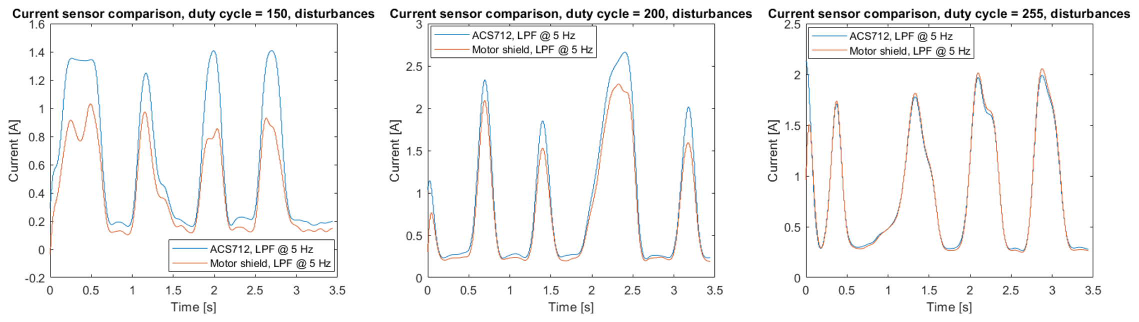


Figure 3.7: Current sensor comparison for varying PWM duty cycles with disturbances, low-pass filtered at 5 Hz with a second order Butterworth filter.

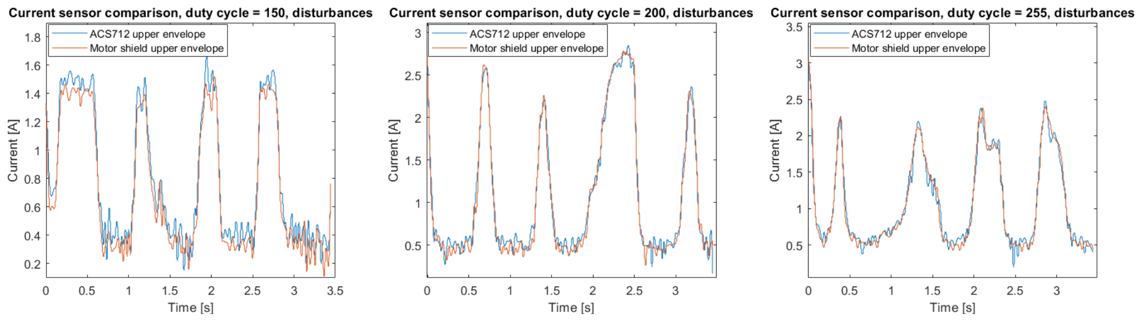


Figure 3.8: Current sensor comparison for varying PWM duty cycles with disturbances, upper envelopes using spline interpolation over maxima separated by at least 10 samples.

Discussion: Since the sensor noise seems to be much higher in the ACS712 current sensor than in the motor shield (for duty cycles of 0 - 100 in Figure 3.5), it may be concluded that the motor shield's current sensor is more accurate at these low duty cycles because it seems to filter out the noise that the ACS712 measures, especially at a duty cycle of 0.

The measured peak current when the shaft's rotation was resisted is around 2.8 A (centre plot of Figure 3.8), which is close to the specified maximum stall current of the motor (3.25 A [45]), considering its rotation was never completely stopped and therefore the true stall current was not reached.

At maximum duty cycle, the sensors' measurements correspond quite well; since a duty cycle of 255 means that the PWM signal is permanently "high", the observed differences at a "low" PWM signal are no longer present. This is supported by the similarities of the upper envelopes, which only show the measured current values when the PWM signal is "high", validating the accuracy of the motor shield's current measurement (at least at maximum duty cycle).

While the upper envelope of the ACS712 is slightly higher than the upper envelope of the motor shield for a duty cycle of 150, it can not be the only cause of the much higher low-pass filtered measurements. This supports the previously made observation that the largest differences occur when the PWM signal is "low", likely originating from one or both of the following causes:

Firstly, the noise that the ACS712 measures results in a certain "noise floor", causing it to measure a slightly higher current than the motor shield; in the top-left plot of Figure 3.5, the mean of this noise (floor) is mostly positive. This is especially noticeable at lower duty cycles, illustrated by the generally slightly higher upper envelope in the leftmost plot of Figure 3.8.

Secondly, because the current measurements need to be saved via serial communication to the computer, the loop frequency (= sampling frequency) of the Arduino could not exceed 690 Hz, while the PWM frequency was roughly 31 kHz. Hence, the measurements of both current sensors were likely taken at slightly different points in time (first the motor shield, then the ACS712), which could mean that one current sensor measured the current for a high PWM signal while the other measured the current for a low PWM signal in the same loop.

Nevertheless, this issue was not resolved at the time of this experiment. To know if the currents match perfectly, an oscilloscope could be used to measure the output voltages of the motor shield and ACS712 and zoom in on one PWM period ($\frac{1}{31372.55}$ [s]). Alternatively, the measurements could be saved locally on the Arduino and exported to the computer after the measurement. However, each of these measurements takes up 37-40 kB while the Arduino's dynamic memory is only 2 KB. In one of the final experiments of this thesis (experiment 4.4 in Section 4.2.3), it was found that using a higher baud rate for the serial communication drastically reduced the loop time. Since the baud rate can be set up to 200 times higher than the baud rate used in this experiment, the sampling frequency could likely exceed the PWM frequency as this is only 45 times higher.

Conclusion: Since there is a noticeable discrepancy between the sensors' measurements when the PWM signal is "low" and the cause of this discrepancy is not confirmed, the use of the motor shield's current sensor can not be justified. Even if the measured current would be sufficiently accurate (which is probable considering the measurements at maximum duty cycle), these measurements should be filtered in real-time to obtain a DC current that is required to control the motor's torque, which inherently delays the signal.

Furthermore, from this experiment it can be concluded that the motor shield is able to provide more current than its maximum rating of 2 A per motor channel, at least for a short time period.

As mentioned at the start of this section, the alternative to controlling the motor's torque by measuring the current is estimating the torque for any PWM duty cycle. This also eliminates the need for a separate torque/current controller, which would (ideally) have operate at an even higher frequency than the PWM frequency. This model-based estimation is explained in Section 2.1.5 and results in Equation 3.7, where T is the motor (output) torque, K_m is the motor constant and R is the motor's electrical (series) resistance.

$$T = K_m \frac{\frac{\text{duty cycle}}{255} V_{\text{supply}}}{R} \quad [\text{Nm/A}] \tag{3.7}$$

Since the force at the load-cell should be estimated instead of the motor torque, the constants that approximate the force at the load-cell for a current and PWM duty cycle (0 - 255) is determined as well: $K_{f,c}$ and $K_{f,d}$ respectively.

Experiment 3.2: Determining the (motor) constants and motors' electrical series resistances

Method: To determine the constants K_m, $K_{f,c}$ and $K_{f,d}$, the force measured by the load-cell is used as it was proven to be very accurate (experiment 3.3 in Section 3.2.5). The setup is placed between (and partly under) heavy objects such that the setup can not move when the motor's torque is applied as a force to one of these objects: Figure 3.9.

Firstly, the motor is driven with an external power supply to measure its current draw for varying (DC) voltages during the force measurement. With these currents and voltages, its electrical

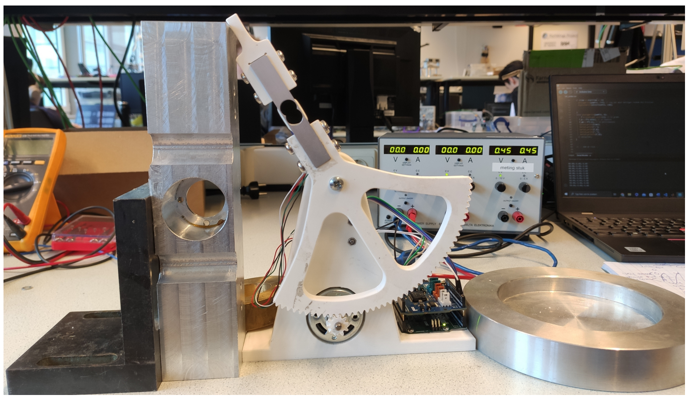


Figure 3.9: Measurement setup used in experiment 3.2 to determine $K_{f,c}$, $K_{f,d}$, K_m and electrical series resistance (ESR).

resistance can be calculated with Ohm's law ($R = \frac{V}{I}$) since the effects of back EMF[2] are absent when the motor's shaft/core is not rotating. During these measurements, the force measured by the load-cell is recorded to obtain $K_{f,c}$. Also, the obtained ESR is compared with the motor's series resistance measured by a (HAMEG HM8118) LCR-meter.

Then, the motor is driven with the Arduino and motor shield via PWM (for the duty cycles listed in Table 3.1) and the force measured by the load-cell is recorded to obtain a constant that approximates this force at the load-cell for a certain duty cycle: $K_{f,d}$. To obtain a rough relation between PWM duty cycle and current, the current through the motor is measured at maximum duty cycle (always "high") with a multimeter; the current at any other non-zero duty cycle is alternating due to PWM and therefore can not be related to the power supply's current.

For consistency, each measurement is conducted three times for both setups and the measured force is recorded once it has stabilised. Furthermore, after each measurement, the load cell's reading is reset to 0 N.

As mentioned in Table 3.1, the measurements with the motor shields are performed for different (setup) temperatures: cold, warm and hot.

Finally, while the setup was fixed in place for these measurements, this will not be the case when it is used for teleoperation. Hence, the maximum duty cycle before the setup's base starts moving or tilting is measured, while holding the handle still and not fixing the setup in place. At least up to this duty cycle, the $K_{f,d}$ should be sufficiently accurate for both setups. So, $K_{f,d}$ (one constant that fits both setups) is determined as the slope of the fitted line between the duty cycle at which a force is first measured and the duty cycle at which the setup's base would start to move or tilt. Since the setups should be closest to the temperature of the "warm" measurements most of the time, the line is fitted to these measurements.

Results:

The results from driving the motors of both setups with the power supply can be found in Figure 3.10 and Figure 3.11. The ESRs of both motors are quite similar, with the motor's ESR of the second setup being slightly lower than the motor's ESR of the first setup. Additionally, the ESRs are consistent with respect to the motor voltages, except for lower voltages (up to 0.4 V). However, the series resistances measured by the LCR-meter were much higher: 2.15 Ω and 2.30 Ω respectively. When measuring a (maxon RE 36 118798) motor with a specified series resistance of 1.11 Ω [61], the LCR-meter measured a series resistance of 3.35 Ω.

[2]An opposing voltage due to the electromotive force induced by the armature conductors (copper wires wound around the rotating part of the motor) moving through the magnetic field created by the magnets in the static part of the motor (stator) [60].

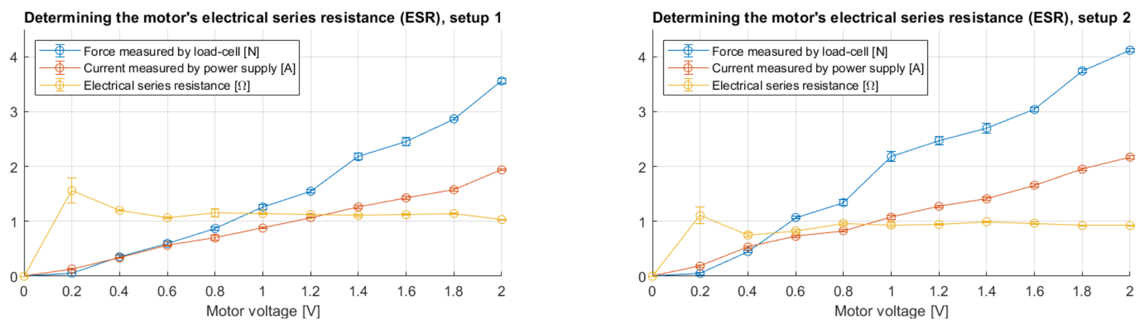


Figure 3.10: Driving the motors with a power supply to calculate their electrical series resistances. The dots indicate the mean of the 3 measurements and the vertical bars indicate the standard deviation.

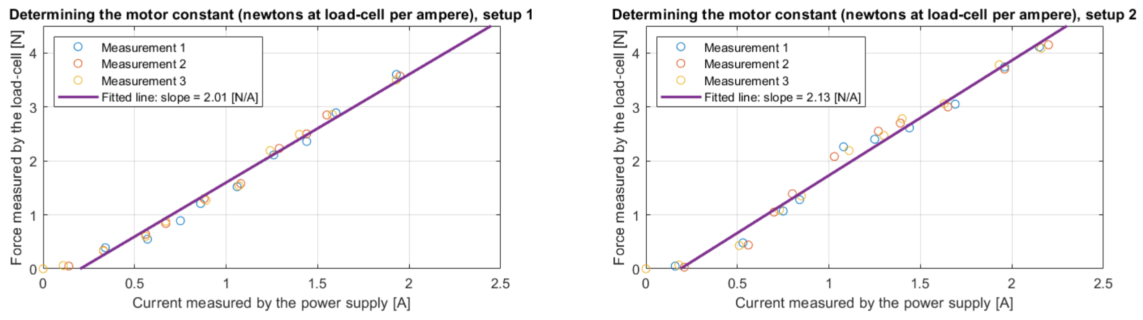


Figure 3.11: Driving the motors with a power supply to determine $K_{f,c}$ for both motors.

The slopes of the fitted lines ($K_{f,c}$) also correspond quite well, with the slope for the second setup being slightly higher than the slope for the first setup. Notably, when comparing the force measurements of both setups, the measurements for the first setup are generally closer to the fitted linear relation with respect to the motor's current.

The results from driving the motors of both setups with the Arduino and motor shield via PWM (for different temperatures) can be found in Figure 3.12. Only the measurements at duty cycles up to 230 are included as the motor shafts started to slip (inside the bottom gear of Figure 1.1) at higher duty cycles. In the final experiments of this thesis, this slipping started to occur at low duty cycles as well, so the gears were reattached to the motor shafts with Loctite 601 glue.

Before this slipping started to occur more frequently, the current at maximum duty cycle was measured to be able to relate these measurements to the power supply measurements. At maximum duty cycle, the current through the motors was 2.57 A for setup 1 and 2.65 A for setup 2, leading to an average 2.61 A.

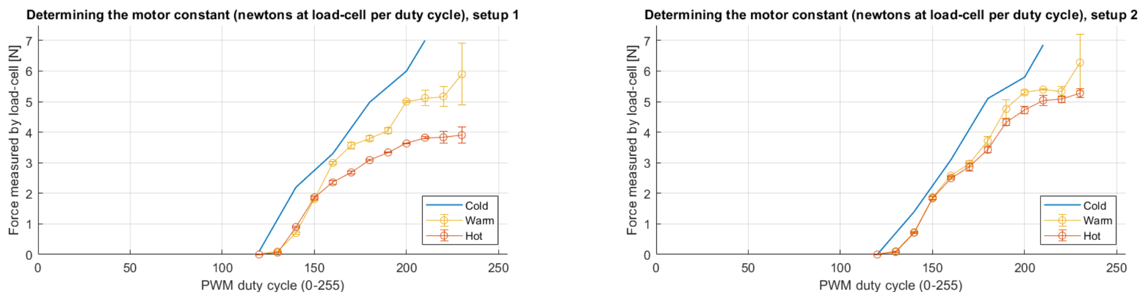


Figure 3.12: Driving the motors with the motor shield via PWM to determine $K_{f,d}$.

The first notable observation is the duty cycle for which the load-cell starts to measure a force: 120 in case of a cold setup and 130 otherwise. This is already near half of the maximum duty cycle, while the load-cell started to measure a force much earlier in the measurements with the power supply: at 0.2 V and 0.2 A of the maximum 2.0 V and ~2.0 A respectively: Figure 3.10. Furthermore, as suspected, the temperature of the motors in both setups clearly affects force at the load-cell, and therefore the torque that the motors provide. Notably, (the measured force in) setup 1 is less affected by increasing temperature than setup 2. Nonetheless, the precision of the 3 measurements at the same duty cycle and temperature is quite high for duty cycles up to 190. At higher duty cycles, this precision decreases rapidly.

The force, measured by the load-cell, at which the setup starts to move or tilt is determined to be roughly 4 N. So, the line representing $K_{f,d}$ is fitted to the "warm" measurements of both setups up to 4 N: Figure 3.13. This fitted line starts at a duty cycle of 125 with a slope of 0.0714 N. Hence, $K_{f,d}$ for both setups is 0.0714 N from a duty cycle of 125 onwards.

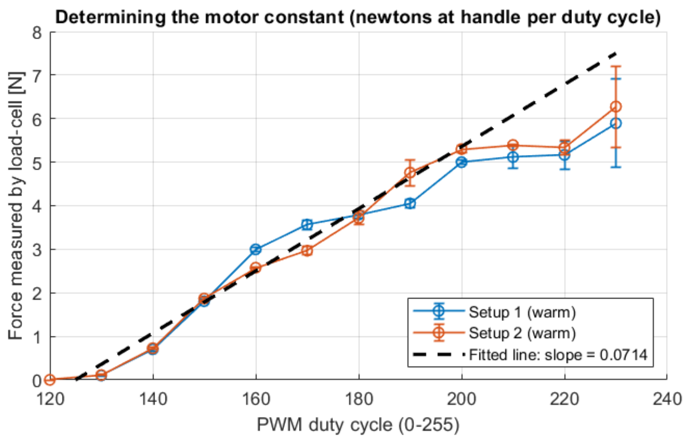


Figure 3.13: Fitting a line to the "warm" measurements of Figure 3.12 up to 4 N to find the optimal $K_{f,d}$.

Assuming a similar current draw at 255 duty cycle and 200 duty cycle (since the measured force remains roughly the same when the setup is warm/hot), a constant ESR up to a duty cycle of 200, and the same current for which a force is first measured (0.2 A: Figure 3.11) for both experiments, this $K_{f,d}$ can be compared with the measured $K_{f,c}$ using the aforementioned measured current draw of 2.61 A at maximum duty cycle. First, the duty cycle range is converted into a current range by assuming a current draw of 0.2 A at 125 duty cycle and 2.61 A at 200 duty cycle, resulting in $\frac{200-125}{2.61-0.2} \approx 31.12\ A^{-1}$. Applying this conversion to the determined $K_{f,d}$ of 0.0714 N results in $K_{f,c} = 0.0714 \cdot 31.12 \approx 2.222$ N/A.

Discussion: At low voltages and currents, the motors have to overcome the Coulomb friction. On top of that, the measurements are relatively less accurate at these low voltages and currents due to the power supply readings being limited to two decimals, which explains the inconsistent ESRs at low motor voltages. Although it may be accurate, the slight difference in the motors' ESRs could be attributed to a difference in temperature, as the ESR of a DC motor typically increases when it heats up [60]. This could not be verified with the LCR-meter, as the measured ESR of a DC motor with a specified resistance was inaccurate.

However, this relatively low ESR could explain the notable difference between the voltage and the duty cycle for which the motor starts to apply torque when driving the motor with the power supply and motor shield respectively. The motor shield aims to regulate the motor voltage from 0 to 12 V (its supply voltage) through PWM, but can only provide 2 A to each motor. Hence, with an ESR of roughly 1 Ω, the motor shield can only regulate the motor voltage up to 2 V. Since the setups start to move and tilt after forces of 4 N, a less powerful but more consistent motor would be better suited for these specific DIY setups.

The accuracy of the power supply's current measurements and the load-cells' force measurements was verified with a (Fluke 179) multimeter and mechanical force meter respectively, so the difference in the measured $K_{f,c}$ and calculated $K_{f,c}$ (from $K_{f,d}$) can only be accurate or attributed the setups' temperatures. As the motor's torque was much more dependent on temperature in setup 1, this does imply that there is a difference between the setups.

Since the motor shields were felt to heat up noticeably more than the motors, the torque seems to be limited by the current of the motor shield at high duty cycles, not the increasing ESR of the motors. Nevertheless, fitting a second-order function to better represent these setups can not be justified, as a motor constant (and consequently $K_{f,c}$ and $K_{f,d}$) is represented by a first order line. Moreover, the motor constant should be representative of both motors, as the DIY setup is intended to be assembled by the users, who should not need to characterise the motors themselves.

So, the $K_{f,d}$ of 0.0714 N should be the best choice for every DIY setup using this hardware, but these measurements ought to be performed for more setups to ensure this is the case. When converted to N/A (with certain assumptions), this $K_{f,c}$ (2.222 newton/ampere) is slightly higher than the measured $K_{f,c}$: Figure 3.11. This difference could be attributed to a slightly warmer motor in the power supply measurements, but it is more likely to originate from the assumption that the current is the same at 200 duty cycle and 255 duty cycle; when the setup is cold/warm, the measured force is generally higher at 230 duty cycle than at 200 duty cycle, implying the current draw is higher as well.

While not used in this thesis, the motor constant K_m can be approximated with $K_{f,c}$ (calculated from $K_{f,d}$) and Equation 3.1:

$$T = \frac{F \cdot l}{g} = \frac{2.222 \cdot 9.5}{13.6} = 1.55\ [\text{Ncm}] \rightarrow \text{Motor constant} = 0.0155\ [\text{Nm/A}] \tag{3.8}$$

If a different motor with a specified motor constant is used in a future iteration of the DIY setup, the $K_{f,c}$ calculated with $K_{f,d}$ can be validated using this conversion.

Conclusion: From these measurements, multiple conclusions can be drawn. The approximate ESR of this type of (RS-775) motor can be taken as the average ESR of the measurements of both motors from 0.6 V to 2.0 V: 1.02 Ω. Furthermore, the constant $K_{f,d}$ that represents both setups adequately is 0.0714 N, starting at a duty cycle of 125, although this is very dependent on the temperature of the setups. With this $K_{f,d}$, the motor constant K_m is estimated to be approximately 0.0155 Nm/A.

3.2 Force sensing in the DIY kit

3.2.1 Effect on teleoperation performance of the DIY kit

As mentioned in Section 2.2.2, there is little literature that discussed the need for an accurate force measurement in teleoperation. Nevertheless, the fact that architectures using force measurements generally outperform architectures that predict the force (e.g. position-computed force) suggests that accuracy is key for this improved transparency; otherwise, the computed/estimated force would likely be sufficiently accurate. To verify this hypothesis, the final teleoperation system's performance could be evaluated with and without the improvements made to optimise the force sensor's accuracy.

3.2.2 Type of force sensor

As discussed in Section 2.2.1, there are many types of force sensors that are used in teleoperation systems. Since the DIY kit is to be assembled by the user, the force sensor should be able to be easily attached and integrated into the DIY setup, which eliminates the otherwise very accurate (but also a bit more expensive) FBG sensor. Out of the remaining discussed force sensors, sensors made of strain gauges seem to be most suited for this DIY kit: they are cheap, compact and can be quite accurate when the strain gauges are placed in a Wheatstone bridge configuration and when the sensor is calibrated. While piezoelectric and capacitive force sensors provide a better dynamic response, this is outweighed by their disadvantages; a piezoelectric force sensor is susceptible to temperature and humidity changes, which means the sensor's performance may be insufficient depending on the location of the user, and a capacitive force sensor is susceptible to parasitic capacitances, which will certainly be present in a DIY kit (mainly) without soldered connections.

Fortunately, the force sensor already implemented in the DIY setup does consist of strain gauges; Frank attached a load-cell to the handle of the robot, between the part with which the operator interacts and the transmission (Figure 1.1). This load-cell operates at 5 V and is capable of measuring weights up to 2 kg, so forces up to $2 \cdot 9.81 = 19.62$ N [62]. Internally, it consists of four strain resistances of each 1 kΩ, arranged in a Wheatstone bridge configuration: Figure 3.14.

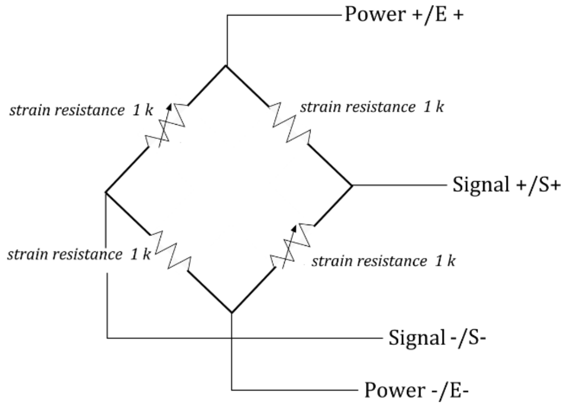


Figure 3.14: Internal structure of the used load-cell [63].

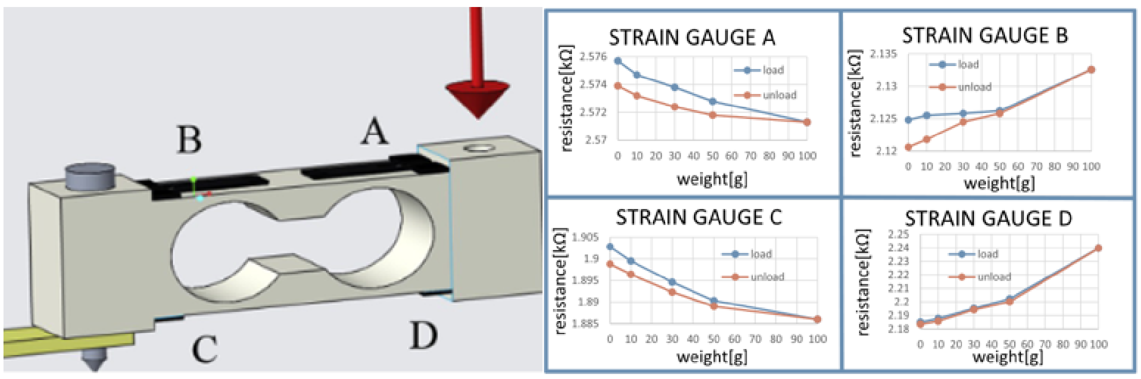


Figure 3.15: Non-linearity of the resistance of strain gauges (made of conductive PLA) with respect to applied force (weight) in a single point bending beam load-cell [64].

As seen in Figure 3.14, the strain gauges are placed on all 4 sides of a diamond shape and the output voltage is measured between the left and right corners. The top and bottom corners are connected to the voltage source. The bridge is placed such that two strain gauges are in compression and two in tension. Looking at the external structure of the implemented force sensor, a single point bending beam (Figure 3.16), the strain gauges are placed on the top and bottom of the beam, like in Figure 3.15 (left).

As indicated with arrows in Figure 3.14, there are two active strain gauges: the top left gauge and the bottom right gauge. As discussed in Section 2.2.1, this is a half Wheatstone bridge configuration and they should be placed such that both are in either compression or tension, so either at A and B or at C and D in Figure 3.15. The output voltage is measured in between

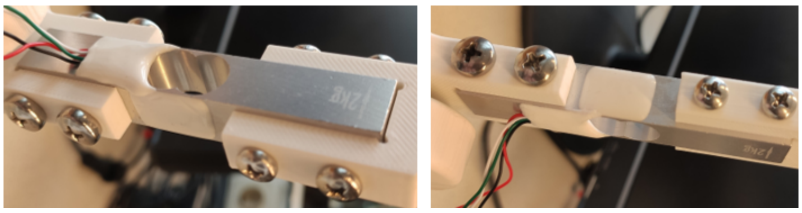


Figure 3.16: The force sensor (load-cell) that is implemented in the DIY setup.

two voltage dividers (left and right) of one active and one non-active strain gauge, leading to Equations 3.9 and 3.10. Here, the strain gauge resistance is $R + \Delta R$ (ΔR only for active strain gauges and depends on the strain), G_F is the gauge factor and l is the strain gauge's length [31].

$$V_{out} = \frac{R}{R+R+\Delta R}V_{supply} - \frac{R+\Delta R}{R+R+\Delta R}V_{supply} = \frac{-\Delta R}{2R+\Delta R}V_{supply} = -V_{supply}\frac{\frac{\Delta R}{2R}}{1+\frac{\Delta R}{2R}} \tag{3.9}$$

$$V_{out} = -\frac{V_{supply}}{2}\frac{\frac{\Delta R}{R}}{1+\frac{\Delta R}{2R}} \approx -\frac{V_{supply}}{2}\frac{\Delta R}{R} = -\frac{V_{supply}}{2}G_F\frac{\Delta l}{l} \tag{3.10}$$

When a force is applied, the wires in the strain gauges get longer and thinner (more resistance) or shorter and thicker (less resistance) depending on their orientation and the direction of the force on the beam, which can also be concluded from equation 3.10: $\frac{\Delta R}{R} = G_F\frac{\Delta l}{l}$, where $G_F > 0$.

Even with this Wheatstone half bridge configuration, which gives twice the output voltage for the same force as the quarter bridge configuration (Section 2.2.1), the output voltage is too small to measure accurately with a microcontroller (Arduino Uno in the DIY setup). Hence, Frank used an HX711 amplifier with 24-bit ADC [65] and a small footprint Arduino library [66].

To get a value in newtons from this amplifier's output, a scaling factor was measured (at 15 N) to convert the output value to a value that should represent the applied force (in newtons).

3.2.3 Shortcomings and limitations of current implementation

While the load-cell should be calibrated for a very linear output voltage versus force relation (0.05% non-linearity [62]), Frank mentions a non-linear error. It is a possibility that the load-cell is not calibrated very well, but this non-linear error could also be caused by his measurement method; first of all, he uses a mechanical force meter, which is difficult to read accurately. Secondly, the load-cell is secured through friction (parallel to the direction of the force measurement) in a bench vise, which could allow for some movement during the measurement of large forces. Finally, the load-cell was placed upright and the force was applied sideways at the top, so the effect of gravity in the measurement increased as the applied force increased.

Furthermore, he notes some sensor noise that he reduces by subtracting the average from 20 measurements of no applied force from the load-cell measurement. However, as mentioned in Section 2.2.3, the sensor noise can originate from all kinds of electrical and mechanical sources. As these noise sources are not all stationary, they may not be properly represented by these measurements to estimate the noise only when the system is turned on.

So, the main shortcomings of the current implementation of the force sensor are the sensor noise and (possibly) non-linearity of the measured force with respect to the applied force.

3.2.4 Improvements of current implementation

To reduce the varying (likely non-deterministic) sensor noise, an adaptive filter can be used, as discussed in Section 2.2.3. To measure or create this filter's input noise in a real system (like the DIY setup), an extra "dummy" load-cell can be used. This dummy load-cell should be placed as close as possible to the active load-cell without measuring the clean signal (force applied to the handle by the operator or task environment) [31]. This way, the output of the dummy load-cell should be mostly uncorrelated with the clean signal, but strongly correlated with the noise of the corrupted signal, resulting in an ideal reference input for the noise cancelling adaptive filter (Figure 2.14).

As for the choice between algorithms discussed in Section 2.2.3, LMS would likely be the best choice considering the DIY kit's limitations and goal; because of its desired cost-effectiveness, the computational complexity should be minimal such that a low-cost microcontroller suffices. Hence, the lower computational complexity of LMS is a clear advantage over (L)RLS. Also, while RLS does perform significantly better in situations with varying noise characteristics, the DIY kit will probably operate in the same room with the same noise sources most of the time, reducing the need for a very fast (and computationally complex) adaptive filter.

While most force sensors are calibrated in the factory, the cost-effective force sensors that are relevant for the DIY kit might not be perfectly linear with respect to the applied force. As discussed in Section 3.2.3, the non-linear error that Frank measured may not be caused by the force sensor, but it is useful to prepare for the scenario in which it is caused by the force sensor.

In this scenario, every DIY setup will need a calibration/linearisation of the force sensor, since every (non-calibrated) force sensor can have a noticeably different output for the same applied force [64]. Since the DIY kit is intended to be bought and assembled by the users themselves [10], this calibration would have to be performed by them as well. Therefore, an easy and cost-effective calibration method would have to be designed, like using known weights to estimate and compensate for the nonlinear behaviour [67].

3.2.5 Experiments

As mentioned in Section 3.2.3, the main shortcomings of the force sensor seem to be sensor noise and non-linear error, decreasing its accuracy for forces other than the calibration forces. However, especially for the non-linear error, it is useful to verify that they are indeed sufficiently present to decrease the system's performance before implementing solutions that require computational resources (adaptive filter) or extra steps by the user (linearisation).

Due to the aforementioned load-cell's specified 0.05% non-linearity [62] and Frank's measurement method to determine its accuracy, it seems unlikely that the non-linear error in the force measurement is as significant as Frank measured. To verify this hypothesis, experiment 3.3 is conducted to measure the sensor's error more accurately, using the configurations listed in Table 3.2. Since the mechanical force meter of 20 N is not very accurate at low forces, the measurements up to 5 N and 10 N are performed once more with mechanical force meters that can measure up to 5 N and 10 N respectively.

In experiment 3.4, the load-cell's noise[3] is determined in various (common) environments and in case of disturbances, as stated in Table 3.2

Experiment 3.3: Determining the load-cell's non-linearity and accuracy

Method: First of all, the load-cell is fixed horizontally in the bench vise such that the direction of the applied force is the same as the direction in which the vise is exerting force. This greatly reduces the possible play in the vise's grip during the experiment. By pulling the other end of the load-cell to the left and right, gravity has no effect on the force measurement (in both directions).

Since a digital force meter with an accuracy rating was not available, the same type of mechanical force meter as the one in Frank's experiment (based on a spring) is used. To measure the

[3]In this context, the term "noise" is used to refer to the load-cell's measurements of (environmental) noise, interference and (physical) disturbances.

Table 3.2: Overview of the experiments conducted to determine the quality of the current force measurement.

Experiment	Objective	Tested configurations
3.3	Determining the load-cell's non-linearity and accuracy	Comparing load-cell's force measurements to measurements of 20 N mechanical force meter for the range of forces used in previous work [44]: -15 to 15 N (each integer value measured three times)
		Comparing load-cell's force measurements to measurements of 5 N, 10 N and 20 N mechanical force meters for forces from -15 to 15 N
3.4	Determining the load-cell's noise	Noise measured in quiet gathering space with ventilation
		Noise measured in busy gathering space with ventilation
		Noise measured in robotics lab
		Noise measured next to PC and keyboard
		Noise measured in case of physical disturbances of the setup's base
		Noise measured during teleoperation (paddle disconnected from motor)

precision of the force sensor, each integer value between 0 and 15 N is measured 3 times for both directions. The forces are applied 3 times in one direction and then 3 times in the other direction to observe if there is fatigue when the load-cell is loaded many times in one direction.

The force meters are attached to the outer screw hole of the load-cell. This results in the measurement setup of Figure 3.17.

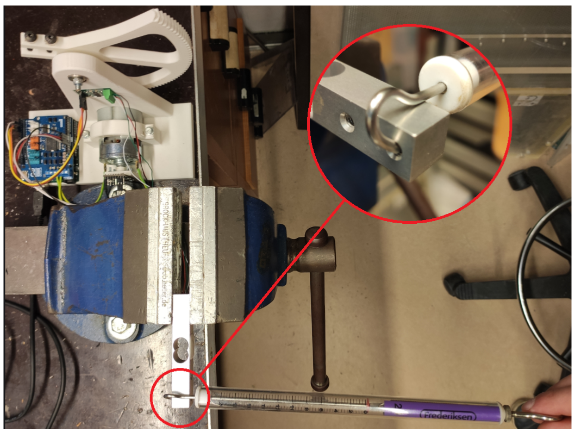


Figure 3.17: Measurement setup used in experiment 3.3 to determine the load-cell's non-linearity.

The load-cell is calibrated at 15 N using the code provided in the HX711 library [66] that Frank suggested. Taking the average of the calibrations in both directions results in a scaling factor of 1080 to convert the load-cell's value to a weight in grams, which is then multiplied by 0.00981 to obtain a force in newtons ($F = m \cdot g$).

The non-linearity of the force measurement error is determined by taking the average of the deviation percentages (deviation from a linear line between the -15 N and 15 N measurement) at each measurement. These percentages are calculated with respect to the full scale output: the difference between the minimum and maximum measured values. Then, the average of the non-linearity percentages (of the three times the measurements were performed) is calculated. Furthermore, the accuracy and precision are calculated by taking the averages of the mean of the absolute difference between the applied and measured forces and the mean of the standard deviation respectively.

Results: Measuring the forces in the aforementioned order resulted in the plot of Figure 3.18. Visually, the load-cell's output value is very linear with respect to the applied force. This is supported by the calculated accuracy and precision values: the accuracy of 0.05 N indicates that on average, the force measurement deviates 0.05 N from the actual applied force in both directions. The precision of 0.03 N indicates the average difference between the (3) measured values at each applied force.

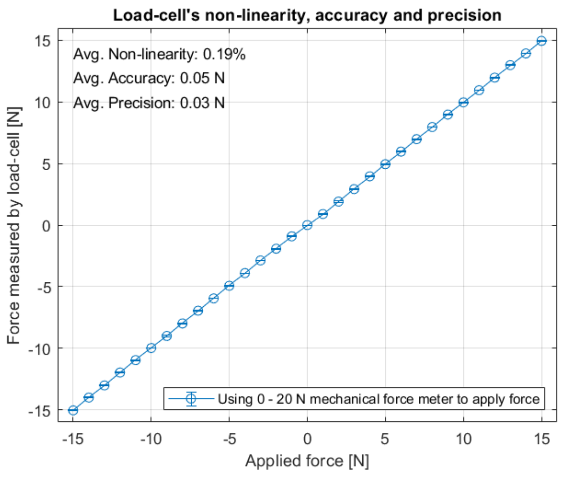


Figure 3.18: Load-cell measurements using a 0-20 N mechanical force meter.

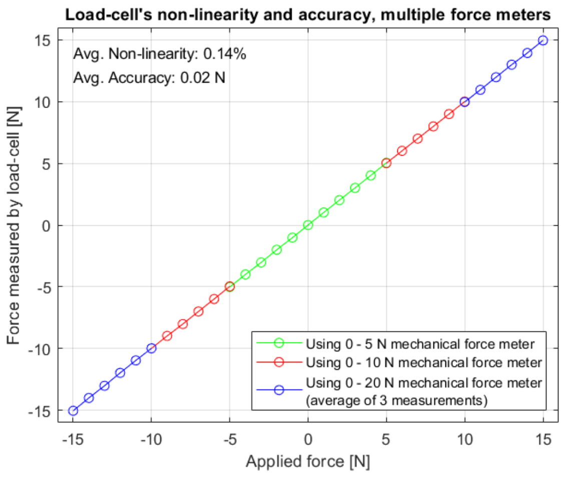


Figure 3.19: Load-cell measurements using multiple mechanical force meters.

When using the other mechanical force meters to measure forces up to 5 N and 10 N, the load-cell's values are much closer to the value read from the mechanical force meter. These measurements result in Figure 3.19, where the precision is omitted because these measurements were only recorded once to show the improved average accuracy: 0.02 N as opposed to the 0.05 N when using only the 20 N force meter.

Discussion: Using mechanical force meters with less range results in an increased measured accuracy since the 20 N force meter seemed less accurate at the start of its measurement range. Nonetheless, this is also caused by the lower force values being easier to read on these mechanical force meters with less range, allowing for a more precise force application. The final accuracy of 0.02 N is a very good result considering the force was applied by reading off the mechanical force meter and holding it at that force during the measurement.

While the measured non-linearity is 0.14%, the load-cell's specified non-linearity of 0.05% [62] is certainly plausible if the measurements are performed with automated and accurate force application. Also, while not likely to have affected the measurements, it should be noted that this experiment was performed in a robotics lab with more electromechanical interference than the usual environment of the DIY kit.

Conclusion: From this experiment, it can be concluded that calibration (other than the determined scaling factor) or linearisation of the load-cell is not necessary before its implementation in the DIY setup; the calibration performed in the factory is sufficient.

Experiment 3.4: Determining the load-cell's noise

Method: In this experiment, the effect of external disturbances and noise sources on the force sensor measurement is explored by placing the setup in various environments and disturbing it in various ways (Table 3.2), without applying any force at the handle.

First, the DIY setup is placed in quiet and busy environments with ventilation to represent the environments it will probably spend most time in; a study/work place or living room. Here, the load-cell's noise is first measured without any intentional disturbances, and then with disturbances: moving a desk chair and opening and closing a door. Secondly, the setup is placed in a lab (without any intentional disturbances), where there are much more mechanical vibrations and possibly some electrical interference that can influence the measurement, as discussed in Section 2.2.3. Next, the setup is placed in a room with as little disturbances as possible: (a quiet room with minimal ventilation), where the effects of a desktop PC and keyboard very close to the setup are investigated. Also, the base of the setup is moved/touched slightly to represent unintended disturbances like someone bumping into the table that the setup is placed on.

Furthermore, the teleoperation setup is turned on (running the provided position-computed force code) to examine the effect of the motor's electromagnetic radiation. Here, the paddle with the force sensor is disconnected from the motor to exclude the forces caused by the movement of the paddle and handle. The fact that the load-cell is not perfectly upright does not influence the measurements as its value without applied force is always measured first and removed from its later measurements, thereby removing the effect of gravity in this experiment.

An extra Arduino Uno is used to perform these load-cell measurements, since using one of the Arduino's in the setup for this caused instability during teleoperation because of the increased loop time (due to the serial communication). This results in the measurement setup of Figure 3.20.

To record and save the measurements, the force (in millinewton, using the aformentioned scaling factor of 1080) and time (in seconds) are sent to the computer via serial communication. To receive and save this data, the same Python script of [58] is used and slightly modified to handle the correct types of data (floats instead of integers). Finally, the data is retrieved, plotted and analysed using Matlab, where the mean, standard deviation, (maximum) peak-to-peak value and power is calculated.

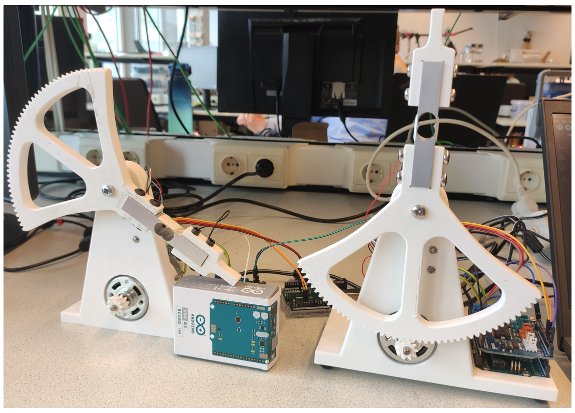


Figure 3.20: Measurement setup used in experiment 3.4 to determine the load-cell's noise in case of an active teleoperation system.

Results: The results of the load-cell measurements in different environments, with and without disturbances, can be found in Figure 3.21.

In the top-left and top-center plots, the load-cell's noise in quiet environments can be seen. Around 60 s in both environments, a chair was moved and around 100 s, a door was opened and closed. These disturbances are not distinctly visible in the measurements. Also, the difference in noise power between a quiet gathering space and busy gathering space is negligible.

The top-right plot shows the noise in the robotics lab, which is clearly not biased around zero. Therefore, the noise power is significantly higher in this robotics lab than in the previous environments.

In the bottom-left plot, the keyboard was used at 60 s, and the PC was turned on at 120 s. Again, these disturbances are not distinguishable in the measurements.

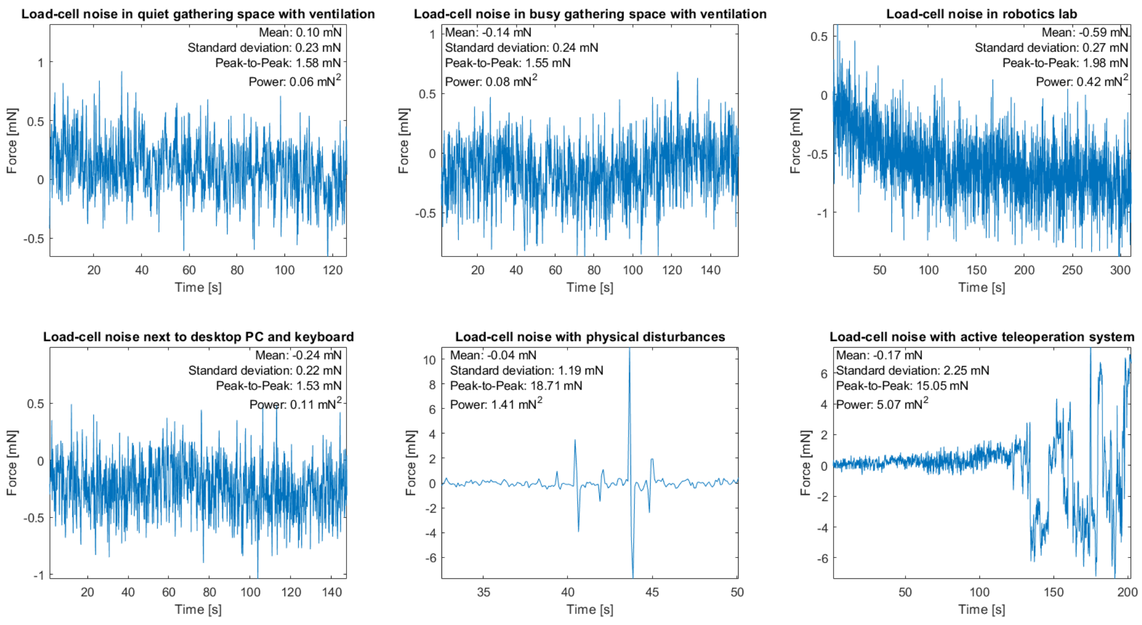


Figure 3.21: Load-cell noise measurements in different environments and with (physical) disturbances.

In the bottom-centre plot of Figure 3.21, the 3 small peaks represent disturbances orthogonal to the sensor's measurement axis, while the 2 large peaks represent disturbances parallel to the measurement axis. These disturbances are created by tapping/moving the setup's base.

Finally, the effect of electromagnetic interference from the motor during active teleoperation can be seen in the bottom-right plot. At roughly 60 s, the DIY setups were turned on and the one on the task side (right setup in Figure 3.20) was interacted with. This can be observed in the measurement, as the noise amplitude seems to be slightly higher from 60 s to 120 s.

At roughly 120 s the setup on the operator's side, which includes the load-cell that is measuring the noise, was interacted with as well to increase the torque that the motor had to provide to follow the other DIY setup. This clearly increased the measured noise amplitude. The large peaks in the noise plot (both positive and negative) occurred when the motor's transmission gear was held still by the operator while the other setup's handle was moved.

Discussion: While the (mean-square) noise power in the typical environment of the DIY kit with typical disturbances (a desktop PC, keyboard, opening/closing door and moving chair) is not significant, the noise power in the robotics lab is relatively high (4 times higher). This difference is a consequence of the noise not being zero-biased: it progressively drifts further away from zero due to an external noise source in the lab. It may not be high enough to noticeably impact the felt transparency (being only a few millinewtons), but it can be detrimental for the teleoperation system's passivity and stability when it results in energy generation.

The noise power in case of physical disturbances of the setup and electromagnetic interference during teleoperation is even higher. While an adaptive filter of low complexity may not be able to filter out the short and intense physical disturbances, the electromagnetic interference should be attenuated since it is always present when the teleoperation system is active. Moreover, this increased noise power does indeed seem to originate from the (electromagnetic radiation of) the motor in the DIY setup, as it is proportional to the current draw of the motor; the noise power is largest when the motor shaft's rotation was resisted, therefore requiring a lot of torque (which is directly related to current).

Conclusion: From the measurements in Figure 3.21, three valuable conclusions can be drawn. Firstly, external disturbances such as (loud) voices and movements of nearby objects do not notably increase the already present sensor noise in environments with ventilation. The same can be concluded for turning on a nearby PC or typing on a keyboard. However, direct physical disturbances of the setup do influence the load-cell's measurement considerably. Secondly, the electrical interference and mechanical vibrations in a lab environment also influence the load-cell's measurement notably; the noise power can be as much as 4 times higher compared to a typical gathering space. Thirdly, when the teleoperation system is active, the electromagnetic radiation of the motor (in the same DIY setup) is very noticeable in the load-cell's measurements and proportional to its current draw.

3.3 Pseudocode and data-flow diagrams

3.3.1 Current iteration of the DIY setup

As mentioned in Section 3.1.5, the current iteration of the DIY kit consists of two Arduino Uno's (one on each setup), and the Processing user interface has not yet been integrated. Currently, the position-computed force architecture is functional in this iteration, but one of the Arduino's seems to reset occasionally. To streamline the integration of the control architectures and adaptive filter, this given code will first be examined with pseudocode and a data-flow diagram.

The pseudocode for both microcontrollers' codes and the data-flow diagram can be found below. In essence, it functions exactly like the standard architecture (Figure 2.5), but there are some notable differences. Firstly, the motors are controlled via PWM (voltage, as discussed in Section 2.1.5). Hence, a "steer" value is computed at the task side (with control gain K_p) instead of the force. Secondly, a "correction" (compensation) is added to overcome the friction in the setup, which otherwise prevents the motors from moving for low steer values. Furthermore, if the steer value is zero, the force felt by the operator should be zero as well. Without the compensation, the operator would still feel the (Coulomb) friction in the setup, which is experienced as a nonzero force. The correction value (in terms of PWM duty cycle) is applied in the same direction as the rotation of the motor, determined with the velocity estimator in Figure 3.22.

At the start of each loop, the loop time of the previous loop is calculated and the current loop will wait until the desired loop time is reached. This way, the code will always run at the same loop frequency, which is important in case of differentiation or integration (where the loop time is used in the calculation). Nevertheless, the loop could occasionally take longer than it is allowed to because of too many computations in a single loop. If this happens, the Arduino's built-in LED is supposed to blink to indicate (to the user) that the desired loop frequency could not be reached. However, in the current (not final) iteration, this is indicated via serial communication to the computer. Ideally, the loop frequency should be chosen such that this never happens when the DIY kit is used as intended. In the current iteration, the loop frequency is set to 500 Hz, which should be sufficiently high to minimise any discretisation effects that result from converting continuous models/functions to discrete ones that can be implemented in the code.

Pseudocode for Arduino on human side

```
#include required libraries
#define motor & sensor pins
#define settings & parameters
Initialise variables
void setup(){
    begin serial monitor
    increase PWM frequency to inaudible
    initialise position (angle) sensor
    initialise wired I2C communication
    send local angle whenever requested
    read computed steer whenever received
}
void loop{
    calculate loop time of previous loop
    while (previous loop time < desired loop time){
        wait until the desired loop time is reached
    }
    if (previous loop time > desired loop time){
        indicate that the desired loop time is exceeded
    }
    read local angle
    local steer = - computed steer
    if (local steer == 0){
        if (moving backwards){
            output PWM = local steer - correction
        } else if (moving forwards){
            output PWM = local steer + correction
        } else {
            output PWM = local steer
        }
    } else if (local steer < 0){
        output PWM = local steer - correction
    } else if (local steer > 0){
        output PWM = local steer + correction
    }
    previous angle = current angle
    determine direction
    ensure output PWM is within bounds
    motor PWM = output PWM
}
```

Pseudocode for Arduino on task side

```
#include required libraries
#define motor & sensor pins
#define settings & parameters
Initialise variables
void setup(){
    begin serial monitor
    increase PWM frequency to inaudible
    initialise position (angle) sensor
    initialise wired I2C communication
}
void loop{
    calculate loop time of previous loop
    while (previous loop time < desired loop time){
        wait until the desired loop time is reached
    }
    if (previous loop time > desired loop time){
        indicate that the desired loop time is exceeded
    }
    send message with computed steer
    request message with remote angle
    read local angle
    error = remote angle - local angle (in radians)
    computed steer = Kp*error
    if (computed steer == 0){
        if (moving backwards){
            output PWM = computed steer - correction
        } else if (moving forwards){
            output PWM = computed steer + correction
        } else {
            output PWM = computed steer
        }
    } else if (computed steer < 0){
        output PWM = computed steer - correction
    } else if (computed steer > 0){
        output PWM = computed steer + correction
    }
    previous angle = current angle
    determine direction
    ensure output PWM is within bounds
    motor PWM = output PWM
}
```

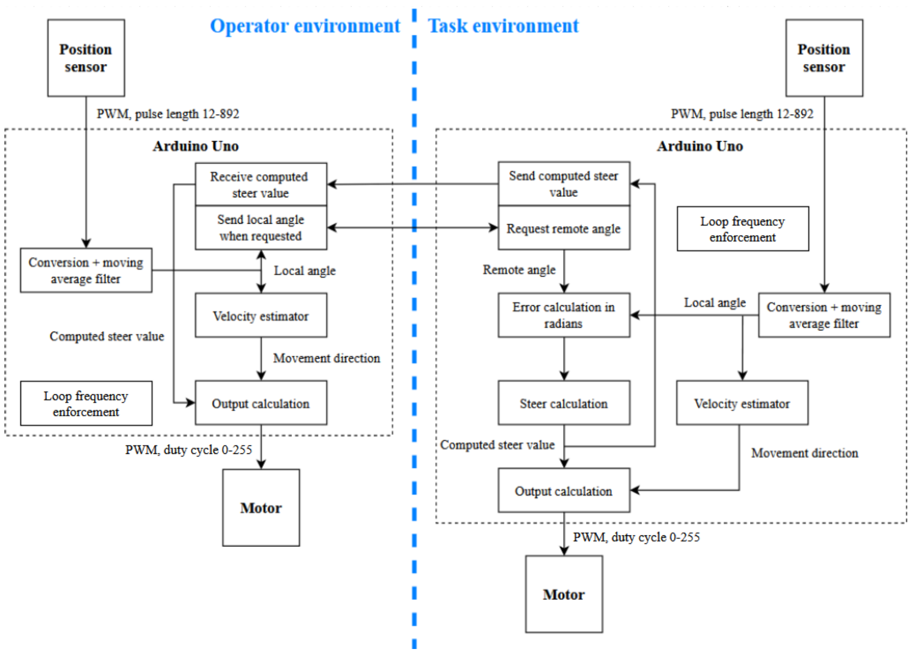


Figure 3.22: Data-flow diagram of the provided position-computed force architecture used in the current iteration of the DIY kit.

3.3.2 Implementation of the adaptive filter in the position-measured force architecture

There are multiple options to implement an adaptive filter with the LMS algorithm in the teleoperation code. There is an Arduino library [68], which would be easy to implement, but it is untested, dated and probably more computationally complex than a direct implementation of the algorithm in the control architecture code. This implementation in the control architecture code also rules out the use of Simulink/Matlab's LMS function [69], which programs and controls the Arduino completely by itself. Hence, the theoretical explanation of Section 2.2.3 will be implemented directly in the teleoperation code.

It is possible that the LMS adaptive filter distorts the relevant force signal; because the dummy load-cell (the input signal for the filter) measures the effects of gravity and inertia, which are changing relatively slowly or constant and therefore correlated with many signals, the filter can create almost everything from the dummy load-cell signal when these effects are sufficiently present. This compromises the implementation of the adaptive filter as noise canceller (Figure 2.14), as this assumes that the filter's input (dummy load-cell) signal and relevant force signal are almost completely uncorrelated; as explained in Section 2.2.3, the estimation of the clean force signal is the filter's output (estimation of the noise in the active load-cell signal) subtracted from the active load-cell signal. While distortion may occur, the LMS algorithm is too "slow" to completely cancel the typical relevant force signal: moving the handle back and forth and thereby applying varying force in different directions.

Moreover, the constant $K_{f,d}$ that was derived in experiment 3.2 (Section 3.1.6) is to be used in the position-measured force architecture (and 4-channel architecture) to translate the force measured by the load-cell in the task environment into a PWM duty cycle that should result in the same force felt by the operator. Since this measured force is opposite to the torque on the motor on the task side (because of the transmission with 2 gears: Figure 1.1), the calculated PWM duty cycle for the human side already acts in the correct direction.

These additions result in the data-flow diagram of Figure 3.23 and a pseudocode that is notably different from the code currently used for the position-computed force architecture:

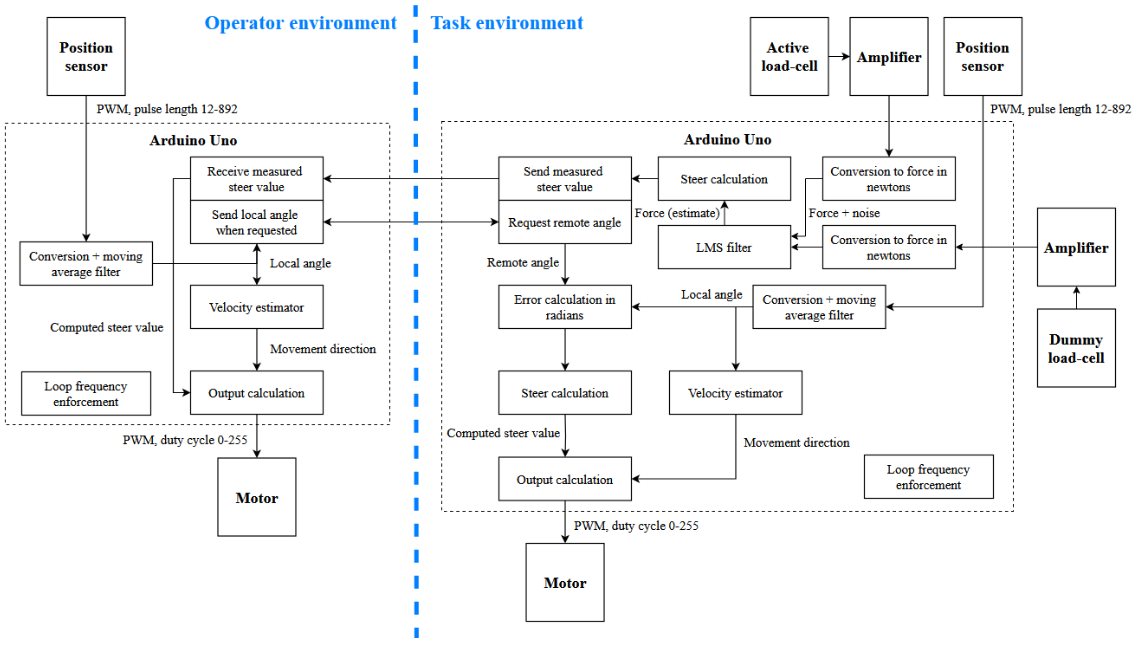


Figure 3.23: Data-flow diagram of the envisioned code for implementing the position-measured force architecture in the DIY kit.

Pseudocode for Arduino on human side

```
#include required libraries
#define motor & sensor pins
#define settings & parameters
Initialise variables
void setup(){
    begin serial monitor
    increase PWM frequency to inaudible
    initialise position (angle) sensor
    initialise wired I2C communication
    send local angle whenever requested
    read measured steer whenever received
}
void loop{
    calculate loop time of previous loop
    while (previous loop time < desired loop time){
        wait until the desired loop time is reached
    }
    if (previous loop time > desired loop time){
        indicate that the desired loop time is exceeded
    }
    read local angle
    local steer = measured steer
    if (local steer == 0){
        if (moving backwards){
            output PWM = local steer − correction
        } else if (moving forwards){
            output PWM = local steer + correction
        } else {
            output PWM = local steer
        }
    } else if (local steer < 0){
        output PWM = local steer − correction
    } else if (local steer > 0){
        output PWM = local steer + correction
    }
    previous angle = current angle
    determine direction
    ensure output PWM is within bounds
    motor PWM = output PWM
}
```

Pseudocode for Arduino on task side

```
#include required libraries
#define motor & sensor pins
#define settings & parameters
Initialise variables
void setup(){
    begin serial monitor
    increase PWM frequency to inaudible
    initialise position (angle) sensor
    initialise wired I2C communication
    initialise force sensor
}
void loop{
    calculate loop time of previous loop
    while (previous loop time < desired loop time){
        wait until the desired loop time is reached
    }
    if (previous loop time > desired loop time){
        indicate that the desired loop time is exceeded
    }
    read measured force
    convert measured force to newtons
    LMS filter measured force
    calculate measured steer: filtered force / K_fd
    send message with measured steer
    request message with remote angle
    read local angle
    error = remote angle − local angle (in radians)
    computed steer = Kp*error
    if (computed steer == 0){
        if (moving backwards){
            output PWM = computed steer − correction
        } else if (moving forwards){
            output PWM = computed steer + correction
        } else {
            output PWM = computed steer
        }
    } else if (computed steer < 0){
        output PWM = computed steer − correction
    } else if (computed steer > 0){
        output PWM = computed steer + correction
    }
    previous angle = current angle
    determine direction
    ensure output PWM is within bounds
    motor PWM = output PWM
}
float LMS {
    read current input signal (dummy load−cell)
    read current desired output signal (active load−cell)
    shift saved forces by one delay, discard the oldest
    save current force
    calculate output with filter coefficients & saved forces:
    calculate error between current & desired output signals
    calculate next filter coefficients with the error,
      convergence factor & saved forces
    return the error as system output: the force estimate
}
```

3.3.3 Implementation of the adaptive filter in the 4-channel architecture

The main difference between the 4-channel and position-(measured or computed) force architectures is the number of communication channels (Section 2.1.4); the 4-channel architecture communicates variables through 4 communication channels as opposed to 2, as seen in Figure 2.10. These variables are all outputs of the (4) feed-forward controllers, the gains of which (K_{p_1}, K_{p_2}, K_{p_3} and K_{p_4}) are numbered in the same way as in Figure 2.10. Since the measured forces of both setups are used in the 4-channel architecture, the LMS adaptive filter is to be implemented on the human side as well. Notably, the 4-channel architecture uses the motors' rotational velocities instead of angles, so differentiation of the angles is required. Therefore, it is even more crucial for this architecture that the loop time remains the same at all times.

These changes results in the data-flow diagram of Figure 3.24 and the following pseudocode:

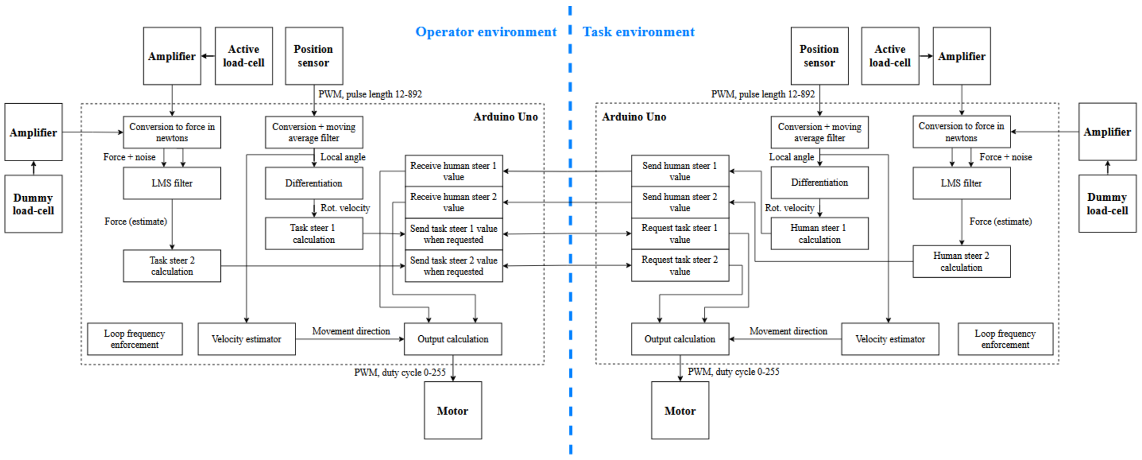


Figure 3.24: Data-flow diagram of the envisioned code for implementing the 4-channel architecture in the DIY kit.

Pseudocode for Arduino on human side

```
#include required libraries
#define motor & sensor pins
#define settings & parameters
Initialise variables
void setup(){
    begin serial monitor
    increase PWM frequency to inaudible
    initialise position (angle) sensor
    initialise wired I2C communication
    send task steer 1 & 2 whenever requested
    read human steer 1 & 2 whenever received
    initialise force sensor
}
void loop{
    calculate loop time of previous loop
    while (previous loop time < desired loop time){
        wait until the desired loop time is reached
    }
    if (previous loop time > desired loop time){
        indicate that the desired loop time is exceeded
    }
    read current (local) angle
    read measured force
    rotational velocity =
        (current angle - previous angle) / loop time
    LMS filter measured force
    task steer 1 = Kp_1 * rotational velocity
    task steer 2 = Kp_3 * filtered force / K_fd
    local steer = human steer 1 + human steer 2
    if (local steer == 0){
        if (moving backwards){
            output PWM = local steer - correction
        } else if (moving forwards){
            output PWM = local steer + correction
        } else {
            output PWM = local steer
        }
    } else if (local steer < 0){
        output PWM = local steer - correction
    } else if (local steer > 0){
        output PWM = local steer + correction
    }
    previous angle = current angle
    determine direction
    ensure output PWM is within bounds
    motor PWM = output PWM
}
float LMS {
    read current input signal (dummy load-cell)
    read current desired output signal (active load-cell)
    shift saved forces by one delay, discard the oldest
    save current force
    calculate output with filter coefficients & saved forces:
    calculate error between current & desired output signals
    calculate next filter coefficients with the error,
      convergence factor & saved forces
    return the error as system output: the force estimate
}
```

Pseudocode for Arduino on task side

```
#include required libraries
#define motor & sensor pins
#define settings & parameters
Initialise variables
void setup(){
    begin serial monitor
    increase PWM frequency to inaudible
    initialise position (angle) sensor
    initialise wired I2C communication
    initialise force sensor
}
void loop{
    calculate loop time of previous loop
    while (previous loop time < desired loop time){
        wait until the desired loop time is reached
    }
    if (previous loop time > desired loop time){
        indicate that the desired loop time is exceeded
    }
    read current (local) angle
    read measured force
    rotational velocity =
        (current angle - previous angle) / loop time
    LMS filter measured force
    human steer 1 = Kp_4 * rotational velocity
    human steer 2 = Kp_2 * filtered force / K_fd
    send message with human steer 1 & 2
    request message with task steer 1 & 2
    local steer = task steer 1 + task steer 2
    if (local steer == 0){
        if (moving backwards){
            output PWM = local steer - correction
        } else if (moving forwards){
            output PWM = local steer + correction
        } else {
            output PWM = local steer
        }
    } else if (local steer < 0){
        output PWM = local steer - correction
    } else if (local steer > 0){
        output PWM = local steer + correction
    }
    previous angle = current angle
    determine direction
    ensure output PWM is within bounds
    motor PWM = output PWM
}
float LMS {
    read current input signal (dummy load-cell)
    read current desired output signal (active load-cell)
    shift saved forces by one delay, discard the oldest
    save current force
    calculate output with filter coefficients & saved forces:
    calculate error between current & desired output signals
    calculate next filter coefficients with the error,
      convergence factor & saved forces
    return the error as system output: the force estimate
}
```

4 Design & Realisation

In this chapter, the solutions to the assignment, found in the previous chapter, are implemented in the DIY kit. Since the quality of the force measurement is crucial to the performance of the position-measured force architecture (discussed in Section 2.2.2), the improvement of this force measurement is addressed first. In Section 4.1, the implementation of the conditional adaptive filter is explained together with new terminology better suited for this specific implementation of the filter (as noise canceller). Also, experiments are introduced and conducted to find the optimal implementation of the filter in the DIY kit. To conclude, the final implementation of the conditional LMS filter is discussed, as well as a potential performance improvement (by modifying the load-cell amplifiers), which are both evaluated in Chapter 5.

In Section 4.2, the implementation of the control architectures is discussed with their pseudocodes and optimised through experiments. However, as mentioned in Section 1.7, the implementation of the 4-channel architecture (and Processing interface) is not completed. So, to conclude, only the final implementation of the position-measured force architecture is discussed, which is evaluated in Chapter 5.

4.1 Force sensing

4.1.1 Conditional adaptive filter

As stated in Section 3.2.4, the LMS algorithm will be used for calculating the filter coefficients of the adaptive filter because of its relatively low computational complexity. This adaptive filter should remove the sensor noise of the load-cell in an active teleoperation system and reduce the effect of physical disturbances on the measurement (measured in the bottom-right and bottom-centre plots of Figure 3.21 respectively). Furthermore, it should reduce the measured force caused by gravity pulling on the top of the setup and the moment of inertia of the top of the setup. The theoretical explanation of the algorithm can be found in Section 2.2.3, which will be applied directly in the control architecture code, as discussed in Section 3.3.2. The clean signal (force applied at the handle) and noise should be mostly uncorrelated, as otherwise the filter will also replicate (and thereby cancel) the clean signal with its input signal (the noise).

Since the standard terminology of an adaptive filter might be confusing when it it used as a noise canceller (e.g. the desired signal for the filter is the clean signal with the noise), terminology that is specific to this application will be used from now on: Table 4.1.

Table 4.1: New terminology to describe the relevant signals for the adaptive filter used as noise canceller.

Standard terminology	Equation 2.35	Figure 2.14	Meaning in this application	New terminology
Desired signal	$d(n)$	$y(n)$	The force applied at the handle and noise measured the by active load-cell	Active load-cell signal
Error signal (system output)	$e(n)$	$e(n)$	The estimation of the applied force in the signal from the active load-cell	Force estimate
Filter output	$y(n)$	$\hat{y}(n)$	The estimation of the noise in the signal from the active load-cell	Noise estimate
Filter input	$x(n)$	$x(n)$	The noise measured by the dummy load-cell, ideally the same noise as measured by the active load-cell	Dummy load-cell signal

However, before the adaptive filter can be applied in the teleoperation system's code, the filter order and step size (convergence factor) have to be determined.

4.1.2 Extra "dummy" load-cell

To implement an adaptive filter (as noise canceller) in the DIY setup, an extra load-cell is required that measures the same noise and disturbances as the load-cell between the handle and paddle (Section 3.2.4). This dummy load-cell should be placed as close as possible to the active load-cell for maximum correlation between the sensor noise of both load-cells. In the first experiment, this correlation will be determined.

Before potentially redesigning some of the 3D-printed parts to accommodate the dummy load-cell and amplifier and purchasing these extra components, the load-cell of the other DIY setup and its amplifier are used and the experiments to determine the order and step size of the adaptive filter. The dummy load-cell is attached next to the active load-cell using threaded rods and its HX711 amplifier is attached next to the active load-cell's amplifier to ensure it measures the same noise and (physical) disturbances. Evenmore, the handle of the other setup remains attached to the dummy load-cell such that the effect of gravity pulling on the the mass of the handle is seen as noise and therefore cancelled by the filter. This results in the measurement setup of Figure 4.1.

Also, next to the aforementioned similar placement of the amplifiers, the wires of both load-cells were held together by tie-wraps to ensure that any electromagnetic interference is the same in both load-cells' measurements.

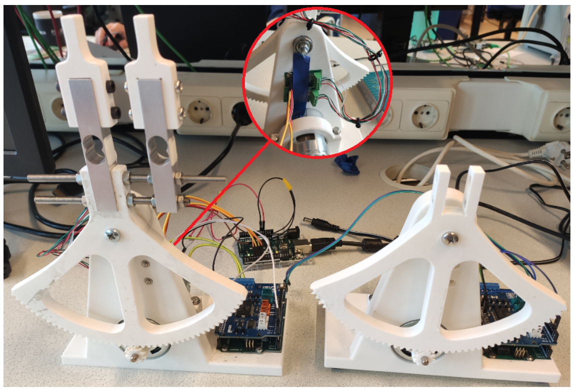


Figure 4.1: Measurement setup used in experiments 4.1 and 4.2 to determine the order and step size of the adaptive filter (as noise canceller) and verify the placement of the dummy load-cell.

4.1.3 Experiments

In experiment 4.1, the placement of the dummy load-cell is evaluated by verifying that it measures similar noise and disturbances as the active load-cell.

In experiment 4.2a, with the help of Matlab, the optimal combination of filter order and step size for the LMS algorithm (listed in Table 4.2) is determined for attenuating the noise and disturbances that affected the load-cell's measurement the most in experiment 3.4 (Figure 3.21): short and intense disturbances, longer and less intense disturbances, and electromagnetic interference from the motor during teleoperation. To determine if the LMS algorithm is too "slow" to attenuate short and intense disturbances, it is compared to the faster and more complex RLS algorithm (Section 2.2.3) with the optimal parameters for this measurement, once again found using Matlab from the filter orders and forgetting factors listed in Table 4.2.

As the optimal LMS filter parameters were found to distort the force estimate when a force is applied at the handle (explained in Section 3.3.2), some variants of other algorithms were used in the adaptive filter to examine if they also distort the clean signal. These algorithms include RLS, LRLS and Normalised LMS (NLMS) with and without regularisation factor (RF), all applied in Matlab using the respective functions. For all algorithms, the clean signal was distorted to the same extent as with LMS, suggesting that the correlation between the signals is too high and the adaptive filter is simply doing what it is supposed to: matching the dummy load-cell signal to the active load-cell signal. Hence, this distortion has to be solved by either decorrelating the dummy load-cell signal and clean signal (through Principal Component Analysis) or disabling the LMS filter when this distortion occurs, which is considered in experiment 4.2b.

In experiment 4.3, the optimal filter parameters (found using Matlab) are verified for a live implementation on the Arduino Uno by slightly changing them to the ones listed in Table 4.2 and observing the effect on its performance in terms of cancelling short and long disturbances (movements of the paddle). To reduce the delay between the load-cells' measurements as much as possible, the dummy load-cell is placed even closer to the active load-cell by removing the handle.

Table 4.2: Overview of the experiments conducted for the improvement of the force sensor in Chapter 4. The ranges for the filter order, step size and forgetting factor were determined mostly empirically.

Exp.	**Objective**		**Tested configurations**	
4.1	Verifying the placement of the dummy load-cell		Attached with threaded rods next to active load-cell	
4.2a	Finding the optimal filter parameters for the adaptive filter in case of:	Short and intense disturbances	LMS with filter order of 1 - 100 (in steps of 1) and step size of 10^{-10} to 1 (in steps of 10^{-10})	
			RLS with filter order of 1 - 100 (in steps of 1) and forgetting factor of 0.95 - 1 (in steps of 0.005)	
		Longer, less intense disturbances	LMS with filter order of 1 - 100 (in steps of 1) and step size of 10^{-10} to 1 (in steps of 10^{-10})	
		Electromagnetic interference during teleoperation	LMS with filter order of 1 - 500 (in steps of 1) and step size of 10^{-10} to 1 (in steps of 10^{-10})	
4.2b	Finding the optimal solution for the distortion of the force estimate		Principal Component Analysis before LMS filter	
			Enabling LMS filter conditionally: only when force is applied at the handle	
4.3	Determining the optimal conditional LMS filter parameters for implementation on Arduino: filter order, step size, disable threshold & timer		Filter order = 1	Step size = $5 \cdot 10^{-6}$
				Step size = $1 \cdot 10^{-5}$
				Step size = $2 \cdot 10^{-5}$
			Filter order = 2	Step size = $1 \cdot 10^{-5}$
			Filter order = 3	Step size = $1 \cdot 10^{-5}$

Experiment 4.1: Verifying the placement of the dummy load-cell

Method: To verify the placement (and attachment) of the dummy load-cell seen in Figure 4.1, the measurements of both load-cells are compared in case of sensor noise with and without physical disturbances (tapping/moving) of the setup's base. Furthermore, the extra Arduino Uno is used once more to avoid instability when recording the measured force in the active teleoperation system.

Results: First of all, the sensor noise measured by both load-cells in the robotics lab can be seen in Figure 4.2. Every characteristic of the measured noise is very similar between the load-cells. Also, it is noteworthy that the source of the measured "downwards drift" in the top-right plot of Figure 4.2 seems to have been absent in the robotics lab during this measurement (which was conducted in a less occupied lab).

In case of sideways physical disturbances (parallel to the force measurement axis), the measurements of both load-cells also correspond quite well, as can be deduced from the first 6 peaks in Figure 4.3. However, the physical disturbances measured by the dummy load-cell are notably (up to 0.1 s) delayed.

Discussion: The fact that the characteristics of the load-cells' noise measurements are very similar implies that they do indeed measure the same environmental noise (and interference). While the noise source of the "downwards drift" of the measurement in Figure 3.21 seems to

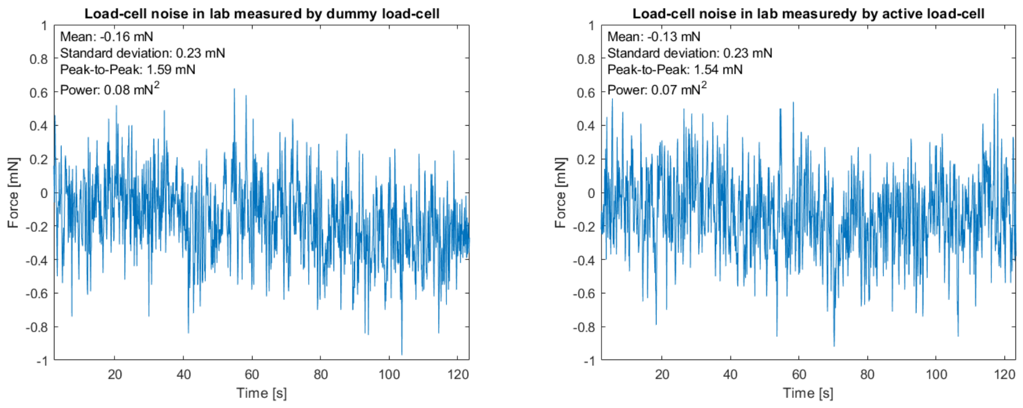


Figure 4.2: Noise measured by active and dummy load-cells in the robotics lab.

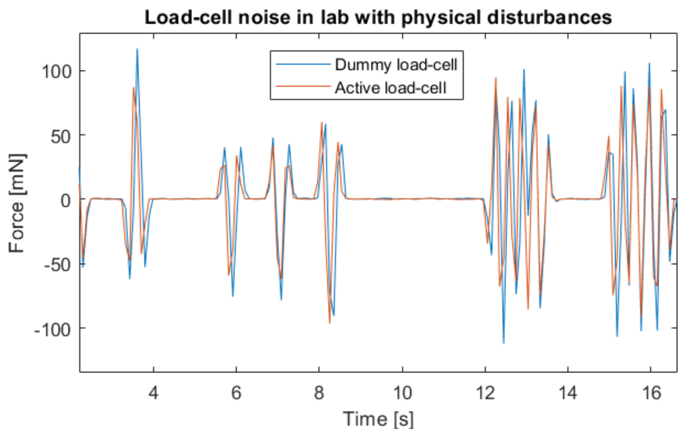


Figure 4.3: Load-cells' measurements in case of disturbances applied to the setup's base parallel to the force measurement axis, conducted in the robotics lab with the setup of Figure 4.1.

have been absent in the robotics lab, the noise power is still larger than in the other environments of Figure 3.21. So, the robotics lab does increase the noise power in the load-cells' measurements.

Since the attachment of the dummy load-cell is very rigid, the measured delay of up to 0.1 second likely originates from the sampling rate of the HX711 amplifiers; this can not exceed 10 samples per second (SPS) [65] (the purchased version does not allow selecting 80 SPS, which some versions do), so the time between receiving a sample from one load-cell and receiving a sample from the other load-cell can be as much as 0.1 second. This delay can be very detrimental to the performance of the adaptive filter; since the LMS algorithm tries to match the current input signal (samples) to the desired signal (samples), the noise in the load-cell signals needs to be as similar as possible at any moment in time.

Nevertheless, the measurements of the sideways (intended) movements of the paddle will always be slightly different as the load-cell is no longer centered relative to the axis of the paddle's rotation.

Conclusion: The placement of the dummy-load cell suffices; it measures similar noise and interference as the active load-cell. As the delay between the load-cells' measurements seems to originate from the low sampling rate of the load-cell amplifiers, the attachment of the dummy load-cell suffices as well.

Experiment 4.2a: Determining the optimal filter order and step size

Method: To determine the optimal filter order and step size, a desired signal is required from which the noise is to be removed. As discussed in Section 2.2.3 and Section 3.2.4, this desired signal is the measurement of the active load-cell, from which the noise is to be replicated by the adaptive filter with the measurement of the dummy load-cell. Hence, a force is to be applied at the handle, which serves as the clean signal that should result from the error between the active load-cell signal and noise estimate (Figure 2.14 and Table 4.1). For this experiment, the LMS filter is applied in Matlab to be able to compare different filter orders and step sizes for the exact same measurement.

First, the filter order and step size will be determined for measurements without applied force; this way, the force estimate should ideally be zero, which simplifies finding the optimal filter order and step size. Then, the optimal filter order and step size will be verified for filtering the measurements with force applied at the handle.

As stated in Table 4.2, the LMS adaptive filter is first optimised for short and intense disturbances where the force due to the inertia of the top of the load-cells predominates, like the disturbances in Figure 4.3. Also, it is compared to an optimised RLS filter to ensure it is not too "slow" to cancel these disturbances in the force estimate.

Next, to examine the effectiveness of the LMS filter for longer and less intense disturbances, the paddle is moved (without touching either of the load-cells) to measure the effect of gravity on the top of the load-cells. As a final measurement without force applied at the handle, the paddle is disconnected (like in Figure 3.20) and the most notable noise source measured in Figure 3.21 is to be removed by the LMS filter: the electromagnetic interference caused by the motor in the active (position-computed force) teleoperation system.

For every measurement and combination of order and step size (or forgetting factor for the RLS filter) listed in Table 4.2, the mean square error (MSE = system output power) is saved in a matrix. Since this error should be zero, the optimal filter order and step size are the ones that result in the lowest MSE in the matrix.

From these measurements without force applied at the handle, the optimal filter order and step size should be determined and verified for measurements with force applied at the handle. If the optimal values for these measurements do not correspond, the ones best suited for removing the effects of slow disturbances in the force estimate should be used; gravity (and inertia) is the most common significant disturbance in the active load-cell's measurements during teleoperation and especially noticeable when one of the setups is not interacted with and therefore should not measure any force.

Results: For short and intense disturbances, the optimal step size (μ) and filter order is $3 \cdot 10^{-5}$ and 8 (meaning 9 filter coefficients: M = 9) respectively, resulting in Figure 4.4.

Even with the optimal step size and filter order, the force estimate is far from completely zero, which it should ideally be as no force is applied at the handle. It does attenuate most of the large peaks (measured disturbances) in the desired signal, especially when they occur right after each other, which implies that the LMS algorithm is too "slow" to update its coefficients for the first peaks.

To verify this, the RLS function of Matlab (from the same Digital Signal Processing Toolbox as the LMS function) is used with the optimal filter order and forgetting factor found with Matlab. This results in Figure 4.5 with a filter order of 1 (M = 2) and forgetting factor (λ) of 1. While it does respond faster in terms of cancelling the first disturbances, the error signal is not much closer to zero than that of the LMS adaptive filter.

Finding the optimal filter parameters for slower and less intense disturbances (moving the paddle) with the same method as before results in an optimal step size of $1 \cdot 10^{-5}$ and optimal

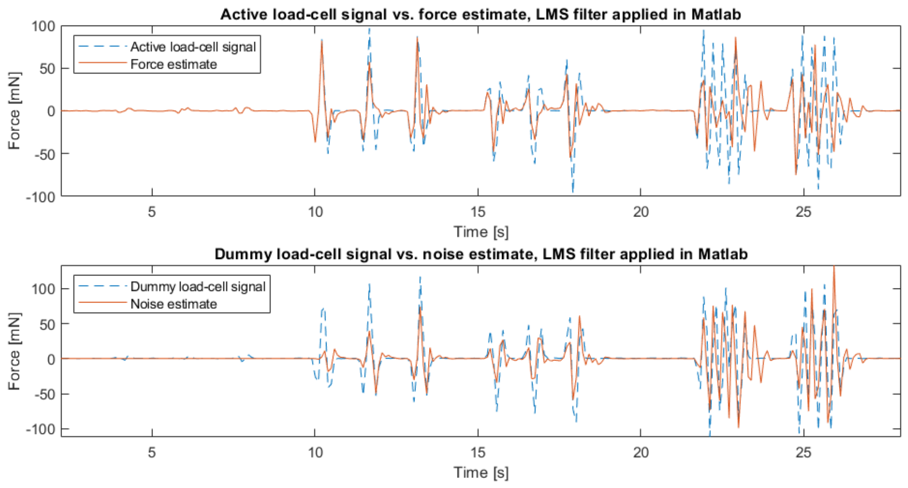


Figure 4.4: Using the LMS adaptive filter with $\mu = 3 \cdot 10^{-5}$ and $M = 9$ to filter out short and intense physical disturbances, conducted in the robotics lab.

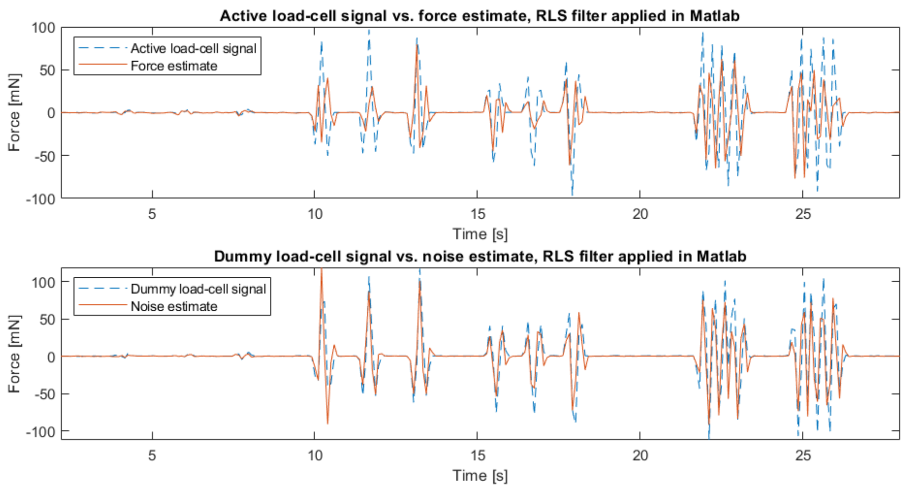


Figure 4.5: Using the RLS adaptive filter with $\lambda = 1$ and $M = 2$ to filter out short and intense physical disturbances, conducted in the robotics lab.

filter order of 1 (M = 2). The resulting force estimate can be seen in Figure 4.6, in which it can be observed that the LMS adaptive filter is much more effective at removing these "slower" disturbances; most of the disturbances in the active load-cell signal have been removed successfully, especially the slowest ones (widest peaks).

For the final measurement without force applied at the handle (electromagnetic interference during teleoperation), the optimal filter order and step size turned out to be 316 and $3 \cdot 10^{-3}$ respectively. The noise amplitude/power measured by the dummy load-cell is considerably less, but the adaptive filter compensates for this; it amplifies its input to remove as much of the (correlated) noise with its output.

Clearly, the optimal step sizes and filter orders do not match for the different types of noise and disturbances. Therefore, as mentioned before, the chosen filter order and step size are the optimal values for filtering out longer and less intense disturbances: 1 and $1 \cdot 10^{-5}$ respectively.

When verifying the adaptive filter with this filter order and step size for a measurement with force applied on the active load-cell, a problem occurs: the noise estimate also replicates some of the applied force, which starts at 20 s in Figure 4.8, where the teleoperation system was

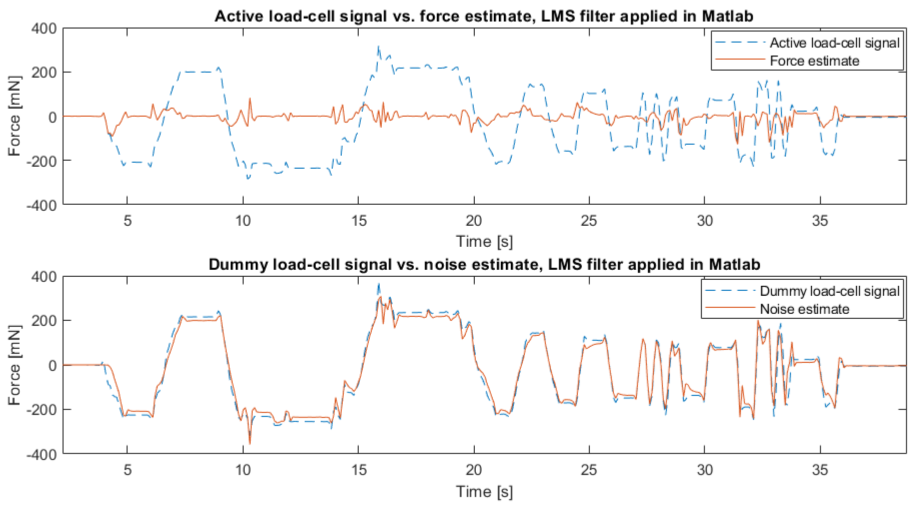


Figure 4.6: Using the LMS adaptive filter with $\mu = 1 \cdot 10^{-5}$ and $M = 2$ to filter out longer and less intense physical disturbances (movements of the paddle), conducted in the robotics lab.

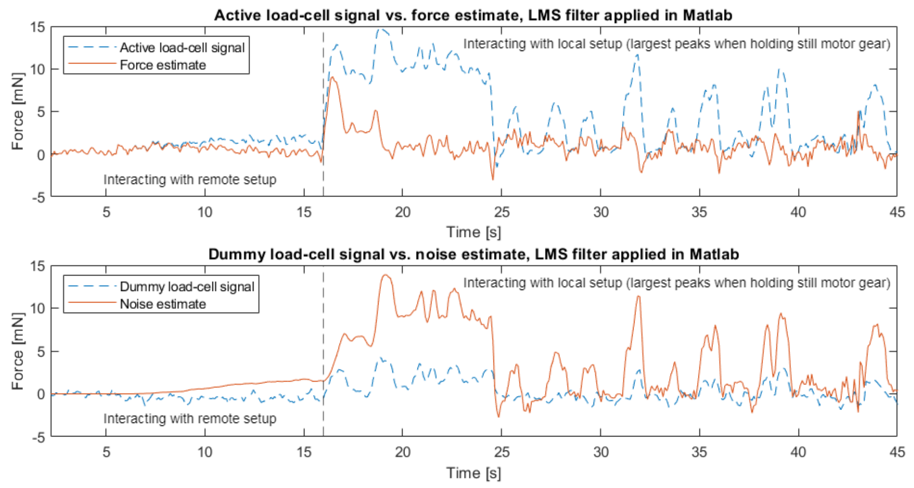


Figure 4.7: Using the LMS adaptive filter (on the "local" setup) with $\mu = 3 \cdot 10^{-3}$ and $M = 317$ to filter out the electromagnetic interference caused by the motor in the active teleoperation system, conducted in the robotics lab. The paddle containing the load-cells was disconnected to only measure the interference (and noise).

turned on. This means that the adaptive filter significantly distorts the clean signal (force estimate).

Discussion: For short and intense disturbances, the LMS algorithm seems to be too "slow" to change its filter coefficients for the first few disturbances. This suggests that the step size (convergence factor) might be too small, especially considering that the noise estimate does not replicate all of the noise (power) measured by the dummy load-cell, as seen in the bottom plot of Figure 4.4. However, for smaller step sizes the noise estimate rapidly diverges, which explains why this step size resulted in the least MSE. If short and intense disturbances are to be filtered out better, a faster and more computationally complex algorithm like RLS should be used. However, this RLS algorithm still failed to fully attenuate these disturbances. Since the noise estimate does replicate the noise measured by the dummy load-cell very well (as it should for no applied force at the handle), this sub-optimal performance of the LMS (and RLS) adaptive filter seems to be caused by the slightly delayed measurement of the dummy load-cell, found in Figure 4.3. The filter updates its coefficients live, so a delayed measurement requires it to change its input samples (dummy load-cell) much more to match the desired (active load-

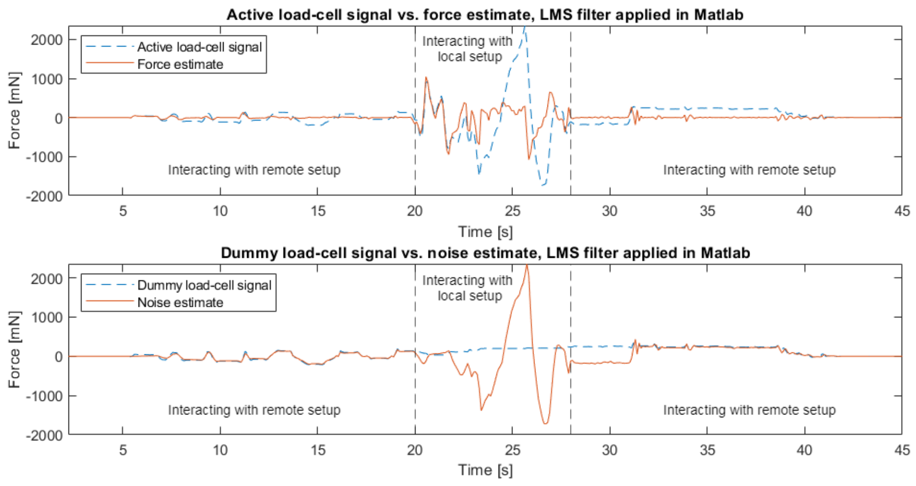


Figure 4.8: Using the LMS adaptive filter (on the "local" setup) with $\mu = 1 \cdot 10^{-5}$ and $M = 2$ to filter out noise and long physical disturbances (movements of the paddle) during position-computed force teleoperation, with force applied on the active load-cell between 20 [s] and 28 [s], conducted in the robotics lab.

cell) samples, which are taken at a slightly different time, especially in case of short and intense disturbances where the force is changing drastically in a short amount of samples.

The LMS adaptive filter does attenuate the electromagnetic interference during teleoperation in the force estimate, but it requires many filter coefficients to do so. This is a consequence of the dummy load-cell being attached further from the motor than the active load-cell, requiring the filter to amplify the dummy load-cell signal before subtracting it from the active load-cell signal. The filter's output (noise estimate) is effectively calculated as the sum of the element-wise multiplications of the filter coefficients with the (delayed) input signal (Equation 2.35). Hence, for an amplification of its input signal, more filter coefficients or a larger step size are required. Since Matlab finds the optimal filter order and step size without considering computational complexity, it opts for a high filter order to still allow for a high resolution (small changes) of the filter output due to the smaller step size.

The filter's attenuation of longer and less intense disturbances is very good, but one could argue that these disturbances (mainly gravity) could also be estimated and cancelled based on the setup's position. While this is possible, it requires the setups to always be turned on at the same motor angle; the motor angles and the measured force are reset to zero every time the system is turned on.

The distortion of the clean signal in Figure 4.8 indicates that the clean signal and signal measured by the dummy load-cell (caused mostly by gravity) are correlated enough for the adaptive filter to replicate the clean signal from its input (the dummy load-cell signal). As this distortion of the clean signal far outweighs the benefits of the adaptive filter in terms of teleoperation transparency, this needs to be solved.

Conclusion: While the LMS adaptive filter is not able to completely cancel short and intense disturbances, it can largely filter out "slower" disturbances such as gravity and inertia. Also, it is able to mostly filter out the electromagnetic interference during teleoperation, albeit with many filter coefficients. Nonetheless, the LMS filter (with $\mu = 10^{-5}$ and $M = 2$) should not be used in the DIY kit without solving the distortion of the clean signal: the estimate of the force applied at the handle.

Experiment 4.2b: Finding the optimal solution for the distortion of the force estimate

Method: To reduce the correlation between the dummy load-cell signal and clean signal (force at the handle), Principal Component Analysis (PCA) is applied before the LMS filter with the respective Matlab function. PCA essentially projects the measurements onto uncorrelated "principal components", which should contain most of the data's variance [70].

Alternatively, the LMS adaptive filter can be applied conditionally (referenced as a conditional LMS filter from now on). With this approach, the LMS filter will only be applied to the dummy load-cell signal if there is no force applied at the handle. When there is force applied at the handle, the filter is replaced by basic noise estimation: subtracting the dummy load-cell signal from the active load-cell signal. To determine if there is a force applied at the handle, a threshold will be used for the active load-cell signal. When exceeded, this will trigger a short timer that indicates how long the adaptive filter should be disabled (if the threshold is not exceeded again).

Results: As seen in Figure 4.9, PCA does help to reduce the distortion of the force estimate (between 20 and 28 s), but at the cost of an offset between the dummy load-cell signal and noise estimate. Consequently, this results in an offset of the force estimate when it should be zero. Also, the step size was greatly reduced to 10^{-10} to obtain this result.

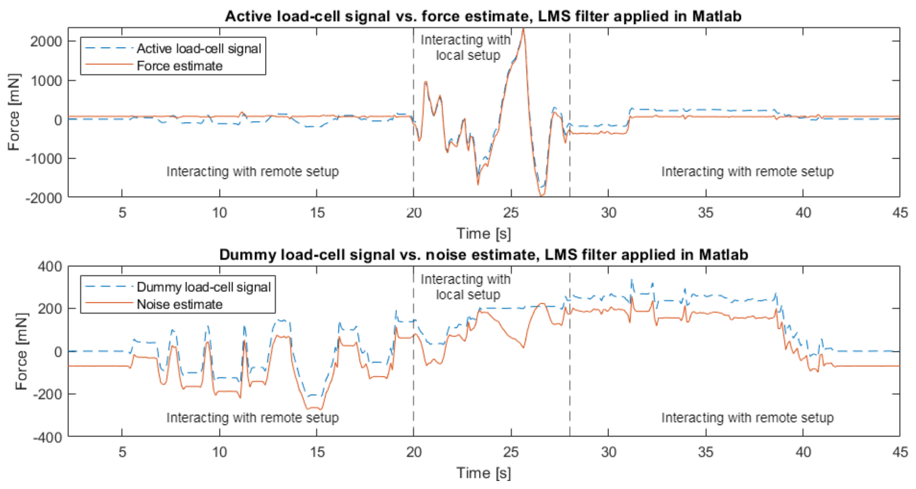


Figure 4.9: Using the LMS adaptive filter (on the "local" setup) with PCA (to reduce correlation) and $\mu = 1 \cdot 10^{-10}$ and $M = 2$ to filter out noise and long physical disturbances (movements of the paddle) during position-computed force teleoperation, with force applied on the active load-cell between 20 [s] and 28 [s], conducted in the robotics lab.

Applying the LMS filter conditionally with a threshold of 450 mN and timer of 1 s results in Figure 4.10. This conditional LMS filter seems to perform well, even with the strong deviation from the dummy load-cell signal between 28 and 31 s. During this time frame, no (intentional) force was applied at the active load-cell, but it seems to measure a negative disturbance which the dummy load-cell does not pick up.

Discussion: While PCA does almost completely prevent the distortion of the clean signal, this comes at the cost of an offset in the force estimate when it should be zero. Furthermore, using PCA before the LMS filter requires significantly more of the limited computational resources of the Arduino.

Using the LMS adaptive filter conditionally provides good results, even with the discrepancy between the load-cells' measurements from 28 to 31 s. Since there was no intentional applied force at the handle in this time frame, it must have been a minor accidental disturbance of the

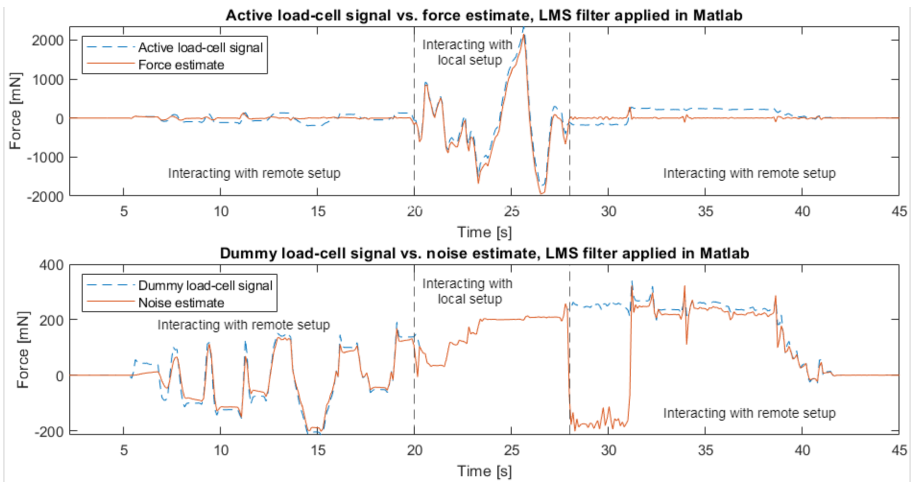


Figure 4.10: Using the LMS adaptive filter (on the "local" setup) conditionally with $\mu = 1 \cdot 10^{-5}$ and $M = 2$ to filter out noise and long physical disturbances (movements of the paddle) during position-computed force teleoperation with force applied on the active load-cell between 20 [s] and 28 [s], conducted in the robotics lab. When this force exceeds 450 mN, the timer that disables the filter is reset to 1 s.

handle. Nonetheless, the LMS filter still cancels this disturbance completely by deviating from the dummy load-cell signal.

Conclusion: Considering the offset of the force estimate and the increased computational complexity when using Principal Component Analysis, it was concluded that applying the LMS adaptive filter conditionally (referenced as a conditional LMS filter from now on) is the best solution to prevent distortion of the force estimate.

Experiment 4.3: Determining the optimal conditional LMS filter parameters on Arduino

Method: Having determined that a conditional LMS filter is the best option, it is implemented on the Arduino Uno to cancel the noise and disturbances (including gravity and inertia) in real-time. Since the screws and handle are removed to place the dummy load-cell closer to the active load-cell and reduce the delay between their measurements as much as possible (Figure 4.11), the weight of the top part is slightly different than that of the active load-cell. However, the adaptive filter should compensate for this as a difference in amplitude should not greatly decrease the correlation between the load-cells' measured disturbances due to inertia and gravity.

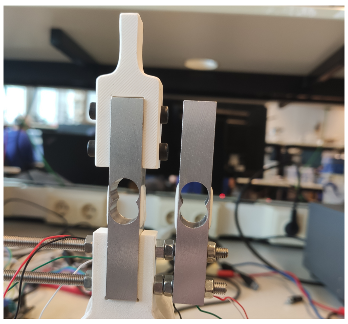


Figure 4.11: Revised dummy load-cell attachment used in experiments 4.3, 5.1, 5.4 and 5.5 to evaluate the conditional LMS filter running in real-time on the Arduino Uno.

While the pseudocode for the LMS algorithm in Section 3.3.2 is still mostly correct, it does not include disabling the LMS filter when a force is applied at the handle. To include this, the pseudocode for the LMS function should be updated as follows:

```
float LMS {
    read current input signal (dummy load-cell)
    read current desired output signal (active load-cell)
    calculate load-cell forces in newtons
    if (threshold is exceeded by active-load cell force or the timer is still active){
        calculate the error by subtracting the dummy load-cell force from the active load-cell force
        if (threshold is exceeded by active-load cell force){
            reset the timer to the duration for which the LMS filter should remain disabled
        }
    } else {
        shift saved forces by one delay, discard the oldest
        save current force
        calculate output with filter coefficients & saved forces:
        calculate error between current & desired output signals
        calculate next filter coefficients with the error, convergence factor & saved forces
    }
    return the error as system output: the force estimate
}
```

To set the threshold for disabling the LMS filter, the maximum measured force without any force at the handle is determined by moving the paddle back and forth as fast as possible multiple times in a row

To determine the duration for which the LMS filter should be disabled when this threshold is exceeded, the setup on the task side is interacted with first and then the setup on the human side is interacted with (via the handle), just like in Figures 4.8, 4.9 and 4.10. This duration should be sufficient to ensure the LMS filter is not enabled while the handle is still being interacted with, yet brief enough to prevent the LMS filter from being disabled when no force is applied to the handle.

To verify that the order of 1 and step size of 10^{-5} are the optimal values for the implementation on the Arduino Uno, the paddle is first moved slowly and then quickly, without applying any force at the handle (as the LMS filter should then be disabled), for slightly different orders and step sizes.

Results: The (active) load-cell's measurement when moving the paddle back and forth as quickly as possible can be seen in Figure 4.12. To ensure that the LMS filter is always enabled when no force is applied at the handle, the threshold is set to 430 mN for both directions (positive and negative force in Figure 4.12).

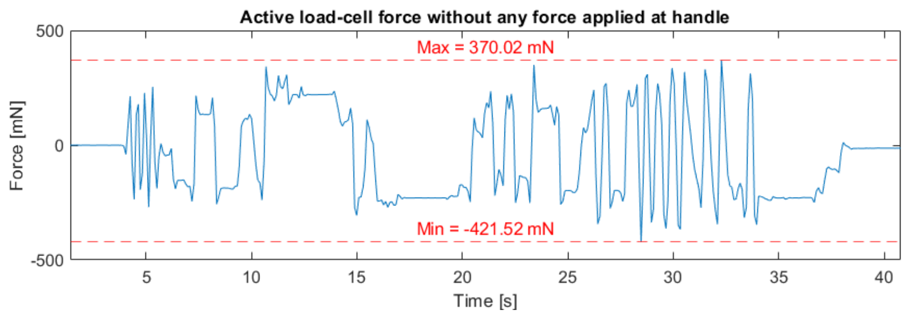


Figure 4.12: Moving the paddle back and forth as fast as possible to determine the maximum force that can be measured without force applied at the handle, conducted in the robotics lab.

To determine the duration for which the LMS filter should remain disabled when the threshold is exceeded, the local setup (with the dummy load-cell and LMS filter) is interacted with from 20 s in Figure 4.13. A duration of 0.1 s is found to be the minimum; for durations shorter than that, the LMS filter is occasionally enabled when the force applied at the handle changes direction, as illustrated by the narrow peaks (such as those at roughly 22 s and 27 s) for a duration of 0.09 s in Figure 4.13.

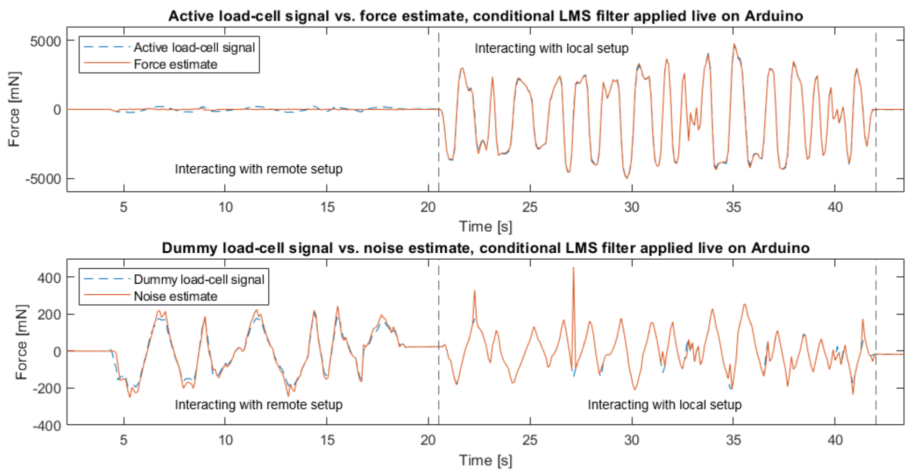


Figure 4.13: Using the LMS adaptive filter (on the "local" setup) conditionally with $\mu = 1\cdot10^{-5}$ and $M = 2$ to filter out noise and long physical disturbances (movements of the paddle) during position-computed force teleoperation, with force applied on the active load-cell between 21 [s] and 42 [s], conducted in the robotics lab. The timer that disables the LMS filter is set to 0.09 s and the threshold to 430 mN.

The results with slightly different step sizes and filter orders can be seen in Figures 4.14 and 4.15. With half the step size ($5\cdot10^{-6}$) or twice the order (2), the filter requires more time to attenuate the disturbances in the force estimate and does so with more oscillation than with the optimal values. When the step size is twice or the order is three times the optimal value, this oscillation in the force estimate becomes much worse.

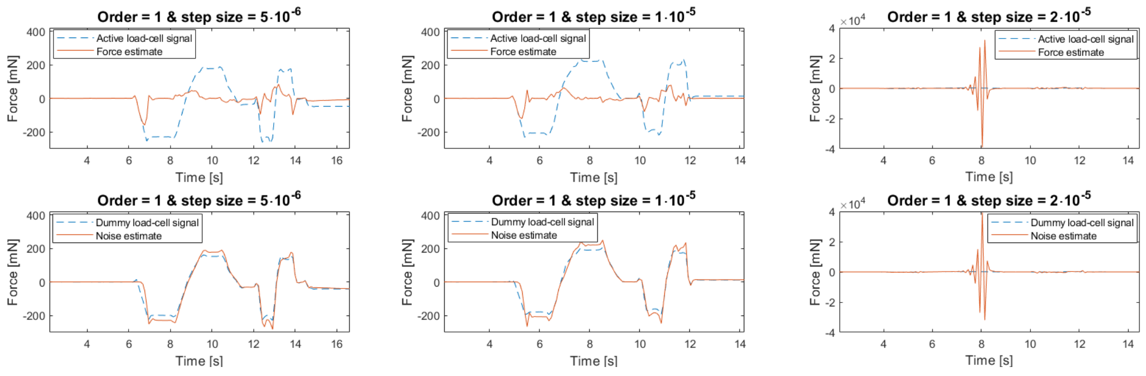


Figure 4.14: Moving the paddle back and forth, first quickly and then slowly, with different step sizes to determine the optimal value, conducted in the robotics lab.

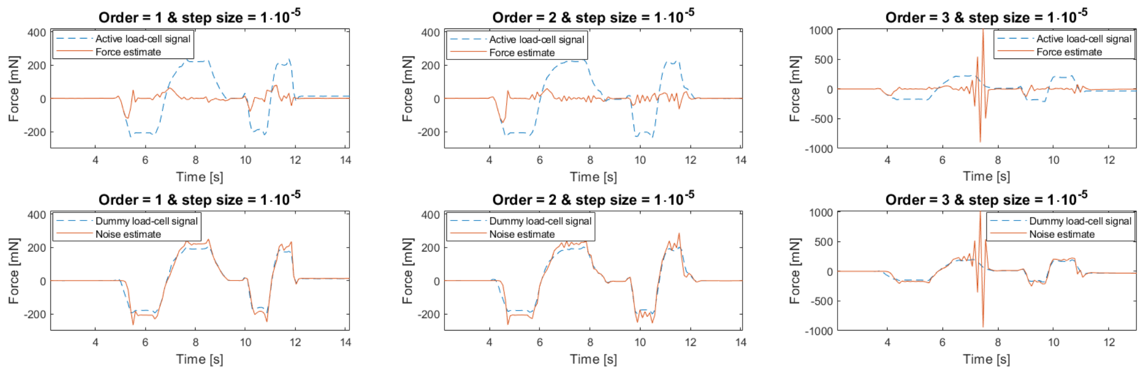


Figure 4.15: Moving the paddle back and forth, first quickly and then slowly, with different filter orders to determine the optimal value, conducted in the robotics lab.

It is worth noting that in these measurements (Figures 4.14 and 4.15), the dummy load-cell signal needs to be amplified slightly to cancel the disturbances measured by the active load-cell, while this was not the case previously: Figure 4.6.

Discussion: From Figure 4.14, 10^{-5} indeed seems to be the optimal value for a filter order of 1; a slightly larger step size results in a severely oscillating force estimate (since it overshoots the optimal filter coefficients), and a slightly smaller step size does not remove the disturbances from the force estimate as quickly (as it does not converge to the optimal coefficients quickly). As for the filter order, 1 seems to be the optimal value as well (for a step size of 10^{-5}). As seen in Figure 4.15, a filter order of 2 results in minor oscillation in the force estimate and a filter order of 3 results in severe oscillation (instability).

The observation that the filter has to amplify the dummy load-cell signal slightly to cancel the disturbances in the active load-cell signal shows that the handle has been removed from the dummy load-cell; due to the decreased weight, it is affected less by gravity and inertia (the main disturbances in this measurement). When the handle was still attached to the dummy load-cell, this was not the case (Figure 4.6).

Conclusion: The LMS filter should be disabled for at least 0.1 s on the Arduino Uno('s) whenever the active load-cell signal exceeds the threshold of 430 mN. When the filter is disabled, the dummy load-cell signal should simply be subtracted from the active load-cell signal. To attenuate slow disturbances (such as gravity) as much as possible, the LMS algorithm should use a step size of 10^{-5} and the filter order should be 1.

4.1.4 Final implementation during teleoperation

Since the current attachment and placement of the dummy load-cell (Figure 4.11) provides sufficient results for cancelling the gravity (and some inertia) in the force measurement, this will be the definitive attachment and placement of the dummy load-cell in both setups.

As mentioned before, the HX711 amplifier has a maximum sampling rate of 10 SPS, which is very problematic for implementation in the (provided) teleoperation code; this code runs at high frequencies (in this case 500 Hz) for optimal transparency. So, to avoid slowing down this teleoperation code to 10 Hz, the measured force values should be reused multiple times. To make sure the loop does not have to wait for a new force sample and only requests one when it is available, the respective *is_ready()* function of the HX711 library [66] is used. Only when this function returns "true", a new force sample is requested from the load-cell's amplifier, which is reused until the next sample is available. Nonetheless, this will harm the adaptive filter's performance and the system's transparency and stability.

4.1.5 Modifying the load-cell amplifiers to increase their sampling rate

The HX711 amplifier seems to be the only cost-effective choice for a load-cell amplifier; while there are cost-effective instrumentation amplifiers with a higher bandwidth (like the INA125P), they are not available on breakout boards, which is a requirement for easy assembly of the DIY kit. Additionally, the HX711 breakout board contains a 24-bit ADC, which would otherwise need to be sourced separately and soldered onto a breakout board as well. Alternatively, the ADC of the Arduino Uno could be used, but this has a much lower resolution (10-bit [51]).

However, the HX711 breakout board can be modified to provide a higher sampling rate of 80 SPS. While the breakout board chosen for this DIY kit requires soldering for this modification, there are breakout boards (e.g. from SparkFun [71]) with a dedicated pin to select between 10 SPS and 80 SPS, which would be better suited for implementation in a DIY kit.

Therefore, after the results with 10 force samples per second have been obtained, the load-cell amplifiers are modified to provide 80 SPS and the performance is reassessed in experiment 5.5. This modification is quite straight-forward: the "RATE" pin of the HX711 chip is currently soldered to the ground (0 V) of the breakout board, while it should be connected to the supply voltage (5 V) to select the higher sampling rate [65].

4.2 Control architectures

4.2.1 Position-measured force architecture

To implement the position-measured force architecture with torque control and the conditional LMS filter, the main loops of the pseudocode of Section 3.3.2 are updated with the conditional LMS filter and the statement to only request a new force sample when it is available:

Pseudocode for Arduino on human side

```
void loop{
    calculate loop time of previous loop
    while (previous loop time < desired loop time){
        wait until the desired loop time is reached
    }
    if (previous loop time > desired loop time){
        indicate that the desired loop time is exceeded
    }
    read local angle
    local steer = measured steer
    if (local steer == 0){
        if (moving backwards){
            output PWM = local steer - correction
        } else if (moving forwards){
            output PWM = local steer + correction
        } else {
            output PWM = local steer
        }
    } else if (local steer < 0){
        output PWM = local steer - correction
    } else if (local steer > 0){
        output PWM = local steer + correction
    }
    previous angle = current angle
    determine direction
    ensure output PWM is within bounds
    motor PWM = output PWM
}
```

Pseudocode for Arduino on task side

```
void loop{
    calculate loop time of previous loop
    while (previous loop time < desired loop time){
        wait until the desired loop time is reached
    }
    if (previous loop time > desired loop time){
        indicate that the desired loop time is exceeded
    }
    if (new force sample available){
        read and update measured force
    }
    LMS filter measured force
    calculate measured steer: filtered force / K_fd
    send message with measured steer
    request message with remote angle
    read local angle
    error = remote angle - local angle (in radians)
    computed steer = Kp*error
    if (computed steer == 0){
        if (moving backwards){
            output PWM = computed steer - correction
        } else if (moving forwards){
            output PWM = computed steer + correction
        } else {
            output PWM = computed steer
        }
    } else if (computed steer < 0){
        output PWM = computed steer - correction
    } else if (computed steer > 0){
        output PWM = computed steer + correction
    }
    previous angle = current angle
    determine direction
    ensure output PWM is within bounds
    motor PWM = output PWM
}
float LMS {
    read current input signal (dummy load-cell)
    read current desired output signal (active load-cell)
    calculate load-cell forces in newtons
    if (threshold is exceeded by active-load cell force or the
      timer is still active){
        calculate the error by subtracting the dummy load-cell
      force from the active load-cell force
        if (threshold is exceeded by active-load cell force){
            reset the timer to the duration for which the LMS
      filter should remain disabled
        }
    } else {
        shift saved forces by one delay, discard the oldest
        save current force
        calculate output with filter coefficients & saved
      forces:
        calculate error between current & desired output signals
        calculate next filter coefficients with the error,
      convergence factor & saved forces
    }
    return the error as system output: the force estimate
}
```

Furthermore, one of the setups occasionally jumped to another position (motor angle) when the system was just turned on. This seemed to originate from the encoder, which sometimes outputs random values just after it has been reset to zero (and the motor angle has not been changed yet). Following the advice from a fellow student working on this DIY kit (Jirne Snijders), this issue has been circumvented by ensuring that the encoder value is equal to zero three times in a row (in the setup) before entering the main loop.

4.2.2 4-channel architecture

The code to implement the 4-channel architecture in the DIY kit has not been completed or tested in this thesis, but it should be based on the pseudocode from Section 3.3.3 with the same changes made to the main loops:

Pseudocode for Arduino on human side

```
void loop{
    calculate loop time of previous loop
    while (previous loop time < desired loop time){
        wait until the desired loop time is reached
    }
    if (previous loop time > desired loop time){
        indicate that the desired loop time is exceeded
    }
    read current (local) angle
    if (new force sample available){
        read and update measured force
    }
    rotational velocity =
        (current angle - previous angle) / loop time
    LMS filter measured force
    task steer 1 = Kp_1 * rotational velocity
    task steer 2 = Kp_3 * filtered force / K_fd
    local steer = human steer 1 + human steer 2
    if (local steer == 0){
        if (moving backwards){
            output PWM = local steer - correction
        } else if (moving forwards){
            output PWM = local steer + correction
        } else {
            output PWM = local steer
        }
    } else if (local steer < 0){
        output PWM = local steer - correction
    } else if (local steer > 0){
        output PWM = local steer + correction
    }
    previous angle = current angle
    determine direction
    ensure output PWM is within bounds
    motor PWM = output PWM
}
float LMS {
    read current input signal (dummy load-cell)
    read current desired output signal (active load-cell)
    calculate load-cell forces in newtons
    if (threshold is exceeded by active-load cell force or the
      timer is still active){
        calculate the error by subtracting the dummy load-cell
      force from the active load-cell force
        if (threshold is exceeded by active-load cell force){
            reset the timer to the duration for which the LMS
      filter should remain disabled
        }
    } else {
        shift saved forces by one delay, discard the oldest
        save current force
        calculate output with filter coefficients & saved
      forces:
        calculate error between current & desired output signals
        calculate next filter coefficients with the error,
      convergence factor & saved forces
    }
    return the error as system output: the force estimate
}
```

Pseudocode for Arduino on task side

```
void loop{
    calculate loop time of previous loop
    while (previous loop time < desired loop time){
        wait until the desired loop time is reached
    }
    if (previous loop time > desired loop time){
        indicate that the desired loop time is exceeded
    }
    read current (local) angle
    if (new force sample available){
        read and update measured force
    }
    rotational velocity =
        (current angle - previous angle) / loop time
    LMS filter measured force
    human steer 1 = Kp_4 * rotational velocity
    human steer 2 = Kp_2 * filtered force / K_fd
    send message with human steer 1 & 2
    request message with task steer 1 & 2
    local steer = task steer 1 + task steer 2
    if (local steer == 0){
        if (moving backwards){
            output PWM = local steer - correction
        } else if (moving forwards){
            output PWM = local steer + correction
        } else {
            output PWM = local steer
        }
    } else if (local steer < 0){
        output PWM = local steer - correction
    } else if (local steer > 0){
        output PWM = local steer + correction
    }
    previous angle = current angle
    determine direction
    ensure output PWM is within bounds
    motor PWM = output PWM
}
float LMS {
    read current input signal (dummy load-cell)
    read current desired output signal (active load-cell)
    calculate load-cell forces in newtons
    if (threshold is exceeded by active-load cell force or the
      timer is still active){
        calculate the error by subtracting the dummy load-cell
      force from the active load-cell force
        if (threshold is exceeded by active-load cell force){
            reset the timer to the duration for which the LMS
      filter should remain disabled
        }
    } else {
        shift saved forces by one delay, discard the oldest
        save current force
        calculate output with filter coefficients & saved
      forces:
        calculate error between current & desired output signals
        calculate next filter coefficients with the error,
      convergence factor & saved forces
    }
    return the error as system output: the force estimate
}
```

4.2.3 Experiments

In experiment 4.4, the optimal controller gains (K_p) for both architectures are determined to ensure a fair comparison of the performances of the position-measured force and position-computed force architectures. To check if the provided position-computed force code already uses the optimal proportional gain ($K_p = 8$), the measurements are performed for two slightly higher gains (Table 4.3) to observe if oscillation or even instability starts to occur. The position-measured force architecture should be able to use a higher proportional gain, as the controller only operates locally on the task side and is therefore not influenced by the communication delays between the setups, as seen in Figure 2.6. Hence, higher proportional gains are tested for this architecture, listed in Table 4.3.

Furthermore, the amplifiers are modified to provide 80 SPS, as explained in Section 4.1.5. While the system's transparency and the conditional LMS filter's performance is evaluated in

Chapter 5, the stability of the system and the effects of the amplifiers' modification are evaluated in experiment 4.5. Because of the 8 times smaller delay between force measurements, the stability should be improved over the system's stability with the standard amplifiers, which is found to be sub-optimal; when the both setups' handles are interacted with by a single operator, the setups start to become difficult to manage due to increasing (synchronised) oscillation. When the operator tries to correct for this by holding still both handles, the system occasionally became unstable.

To observe the effects of modifying a (HX711) load-cell amplifier to 80 SPS, the dummy load-cell's amplifier is modified first. According to the amplifier's datasheet, this should result in a higher (measured) noise power [65]. When both amplifiers are modified to provide 80 SPS, the delay between the load-cells' measurements is reviewed to verify the hypothesis (from experiment 4.1b) that this delay originates from the low sampling rate.

Table 4.3: Overview of the experiments conducted for the implementation of the position-measured force architecture in Chapter 4.

Experiment	Objective	Tested configurations
4.4	Determining the optimal controller gains for the position-measured force and position-computed force architectures	Position-computed force: K_p = 8 -16 (in steps of 4)
		Position-measured force: K_p = 8 - 32 (in steps of 4)
4.5	Evaluating the system's stability with the modified amplifiers	Only dummy load-cell's amplifier modified to provide 80 SPS
		Both load-cells' amplifiers modified to provide 80 SPS

Experiment 4.4: Determining the optimal controller gains for the position-measured force and position-computed force architectures

Method: Ideally, a proportional position controller's gain is as high as possible without causing oscillation when the desired position is suddenly changed. In robotics, this controller gain is usually verified with its step response; the controller's response to an instantaneous change of the desired position to a (very) different value. However, this is not realistic in teleoperation as the unknown operator's hand and environment impedances prevent the positions from changing instantaneously. Hence, to find the optimal proportional gain for each architecture (out of the ones listed in Table 4.3), the setup on the human side is interacted with at different intensities: first, the paddle is moved back and forth slowly, then it is moved as quickly as possible to different positions, and finally it is moved back and forth smoothly and with a constant frequency multiple times in a row.

To ensure no oscillation other than the oscillation caused by the position controller is present in the measurements, both setups are fixed in place with clamps: Figure 4.16.

Because of the data collection through serial communication, the loop frequency is lowered to 300 Hz to reduce the workload of the Arduino's and thereby avoid sudden jumps in positions (motor angles) between the setups. This lower loop frequency does not decrease the performance in terms of (visual) position tracking or (felt) force tracking when compared to the loop frequency of 500 Hz.

Results: In Figure 4.17, the position tracking performance of the position-computed force architecture for different controller gains can be found. At K_p = 16, the system is unstable: the angle diverges almost immediately after turning it on. For K_p = 12, the system is not yet unstable, but there is more oscillation after the sudden position changes from 9 to 11 s. However, the position tracking performance for slow movements (up to 9 s) is slightly better for this

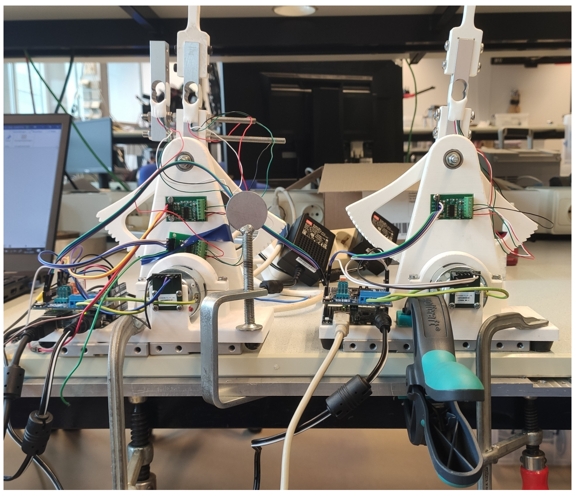


Figure 4.16: Measurement setup used in experiment 4.4 to determine the optimal proportional gains (K_p) for the position-measured force and position-computed force architectures.

higher gain. The position tracking of an oscillatory signal with constant frequency (from 11 s onwards) does not seem to improve notably for a higher control gain.

In Figure 4.18, the position tracking performance of the position-measured force architecture with different controller gains can be observed. Up to $K_p = 16$, the oscillation after sudden position changes is very minimal. Starting at $K_p = 20$, this oscillation becomes more pronounced and persists for more periods before being completely damped out. Notably, the system does not become unstable for these gains, even at $K_p = 32$, which is 2 times higher than the gain for which the system with the position-computed force architecture becomes unstable. Just like with the position-computed force architecture, a higher proportional gain results in slightly better low-speed position tracking, but not better tracking of the oscillatory signal.

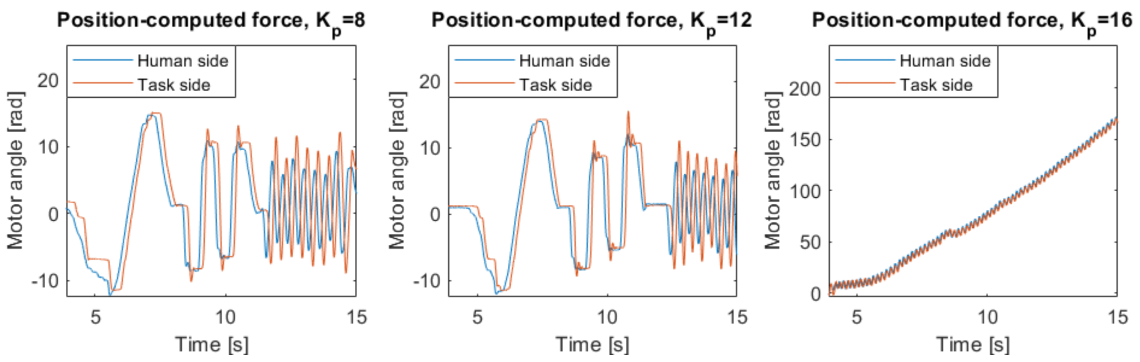


Figure 4.17: Position tracking using the position-computed force architecture with different controller gains K_p.

Discussion: Judging from Figure 4.17, it seems that the proportional gain of 8 that already used in the provided code, is close to the maximum gain that the position-computed force architecture allows without significant oscillation after quick movements of the paddle.

For the position-measured force architecture, this optimal proportional gain seems to be either 16 or 20, offering the best (low-speed) position tracking without significant oscillation after quick movements. While the oscillation for $K_p = 20$ is noticeable in the measurement, it was not clearly visible in the paddle's movement. As these quick and sudden movements without any force on the task side are not likely to occur often, a proportional gain of 20 is chosen.

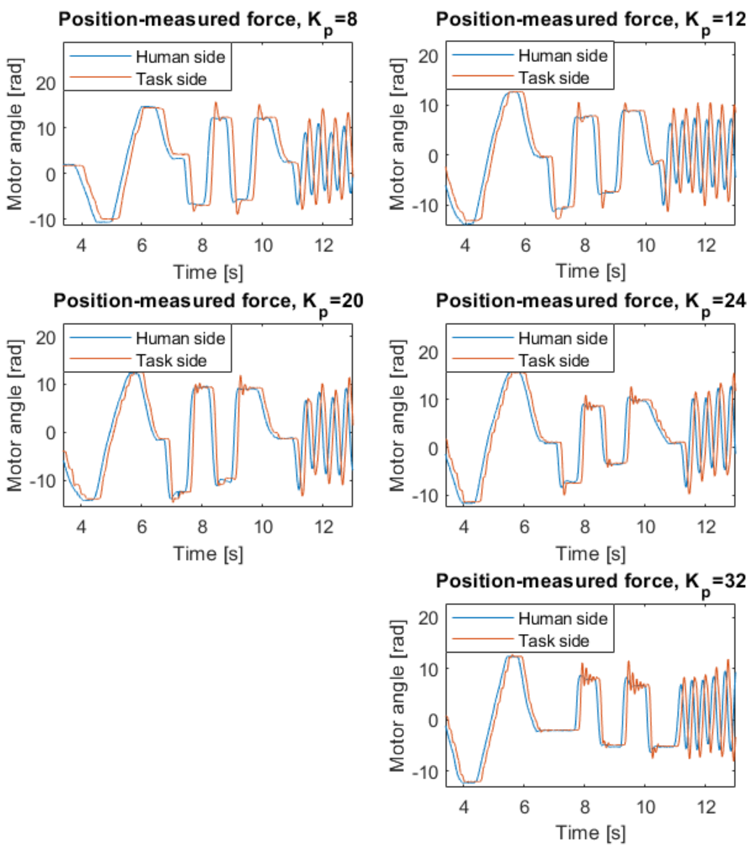


Figure 4.18: Position tracking using the position-measured force architecture with different controller gains K_p.

The similar performance in tracking an oscillatory signal for all proportional gains is a consequence of the inertia of the paddle, which is difficult to overcome for this relatively quickly changing signal.

Conclusion: For the position-computed force architecture, the proportional gain that is already used in the provided code is (close to) the optimal gain for the position controller in this teleoperation system: $K_p = 8$. For the position-measured force architecture, the optimal gain is much higher due to the controller only operating locally: $K_p = 20$.

Experiment 4.5: Evaluating the stability of the position-measured force architecture with the modified load-cell amplifiers

Method: To modify the load-cell amplifiers (Figure 4.20), the "RATE" pin (15 in Figure 4.19) is cut and separated from the breakout board, after which it is soldered to the "DVDD" pin (16 in Figure 4.19) to select the higher sampling rate of 80 samples per second: Figure 4.21.

First, only the dummy load-cell's amplifier is modified to validate that the conversion was successful and evaluate the difference in measured noise. Then, the active load-cell's amplifier is modified to verify that the delay between the load-cells' measurement is reduced and the stability of the system is increased.

Results: With only the dummy load-cell's amplifier modified to 80 samples per second (SPS), the noise measured in the robotics lab can be found in Figure 4.22.

The conversion was successful, as there are 8 dummy load-cell samples for each active load-cell sample. However, the noise power measured by the dummy load-cell at 80 SPS is also (roughly)

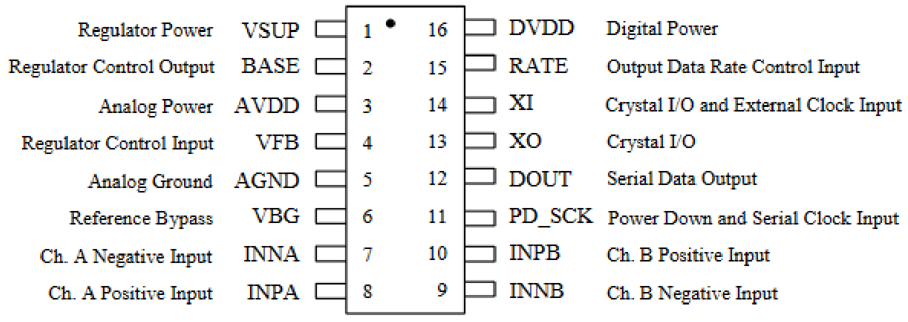

Regulator Power VSUP 1
Regulator Control Output BASE 2
Analog Power AVDD 3
Regulator Control Input VFB 4
Analog Ground AGND 5
Reference Bypass VBG 6
Ch. A Negative Input INNA 7
Ch. A Positive Input INPA 8
16 DVDD Digital Power
15 RATE Output Data Rate Control Input
14 XI Crystal I/O and External Clock Input
13 XO Crystal I/O
12 DOUT Serial Data Output
11 PD_SCK Power Down and Serial Clock Input
10 INPB Ch. B Positive Input
9 INNB Ch. B Negative Input

Figure 4.19: Layout of the HX711 chip used in the breakout board of the load-cell amplifier.

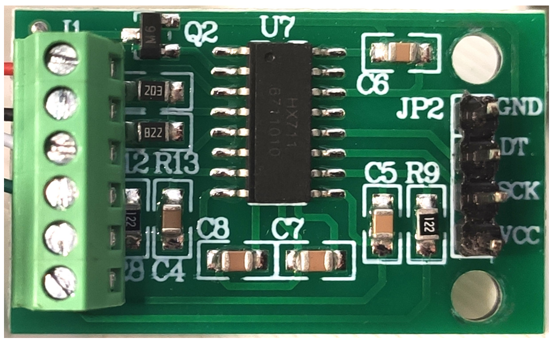


Figure 4.20: HX711 load-cell amplifier, providing 10 samples per second.

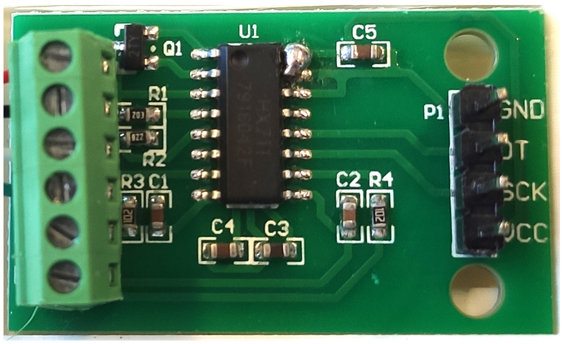


Figure 4.21: HX711 load-cell amplifier, modified to provide 80 samples per second.

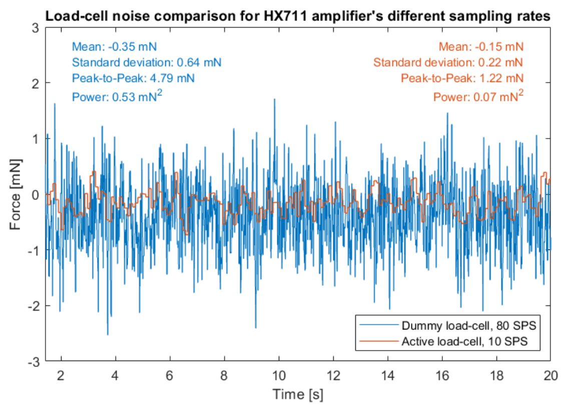


Figure 4.22: The noise measured by the load-cells with different HX711 sampling rates (of 10 and 80 SPS), conducted in the robotics lab.

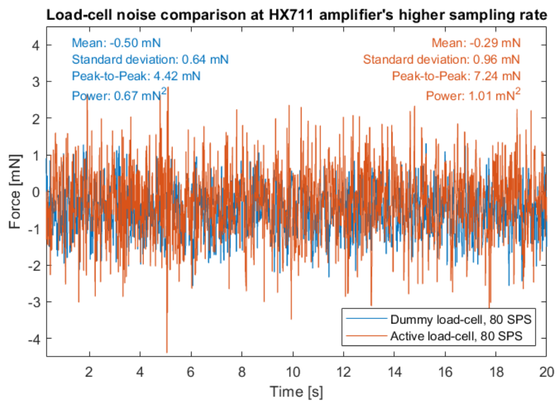


Figure 4.23: The noise measured by the load-cells with the same HX711 sampling rate (of 80 SPS), conducted in the robotics lab.

8 times higher than the noise power measured by the active load-cell at 10 SPS. Furthermore, the mean of the dummy load-cell's measurement deviates further away from zero. When the active load-cell's amplifier is modified as well, its noise power becomes even higher than that of the dummy load-cell: Figure 4.23.

Figure 4.24 shows the force measurements for short and intense physical disturbances of the setup with both amplifiers modified to 80 SPS. The delays between the measured disturbances are approximately 0.01 s with this increased sampling rate. As discussed at the start of experiment 4.3, the observed difference in amplitudes can be attributed to the absence of the handle and screws on the dummy load-cell, reducing the effects of inertia and gravity.

Discussion: According to the HX711 amplifier's datasheet [65], an increase in noise power at 80 SPS is to be expected: Figure 4.25.

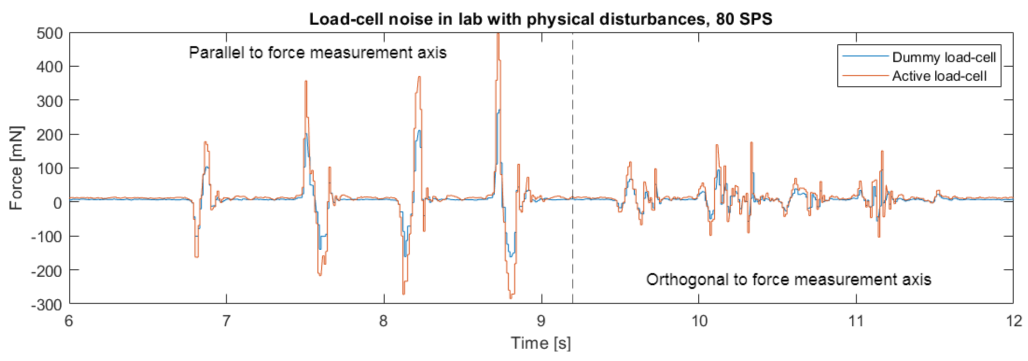


Figure 4.24: Load-cells' measurements at 80 SPS in case of disturbances parallel and orthogonal to the force measurement axis, conducted in the robotics lab.

Input noise	Gain = 128， RATE = 0	50	nV(rms)
	Gain = 128， RATE = DVDD	90	

Figure 4.25: Specified increase in measured (input) noise when the HX711 load-cell amplifier is converted to 80 SPS [65].

When converted to $\mathrm{mN_{rms}}$ with the root of the (mean-square) powers in Figure 4.22, the measurements show an increase from 0.26 $\mathrm{mN_{rms}}$ with 10 SPS to 0.73 $\mathrm{mN_{rms}}$ with 80 SPS. Since the measured force is directly related to the amplifier's (output) voltage, the relative increases may be compared:

$$\frac{90\,\mathrm{nV_{rms}} - 50\,\mathrm{nV_{rms}}}{50\,\mathrm{nV_{rms}}} \cdot 100\% = 80\% \tag{4.1}$$

$$\frac{0.73\,\mathrm{mN_{rms}} - 0.26\,\mathrm{mN_{rms}}}{0.26\,\mathrm{mN_{rms}}} \cdot 100\% \approx 180.8\% \tag{4.2}$$

Clearly, the increase in noise (power) measured in the robotics lab is significantly higher than specified in the datasheet.

The delay between the load-cells' measurements is reduced by roughly the same factor as the increase in sampling rate (8) compared to the delays measured in Figure 4.3. This supports the hypothesis that (most of) this delay is a consequence of the load-cell amplifiers' sampling rates; with the increased sampling rate, the delay can be as much as $\frac{1}{80} = 0.0125$ s, already slightly exceeding the measured delays of approximately 0.01 s.

Finally, the stability of the system is improved with the modified amplifiers; the setups can now be kept under control (with some slight oscillation) when controlled by a single operator and the system no longer becomes unstable. This is a direct consequence of the reduced delay between force samples, as the operator feels any changes in measured force in the task environment sooner and more frequently, and can therefore better react to them.

Conclusion: When modifying the load-cell amplifiers to provide 80 SPS, the stability of the teleoperation system improves and the delay between the load-cells' measurements decreases. However, the noise power in the force measurements significantly increases.

5 Evaluation

In this chapter, the performance of the conditional LMS filter is evaluated and compared to a permanent implementation of an LMS adaptive filter in experiment 5.1. Then, experiment 5.2 is conducted to compare the performance of the optimised position-measured force architecture (using the conditional LMS filter) with the performance of the provided position-computed force architecture in terms of force and position tracking for various teleoperation scenarios, listed in Table 5.1. Since testing time was limited and the performance optimisation for added time delays in the communication channels is out of the scope of this thesis, the performance for the system without added time delays is evaluated.

Because the $K_{f,d}$ of 0.0714 N was proven to be very temperature dependent (Figure 3.12) and did not meet expectations in terms of force tracking, these force and position tracking measurements are performed once more with a different $K_{f,d}$. To achieve the optimal transparency for this specific DIY kit, the optimal $K_{f,d}$ was found to be roughly 0.1 N. This implies that the setups' temperatures were closer to the ones in the "cold" measurements in Figure 3.12 than the ones in the "warm" measurements, from which the $K_{f,d}$ has been determined. This is a reasonable assumption, as the setups were not "warmed up" with the exact same procedure because of the restricted time for the final experiments.

To verify that the conditional LMS filter performs similarly to previous experiments when implemented in the position-measured force architecture, several tests are performed in experiment 5.4, as stated in Table 5.1.

Finally, experiment 5.5 evaluates the effects of modifying the load-cell's amplifiers (to provide 80 SPS) on the force/position tracking and (conditional) adaptive filter's performance. To verify that the increased noise in the force measurement originates from the modification of the amplifiers, the load-cells and amplifiers have been interchanged between setups in multiple ways, but this did not present any notable difference.

5.1 Experiments

Experiment 5.1: Verifying the conditional LMS filter on Arduino

Method: With the optimal filter parameters determined in experiment 4.3, the conditional LMS filter is compared to the permanent LMS filter. First, their performance is compared during teleoperation, interacting with the remote setup (without adaptive filter) and then with the local setup (with adaptive filter), such that the LMS filter should be disabled during the measurement.

Since electromagnetic interference during teleoperation was quite noticeable in the load-cell's measurement (Figure 4.7), it is important that the filter cancels this in the force estimate. So, just like for Figure 4.7, the paddle is disconnected to only measure this interference during teleoperation. First, only the remote setup is interacted with. Then, the local setup (motor gear) is interacted with as well to increase the current that is drawn by the local motor to replicate the force and position of the task environment. In this (and the following) measurements, the LMS filter is always enabled since no force is applied at the handle. Hence, the measurements are only performed with the conditional LMS filter.

The performance for "slow" disturbances like gravity is already evaluated in experiment 4.3 and can be derived from the first part of this experiment's first measurement as well. However, the performance for short and intense disturbances is yet to be evaluated by once again moving/tapping the setup's base orthogonal and parallel to the force measurement's axis.

Finally, the attenuation of the environmental noise present in the robotics lab is evaluated.

Table 5.1: Overview of the experiments conducted to evaluate the performance of the final implementations of the conditional LMS filter and position-measured force architecture.

Experiment	Objective	Tested configurations	
5.1	Verifying the conditional LMS filter on Arduino	Interacting with remote setup and local setup to test if LMS filter is disabled/enabled correctly	
		Paddle disconnected from the motor to evaluate the attenuation of electromagnetic interference during (position-computed force) teleoperation	
		Touching/moving the setup's base to evaluate the attenuation of short and intense disturbances	
		Evaluating the attenuation of the noise present in the robotics lab	
5.2	Comparing the performance of the position-measured force architecture with the performance of the (provided) position-computed force architecture	Force and position tracking when interacting with a fixed object through the teleoperation system	
		Force and position tracking when interacting with springs with different spring constants:	50 [N/m]
			100 [N/m]
			200 [N/m]
5.3	Comparing the performance of $K_{f,d} = 0.0714$ [N] and $K_{f,d} = 0.1$ [N] when implemented in the position-measured force architecture	Force and position tracking when interacting with a fixed object through the teleoperation system	
		Force and position tracking when interacting with springs with different spring constants:	50 [N/m]
			100 [N/m]
			200 [N/m]
5.4	Verifying the functionality of the conditional LMS filter in the position-measured force architecture	Moving the paddle back and forth at different velocities (without touching the handle) to evaluate the attenuation of slow disturbances such as gravity	
		Interacting with remote setup and local setup to test if LMS filter is disables correctly	
5.5	Evaluating the effects of modifying the load-cell amplifiers to 80 SPS on the performance of the position-measured force architecture with $K_{f,d} = 0.1$ N and the conditional LMS filter	Force and position tracking when interacting with a fixed object through the teleoperation system	
		Force and position tracking when interacting with springs with different spring constants:	50 [N/m]
			100 [N/m]
			200 [N/m]
		Moving the paddle back and forth at different velocities (without touching the handle) to evaluate the attenuation of slow disturbances such as gravity	
		Interacting with remote setup and local setup to test if LMS filter is disables correctly	

Results: As seen in the load-cells' measurements during teleoperation (Figure 5.1), the conditional LMS filter does not distort the force estimate when the local setup is interacted with, while the permanent LMS filter does by drastically amplifying the noise measured by the dummy load-cell to create its noise estimate. When only the remote setup is interacted with, their performance is similar and satisfactory; the force estimate is almost completely zero.

The (conditional, but in this case always enabled) LMS filter greatly reduces the electromagnetic interference during teleoperation, as seen in the force estimate of Figure 5.2, albeit with some remaining slight oscillation. Notably, the interference measured by the dummy load-cell is amplified slightly to completely cancel the interference measured by the active load-cell.

As seen in Figure 5.3, the LMS filter can not cancel the first short and intense disturbances: Figure 5.3. However, when these disturbances are recurring, the filter is increasingly able to replicate them with the noise estimate and thereby cancel them in the force estimate. Nonetheless, they are never completely removed from the force estimate.

The (conditional) LMS filter is not able to attenuate the noise; as seen in Figure 5.4, the noise estimate is zero and the force estimate is equal to the noise measured by the active load-cell.

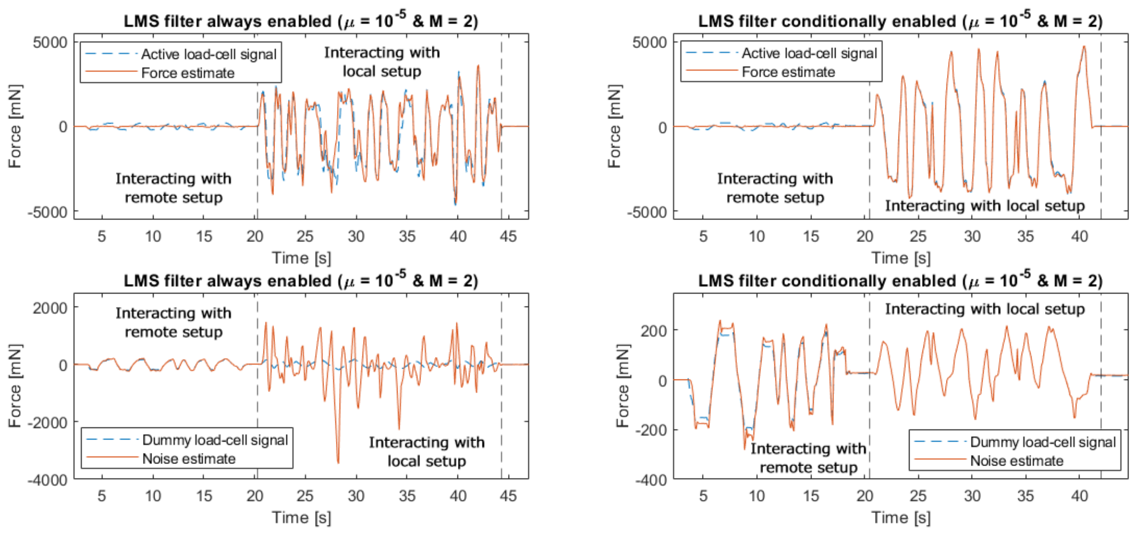


Figure 5.1: Comparing the permanent and conditional LMS filters (on the "local" setup) with a threshold of 430 mN and timer of 0.1 s during position-computed force teleoperation, conducted in the robotics lab. Force is applied at the handle between 20/21 [s] and 44/42 [s] (left/right plot respectively).

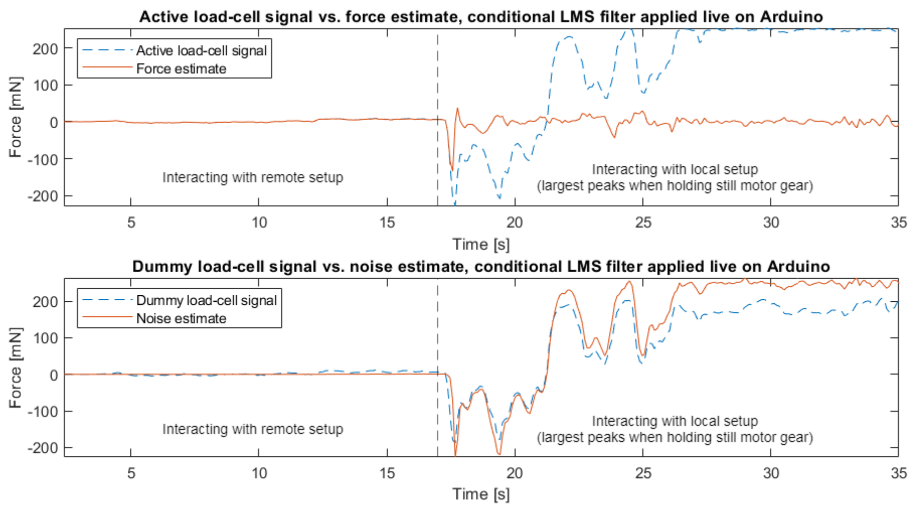


Figure 5.2: Using the LMS adaptive filter (on the "local" setup) conditionally with $\mu = 1 \cdot 10^{-5}$ and $M = 2$ to filter out the electromagnetic interference caused by the motor in the active teleoperation system, conducted in the robotics lab. The paddle containing the load-cells was disconnected to only measure the interference (and noise).

Discussion: The conditional LMS filter performs much better when a force is applied at the handle since it simply subtracts the dummy load-cell signal from the active-load cell signal. In case of no force applied at the handle, it performs equally well as the permanent filter since the exact same LMS filter is enabled and applied to the measurements.

For the electromagnetic interference measurements, the load-cells were placed closer to each other and closer to the motor when compared to Figure 4.7, which explains the difference in amplitudes. Also, the dummy load-cell was placed slightly further away from the motor compared to the active load-cell (the paddle was rotated to the left), resulting in its generally slightly lower measured interference amplitude.

Figure 5.3 indicates that the LMS filter can not update its coefficients fast enough to cancel (the first) short and intense disturbances. After 15 s, the remaining force estimate is caused by

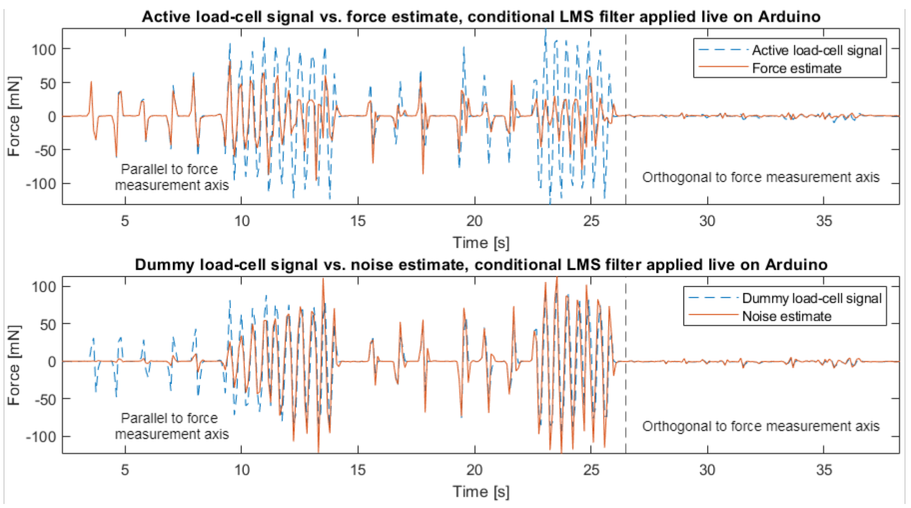


Figure 5.3: Using the LMS adaptive filter conditionally with $\mu = 1 \cdot 10^{-5}$ and $M = 2$ to filter out short and intense disturbances, conducted in the robotics lab.

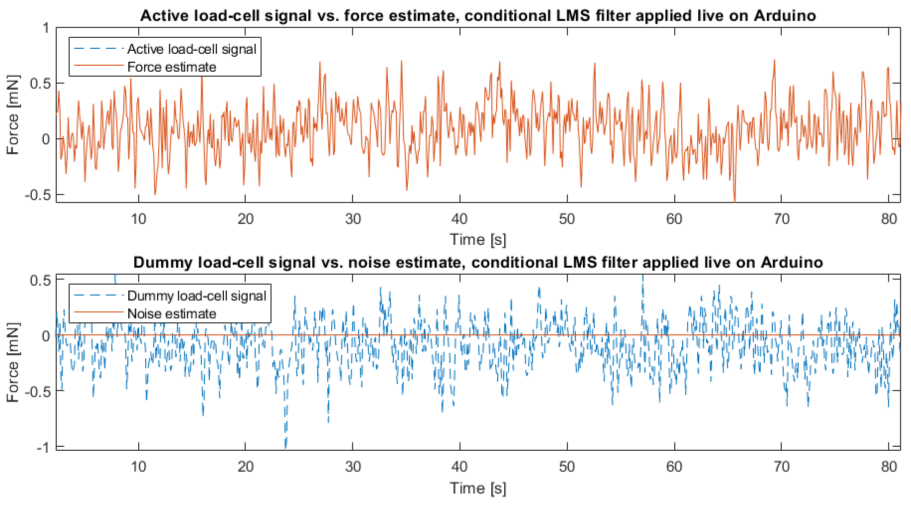


Figure 5.4: Using the LMS adaptive filter conditionally with $\mu = 1 \cdot 10^{-5}$ and $M = 2$ to filter out environmental noise, conducted in the robotics lab.

the aforementioned delay between the load-cells' measurements, as the noise estimate almost perfectly replicates the disturbances measured by the dummy load-cell.

Since the (environmental) noise does not present this recurring behaviour, the filter coefficients are never updated fast enough to replicate it with the noise estimate. Even though this may marginally increase the energy generation in the system (if the noise is not biased around zero), the force due to this noise will never be larger than the friction in the system, and therefore not felt by the operator. Simply subtracting the dummy load-cell's noise from the active load-cell's noise without the LMS filter is not recommended, as this would increase the total noise in the force estimate when one load-cell measures a slightly negative noise value and the other a slightly positive noise value.

Nonetheless, the subpar performance of the filter for environmental noise is not helped by the limited amount of (10) samples per second that can be obtained from the load-cell through the amplifier, as will be discussed in the next section.

Conclusion: The conditional LMS filter with a step size of 10^{-5} and order of 1 provides adequate results when implemented in real-time on the Arduino Uno. While noise in the robotics lab and sudden intense disturbances are not sufficiently attenuated, the most frequent sources of noise in the force estimate are: gravity and electromagnetic interference. Most importantly, disabling the filter when a force is applied on the handle eliminates the distortion of the force estimate.

Experiment 5.2: Comparing the performance of the position-measured force architecture with the performance of the (provided) position-computed force architecture

Method: To compare the performance (i.e. position and force transparency/tracking), two experiments are conducted for both architectures. These experiments should represent most task environments (impedances) of this DIY teleoperation system.

First, the setup on the task side is placed next to a fixed (steel) object, while both setups are still clamped to the table: Figure 5.5. The paddle's angle on the human side is offset to the left, such that any position difference between the motors (due to sub-optimal position control) can be measured without this paddle rotating off the motor gear when the handle on the task side is placed against the object.

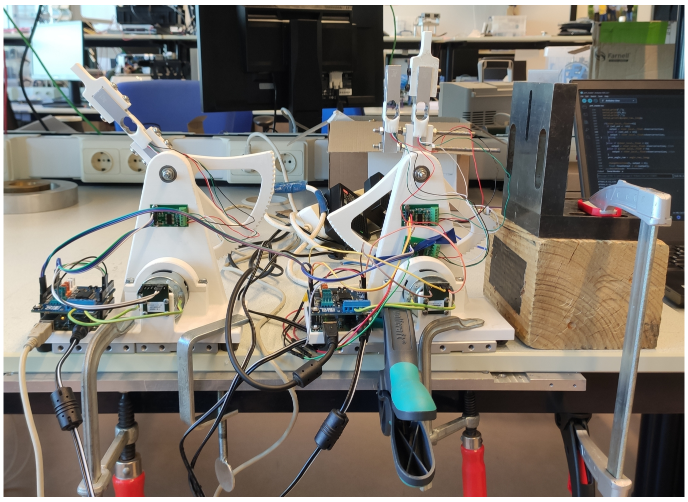


Figure 5.5: Measurement setup used in experiments 5.2, 5.3 and 5.5 to measure the transparency in terms of position and force tracking when interacting with a fixed object.

During this experiment, the handle on the human side is rotated to the right multiple times, such that the handle on the task side hits the fixed object. Also, when this handle is in contact with the object, different levels of force will be applied to the handle on the human side to simultaneously evaluate the force tracking performance (both active load-cells measure the same force) and position tracking performance (the motor angles should be equal and constant).

Secondly, the handle on the task side is connected to the fixed object through different mechanical force meters (the same 5 N, 10 N and 20 N ones as used in experiment 3.3 in Section 3.2.5): Figure 5.7. With this experiment, the measured force tracking performance can be validated by comparing it to the force calculated from the measured motor angle with the force-position relation of a spring (Hooke's law): $F = -k \cdot x$, where k is the spring constant. The spring constants for the mechanical force meters have been determined based on the measured length to which their spring extends for their maximum force: Table 5.2.

To calculate the force in newtons that is required to extend the spring per motor angle (φ_m) in radians, the information in Figure 5.6 is to be used. Due to the transmission, a change in motor angle results 13.6 times smaller change in paddle angle. Using this transmission ratio and the cosine, the horizontal position on the light-blue dotted line can be calculated with the motor angle: $x = 0.095 \cdot \cos\left(\frac{\text{motor angle}}{13.6}\right)$. However, the angle of the paddle started at roughly 45 degrees ($\frac{\pi}{4}$ radians, starting from the right end of the semicircle in Figure 5.6), where the extension of

Table 5.2: Determined spring constants for the mechanical force meters.

Measurement range [N]	Length of measurement range [m]	Spring constant K [N/m]
0 - 5	0.10	50
0 - 10	0.10	100
0 - 20	0.10	200

the spring (x in Hooke's law) was zero, as seen in Figure 5.7. Hence, the extension of the spring should be calculated as Equation 5.1, leading to Equation 5.2 to calculate the theoretical spring force with the spring constant K and motor angle φ_m.

$$x = 0.095 \cdot \cos\left(\frac{\varphi_m}{13.6} + \frac{\pi}{4}\right) - 0.095 \cdot \cos\left(\frac{\pi/4}{13.6} + \frac{\pi}{4}\right) \approx 0.095 \cdot \cos\left(\frac{\varphi_m}{13.6} + \frac{\pi}{4}\right) - 0.0632 \tag{5.1}$$

$$F_{\text{spring}} = -K \cdot x \approx -K \cdot \left(0.095 \cdot \cos\left(\frac{\varphi_m}{13.6} + \frac{\pi}{4}\right) - 0.0632\right) \tag{5.2}$$

This calculation (Equation 5.2) should be seen as an approximation as it assumes that the spring remains perfectly horizontal and applies force perfectly orthogonal to the load-cell throughout the measurement.

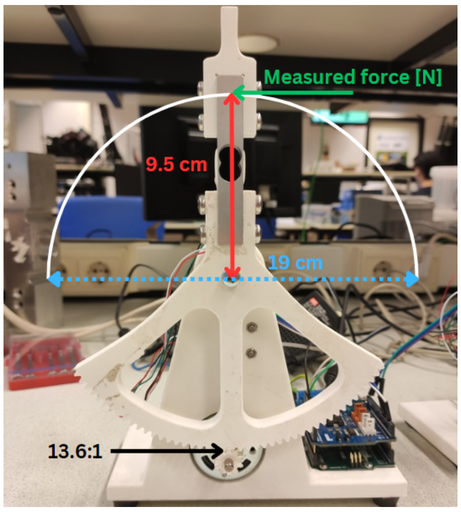


Figure 5.6: Characteristics of the DIY setup required to convert the motor angle into a horizontal displacement of the handle.

To measure and record this data, the setups' encoders and load-cells are used. This means that each setup has to send the force (estimate), motor angle and time, so the loop frequency has to be reduced to 300 Hz for this experiment as well.

The performance is quantified in terms of the average force and position tracking error percentages, calculated by taking the average absolute error over time and dividing it by the absolute force or position on the task side. This division by the absolute force/position on the task side ensures that the error percentages can be compared between measurements, as the measurements are not performed in exactly the same way (with the same position/force amplitudes) for each architecture.

The optimal proportional gains for the architectures are used, as found in the previous experiment: $K_p = 8$ for the position-computed force architecture and $K_p = 20$ for the position-measured force architecture. Furthermore, for the position-measured force architecture, the constant $K_{f,d}$ that was determined in experiment 3.2 (Section 3.1.6) is used: 0.0714 N.

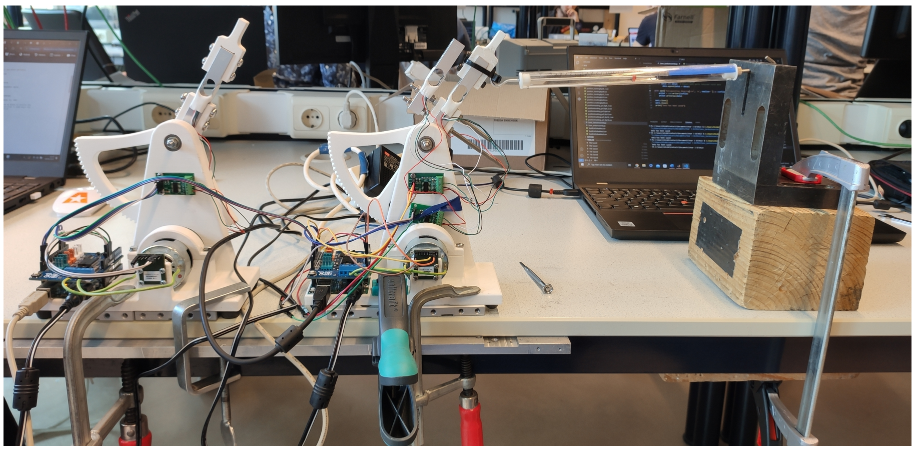


Figure 5.7: Measurement setup used in experiments 5.2, 5.3 and 5.5 to measure the transparency in terms of position and force tracking when interacting with (one end of) a spring.

Finally, the "steercorrection" value in the provided position-computed force code is increased to the value used in the position-measured force architecture (125, from Figure 3.13) to ensure a fair comparison.

Results: The results from interacting with a fixed object through the teleoperation system can be seen in Figure 5.8. For both measurements, the motor angle on the task side is restricted around -8 radians, indicating that the handle is touching the object at that angle. While the motor angle on the human side does not go far beyond that limit of -8 radians with the position-measured force architecture, it does with the position-computed force architecture, sometimes doubling the motor angle on the task side. This is reflected in the average position error percentages displayed in the bottom plots of Figure 5.8.

However, the average force error percentages do not differ as much between the architectures; the average force error with the position-measured force architecture is only slightly lower than with the position-computed force architecture. This higher error percentage for the position-computed force architecture seems to be (at least partially) caused by the significant delay between the force on the human side and the force on the task side, which is not nearly as noticeable for the position-measured force architecture.

The results from interacting with the mechanical force meters (springs) through the teleoperation system can be seen in Figures 5.9, 5.10 and 5.11 for the 5 N, 10 N and 20 N force meters respectively. In the force tracking plots, the "Theoretical" force represents the spring force calculated with the motor angle using Equation 5.2 and Table 5.2. The clear lack of "smoothness" of these force tracking plots is a consequence of reusing the measured force value for multiple (50) loops.

Most observations made in the measurements resulting from interacting with a fixed object reappear in these measurements; the position tracking error percentages and force tracking delay with the position-measured force architecture are much lower than with the position-computed force architecture. However, in these measurements, the difference between the force tracking error percentages is greater; the position-measured force architecture significantly outperforms the position-computed force architecture in terms of force tracking for all mechanical force meters. Also, the average force error percentages decrease for an increasing spring constant, while the average position error percentages increase.

Notably, the theoretical spring force for the 5 N mechanical force meter deviates quite a bit from the force measured by the load-cell on the task side. However, for the 10 N and 20 N mechanical force meters, this deviation is much smaller.

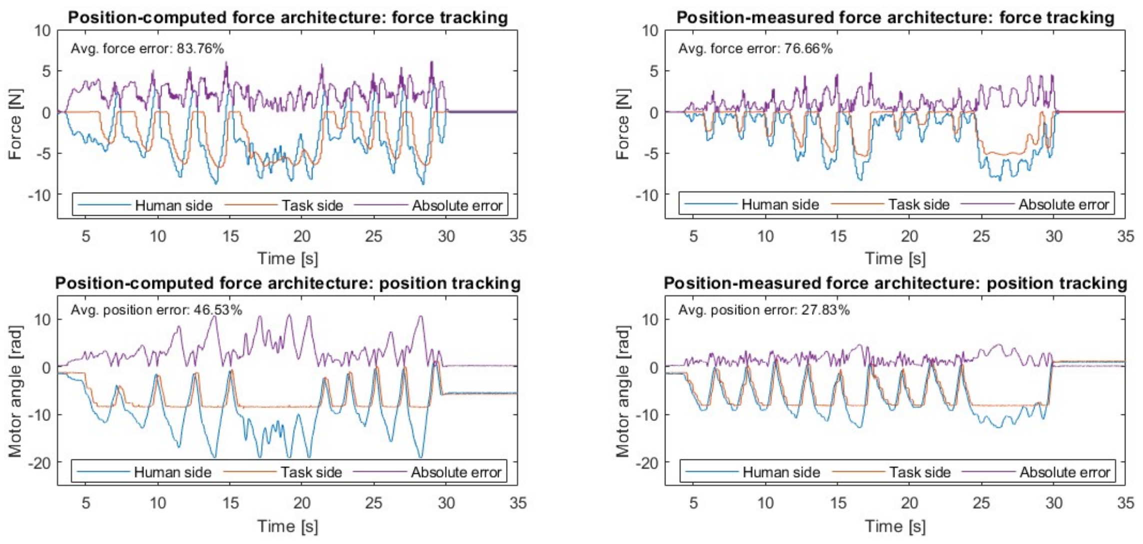


Figure 5.8: Comparing the transparency of the position-computed force ($K_p = 8$) and the position-measured force ($K_p = 20$ and $K_{f,d} = 0.0714$ N) architectures in terms of force and position tracking when interacting with a fixed object.

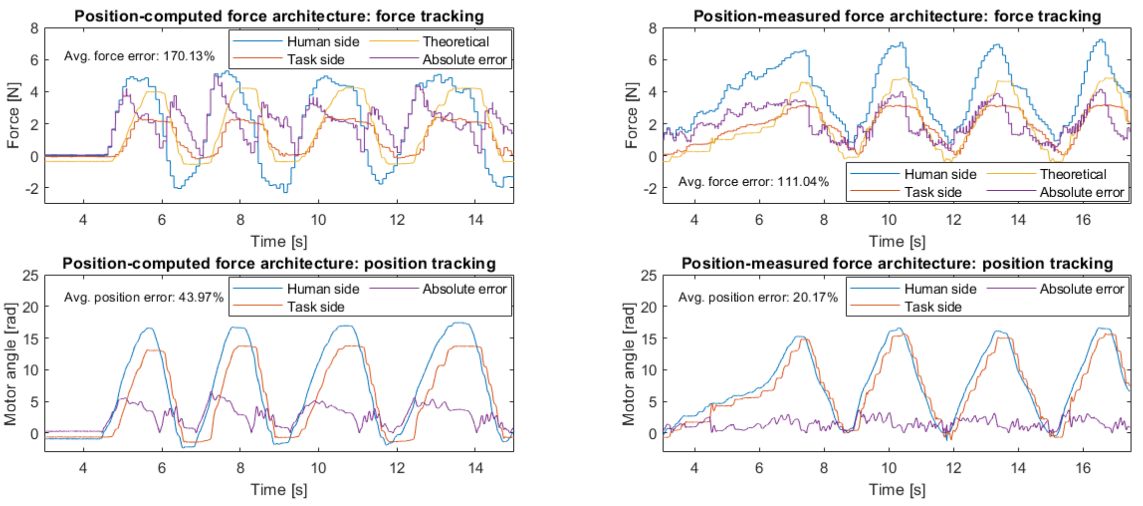


Figure 5.9: Comparing the transparency of the position-computed force ($K_p = 8$) and the position-measured force ($K_p = 20$ and $K_{f,d} = 0.0714$ N) architectures in terms of force and position tracking when interacting with a 5 N mechanical force meter (50 N/m).

Furthermore, it should be noted that more resistance or friction was felt when the system was using the position-computed force architecture, even though the "steercorrection" value that compensates for the Coulomb friction (among other things) was the same as for the position-measured force architecture. Also, when the both setups' handles were interacted with by a single operator, the setups started to become difficult to manage due to increasing (synchronised) oscillation. When the operator tried to correct for this by holding still both handles, the system occasionally became unstable.

Across all measurements in this experiment, the position-measured force architecture reduces the average force and position tracking error percentages of the position-computed force architecture by 26.2 % and 39.4 % respectively.

Discussion: From these results, it is clear that the position-measured force architecture provides a higher overall transparency in all measurements. However, the force transparency is not as much higher as expected; since the force on the task side is measured directly, the

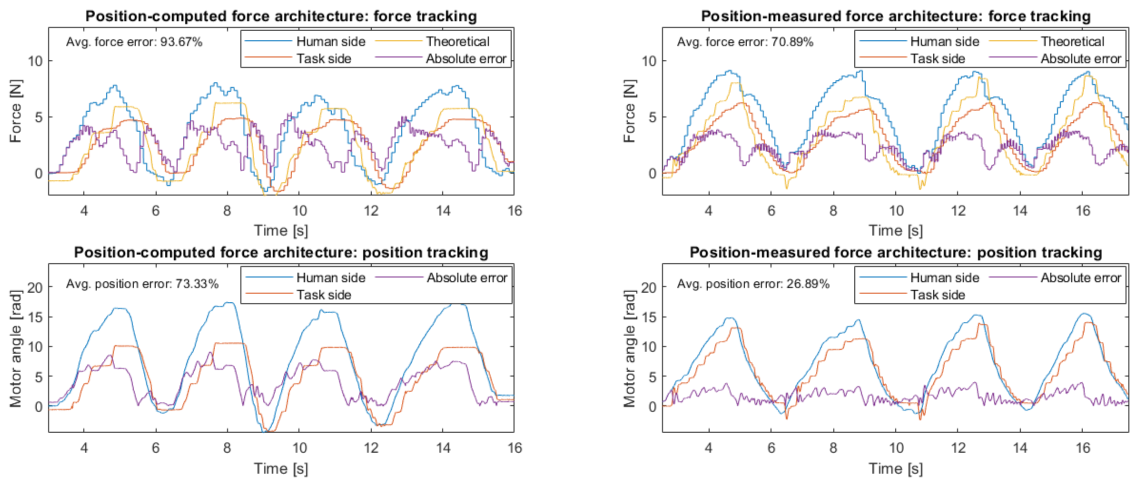


Figure 5.10: Comparing the transparency of the position-computed force ($K_p = 8$) and the position-measured force ($K_p = 20$ and $K_{f,d} = 0.0714$ N) architectures in terms of force and position tracking when interacting with a 10 N mechanical force meter (100 N/m).

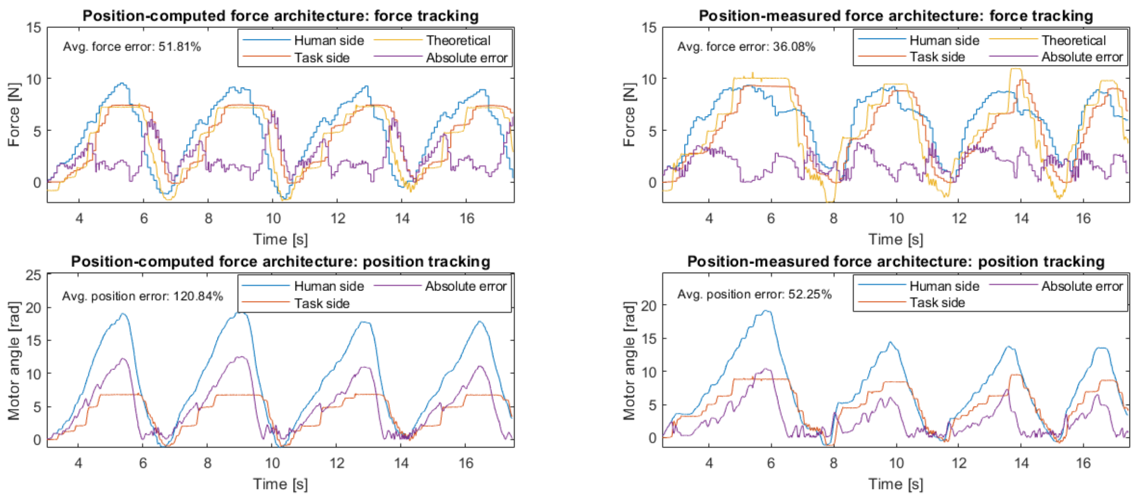


Figure 5.11: Comparing the transparency of the position-computed force ($K_p = 8$) and the position-measured force ($K_p = 20$ and $K_{f,d} = 0.0714$ N) architectures in terms of force and position tracking when interacting with a 20 N mechanical force meter (200 N/m).

dynamics of the setup on the task side are masked: as explained in Section 2.1.3, the force felt by the operator is equal to $F_{h,i} = F_{r,i} + F_{h,d}$, while the force felt by the operator in the position-computed force architecture is equal to $F_{h,i} = F_{r,i} + F_{h,d} + F_{r,d}$, where $F_{h,d}$ and $F_{r,d}$ are the dynamics of the setups on the human side and task side respectively. So, the force transparency should be able to be significantly higher when the force is measured instead of computed based on the position error between the setups.

In the position-measured force architecture, the constant $K_{f,d}$ is the crucial factor for force transparency as this constant is used for translating the measured force on the task side into a PWM duty cycle for the setup on the human side. Hence, the constant $K_{f,d}$ (0.0714 N) seems to be slightly too low, translating the measured force into a PWM duty cycle that results in a slightly higher force. This slightly higher force does contribute to the system being difficult to manage when both setups are interacted with by the operator, but the primary cause of this issue is the load-cell amplifiers' very low sampling rate compared to the loop frequency. Because of this low sampling rate, a force that is measured briefly on the task side is sustained on the human side for an extended duration.

Since the force on the human side was not filtered with the conditional LMS adaptive filter, while the force on the task side was, a slight deviation between these measurements should be expected due to gravity on the human side's setup; when rotating the paddle all the way from the left to the right, this deviation can be up to 0.4 N (Figure 4.6). However, this applies to the measurements for both architectures, so the transparency comparison between the architectures is not influenced by this deviation.

The proportional controller gain K_p is the crucial factor for position transparency in the position-measured force architecture. Since the setup's dynamics on the task side are masked by measuring the force directly, the proportional gain can be significantly increased from the position computed-force's gain ($K_p = 8$) before the system exhibits excessive oscillation or instability.

The observed deviation of the measured forces from the theoretical spring forces is a consequence of the aforementioned assumptions in the spring force calculation, which do not hold in case of a large spring extensions (x). When the handle on the task side rotates to the left in Figure 5.7, the spring does not stay completely horizontal; it moves up and down slightly. More importantly, the spring force is only applied orthogonal to the load-cell when the handle is pointed straight up. At all other angles, the spring force is applied at an angle and therefore not measured accurately by the load-cell. Since the 10 N and especially 20 N mechanical force meters do not extend as much as the 5 N force meter due to their higher spring constants (Table 5.2), the assumptions made in the calculation are more accurate and the theoretical and measured spring forces align better.

This explains the decreasing force error for an increasing spring constant, but not the increasing position error. This position error is a consequence of the limited torque that the motors can provide; this maximum torque depends greatly on the setup's temperature, but the force at the handle can never reach 20 N (Figure 3.12). Therefore, positions can not be kept equal in situations where the torque on the task side has to exceed the motor's limit to do so, such as applying a large force to a fixed object or pulling on spring with a large spring constant. Nevertheless, these forces are way beyond the force for which the setup starts to move or tilt (4 N), so this is not a realistic scenario for this DIY kit.

Finally, the relatively high resistance or friction that was felt with the position-computed force architecture can be observed in Figure 5.8; to rotate the paddle back (away from the object), a larger force had to be used than for the position-measured force architecture, as can be concluded from the significant positive peaks in the force measured on the human side with the position-computed force architecture. This resistance may be attributed to the noticeable delay in the force tracking plots, as the (Coulomb) friction compensation was equal for both architectures. Also, as mentioned before, the operator is expected to feel more of the system's dynamics, as the dynamics of the setup on the task side are not masked in the position-computed force architecture.

Conclusion: When compared to the position-computed force architecture, the position measured force architecture provides an overall better transparency in terms of force tracking an position tracking in the most probable teleoperation scenarios. However, with the current $K_{f,d}$ of 0.0714 N, the force reflected to the operator is slightly amplified and system can be difficult to manage if both setups are controlled by one operator.

Experiment 5.3: Verifying the revised (motor) constant for the position-measured force architecture

Method: To verify that the empirically determined $K_{f,d}$ of 0.1 outperforms the $K_{f,d}$ that is based on measurements (for this specific DIY kit), the exact same measurements as in experiment 5.2 are performed for the position-measured force architecture with this revised $K_{f,d}$. However, in this experiment these results are compared with the same architecture using the $K_{f,d}$ of 0.0714 N, instead of the position-computed force architecture.

Results: The results from interacting with a fixed object through the teleoperation system with the measured and empirically determined $K_{f,d}$ can be seen in Figure 5.12. With the revised $K_{f,d}$ of 0.1 N, the average force tracking error percentage is drastically reduced to less than half of the error that occurs with a $K_{f,d}$ of 0.0714 N. However, the average position error percentage is noticeably larger with the revised $K_{f,d}$.

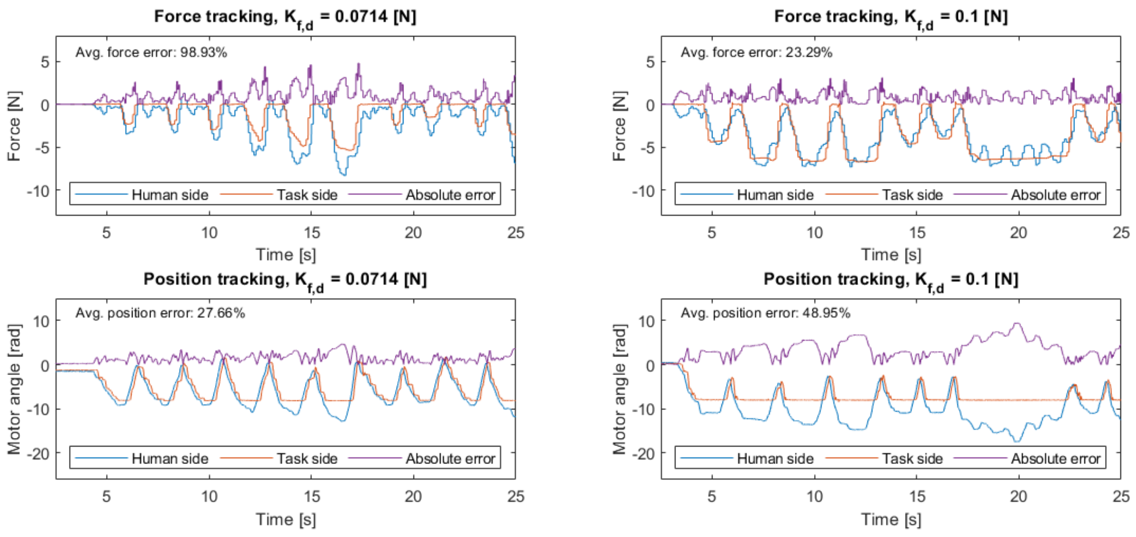


Figure 5.12: Comparing the transparency of the position-measured force architecture with $K_{f,d}$ = 0.0714 [N] and $K_{f,d}$ = 0.1 [N] in terms of force and position tracking when interacting with a fixed object.

For the measurements with the mechanical force meters (Figures 5.13, 5.14 and 5.15), this trend of a drastically lower average force error percentage for the revised $K_{f,d}$ continues. Notably, the difference in average position error percentages is no longer significant in these measurements.

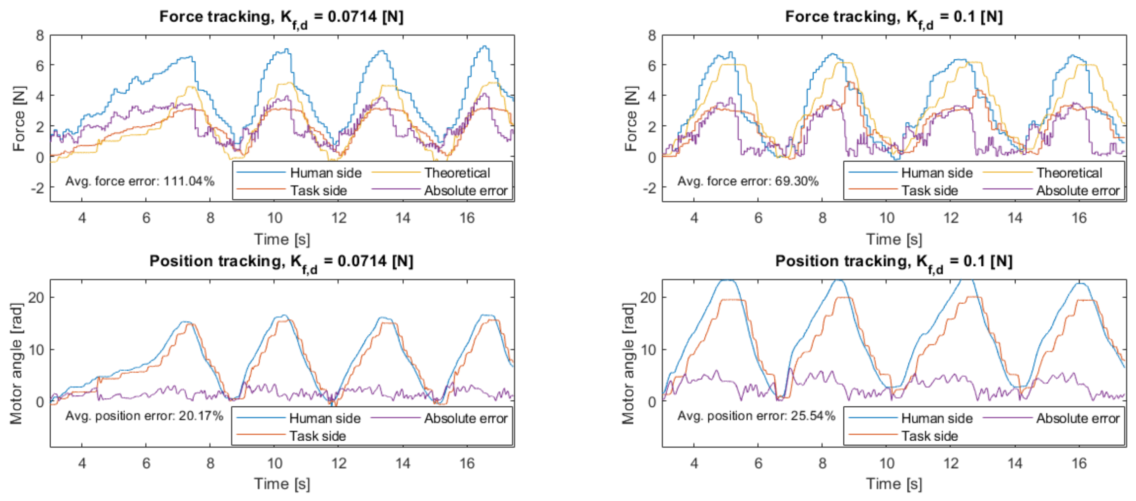


Figure 5.13: Comparing the transparency of the position-measured force architecture with $K_{f,d}$ = 0.0714 [N] and $K_{f,d}$ = 0.1 [N] in terms of force and position tracking when interacting with a 5 N mechanical force meter (50 N/m).

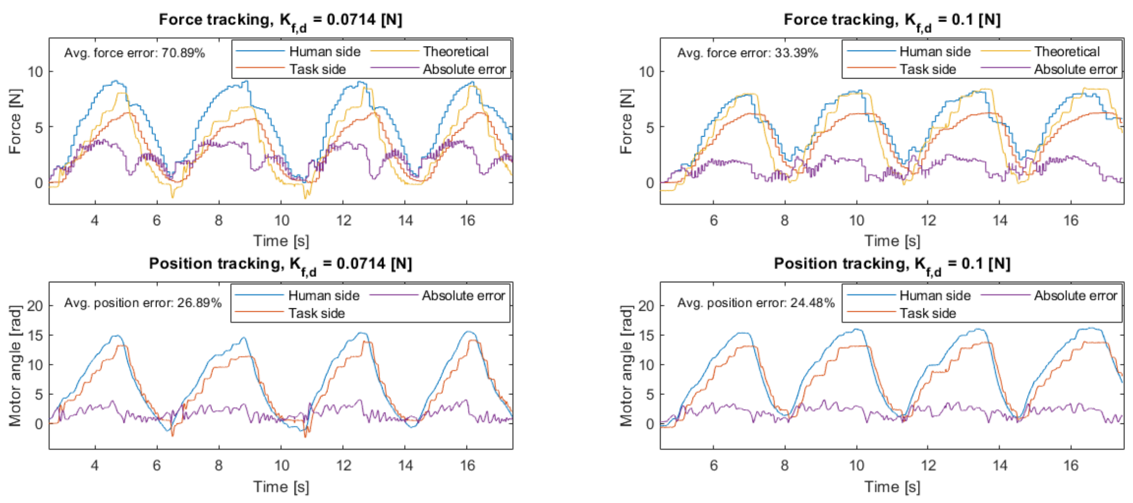


Figure 5.14: Comparing the transparency of the position-measured force architecture with $K_{f,d}$ = 0.0714 [N] and $K_{f,d}$ = 0.1 [N] in terms of force and position tracking when interacting with a 10 N mechanical force meter (100 N/m).

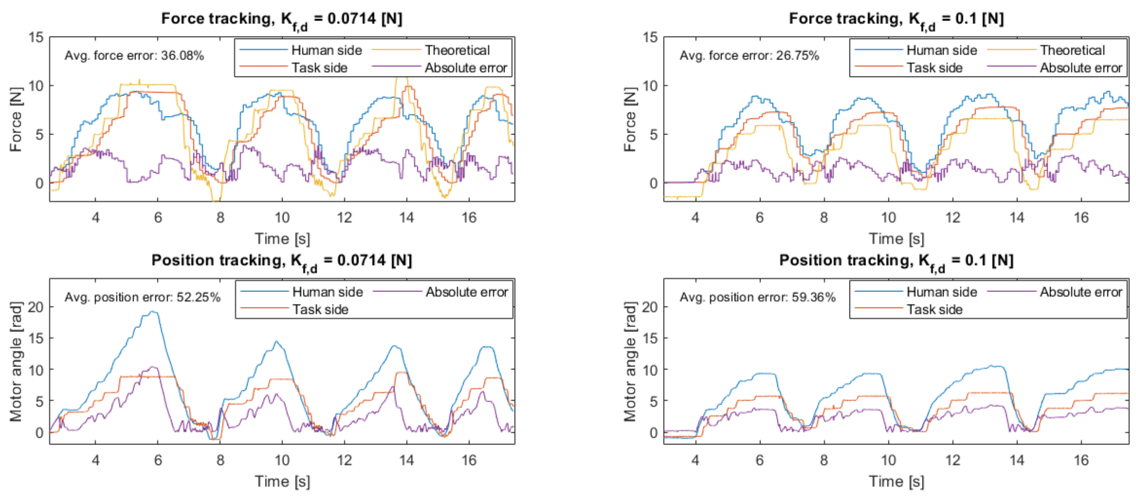


Figure 5.15: Comparing the transparency of the position-measured force architecture with $K_{f,d}$ = 0.0714 [N] and $K_{f,d}$ = 0.1 [N] in terms of force and position tracking when interacting with a 20 N mechanical force meter (200 N/m).

Across all measurements in this experiment, using the revised $K_{f,d}$ in the position-measured force architecture reduces the average force tracking error percentage of using the measured $K_{f,d}$ by another 35.5 %, but increases the average position tracking error percentage by 7.8 %.

Discussion: Almost all measurements support the claim that the revised $K_{f,d}$ of 0.1 N provides a higher transparency with the position-measured force architecture than the $K_{f,d}$ of 0.0714 N. The one measurement that opposes this claim is the position tracking performance for interacting with a fixed object, where the average position error percentage increases when using the revised $K_{f,d}$.

Since the average position error percentages in the other measurements (with the mechanical force meters) do not differ notably between per $K_{f,d}$, this difference seems to be specific to interacting with a fixed object. As discussed previously, the position error in this measurement is mainly caused by the limited amount of torque that the motors can provide. As the force reflected to the operator is less, but more accurate, when using the revised $K_{f,d}$, the paddle on the human side offers less resistance/force to the operator's hand and can be rotated a bit further beyond the angle at which the paddle on the task side was fixed, which explains the higher average position error percentage.

While this revised $K_{f,d}$ provides a much improved overall transparency, it does deviate considerably from the measured $K_{f,d}$. So, before implementing this $K_{f,d}$ for all DIY setups, it should at least be verified that the transparency remains similar when the setup is used for an extended period of time (increasing temperature). Ideally, this should be verified for multiple different pairs of DIY setups.

Conclusion: When used in the position-measured force architecture and this specific DIY kit, the revised $K_{f,d}$ of 0.1 N results in a significantly improved transparency in terms of position tracking and (especially) force tracking over the measured $K_{f,d}$ of 0.0714 N.

Experiment 5.4: Verifying the functionality of the conditional LMS adaptive filter in the position-measured force architecture

Method: To ensure that the conditional LMS adaptive filter with the optimal values ($\mu = 10^{-5}$ and $M = 2$, determined in experiment 4.3) still works when implemented in this position-measured force architecture, the paddle is moved back and forth at different speeds with and without applying force at the handle. As discussed in experiment 4.2b (Section 4.1.3), without force applied at the handle the filter should always be enabled and cancel every disturbance in the active load-cell signal. When a force is applied at the handle, this filter should be disabled and the dummy load-cell signal should simply be subtracted from the active load-cell signal.

Results: As seen in Figure 5.16, the LMS filter is indeed disabled only when the handle is interacted with; only in this case, the noise estimate perfectly aligns with the dummy load-cell signal and the force estimate is not distorted.

However, to be able to evaluate the filter's performance when it is enabled, the different sections of Figure 5.16 are isolated and enlarged: Figures 5.17 and 5.18.

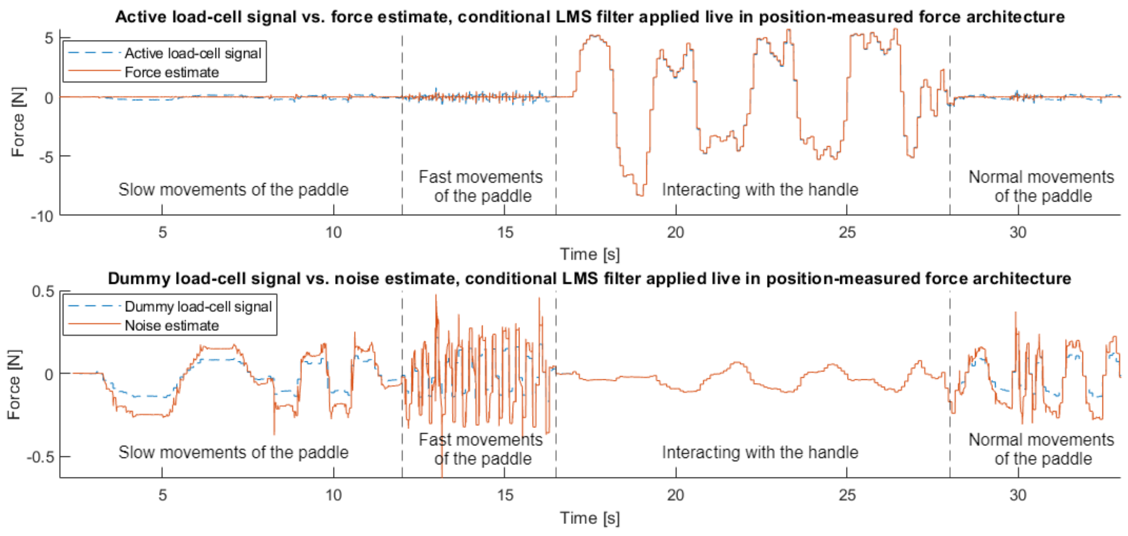


Figure 5.16: Using the LMS adaptive filter conditionally with $\mu = 1 \cdot 10^{-5}$ and $M = 2$ in the position-measured force architecture implemented live on the Arduino Uno, conducted in the robotics lab. During the slow, normal and fast movements of the paddle, the handle was not interacted with.

Aside from some slight oscillation and very narrow peaks in the force estimate in case of fast(er) movements, the filter almost completely filters out the forces caused by the movements (inertia) of the paddle and the effect of gravity.

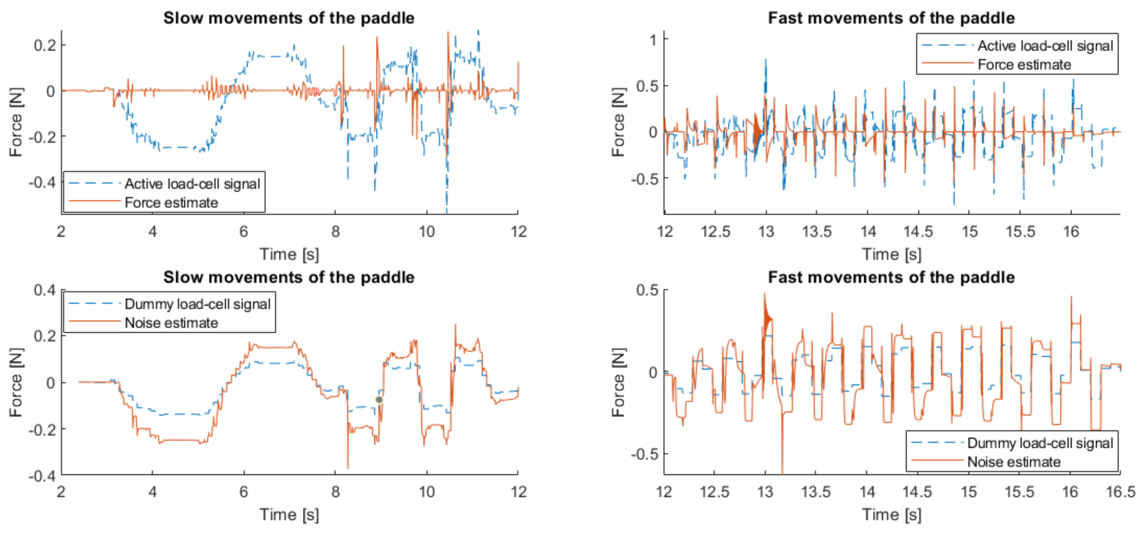


Figure 5.17: Using the LMS adaptive filter conditionally with $\mu = 1 \cdot 10^{-5}$ and $M = 2$ in the position-measured force architecture implemented live on the Arduino Uno, conducted in the robotics lab. During these measurements, no force was applied at the handle (active load-cell).

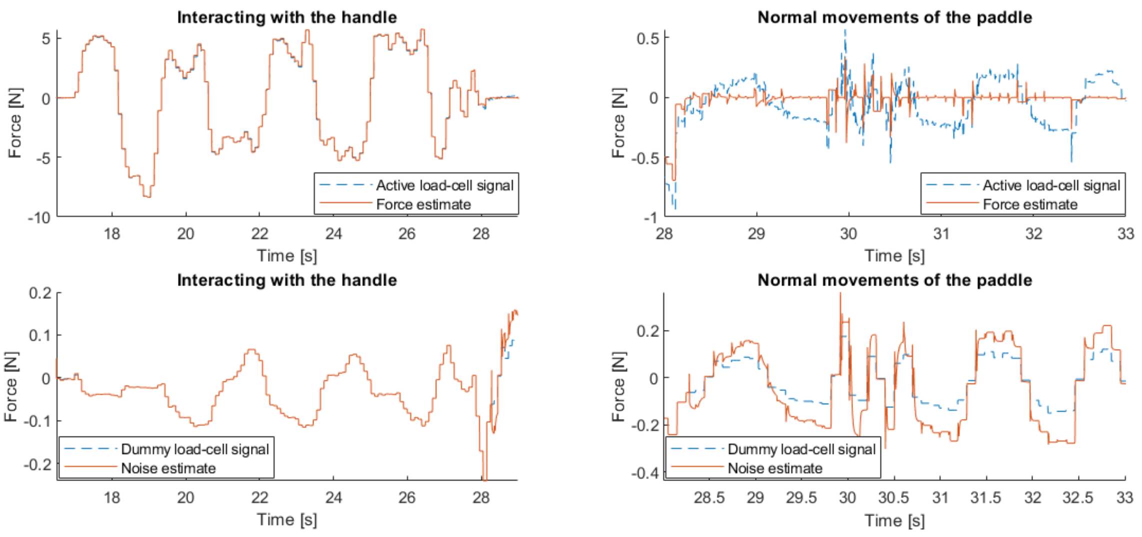


Figure 5.18: Using the LMS adaptive filter conditionally with $\mu = 1 \cdot 10^{-5}$ and $M = 2$ in the position-measured force architecture implemented live on the Arduino Uno, conducted in the robotics lab. During the normal movements of the paddle, no force was applied at the handle (active load-cell).

Discussion: When implemented in the position-measured force architecture, the determined threshold of 430 mN and timer of 0.1 s to disable the LMS filter still ensure that the filter is disabled when a force is applied at the handle (active load-cell). Furthermore, the filter (with $\mu = 10^{-5}$ and $M = 2$) still sufficiently attenuates the disturbances that are most commonly acting on the paddle, showing similar performance to experiments 4.2 and 4.3.

As discussed previously, the minor oscillation and narrow peaks in the force estimate are caused by the relatively slow LMS algorithm and the remaining slight delay between the dummy and active load-cells' measurements.

Conclusion: The functionality and performance of the conditional LMS adaptive filter is not significantly affected by its implementation in the position-measured force architecture; it continues to suffice for filtering out the most common disturbances during teleoperation.

Experiment 5.5: Evaluating the performance of the position-computed force architecture with the modified load-cell amplifiers

Method: To evaluate the effects of modifying the load-cell amplifiers (to provide 80 SPS) on the performance of the position-measured force architecture with a $K_{f,d}$ of 0.1 N, the exact same measurements as in experiments 5.2 and 5.3 are performed once more and compared to the measurements obtained with the standard load-cell amplifiers.

Results: When interacting with a fixed object through the teleoperation system, the increased amplifier's sampling rates slightly increases the average force tracking error percentage but decreases the average position tracking error percentage: Figure 5.19.

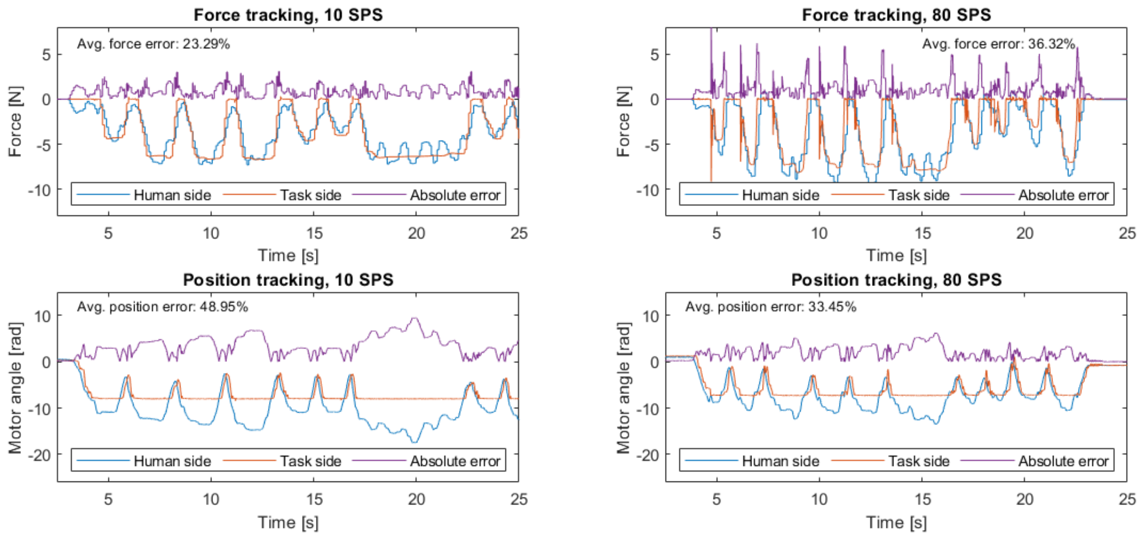


Figure 5.19: Comparing the transparency of the position-measured force architecture with HX711 sampling rates of 10 SPS and 80 SPS and $K_{f,d}$ = 0.1 [N] in terms of force and position tracking when interacting with a fixed object.

When interacting with springs (mechanical force meters) through the teleoperation system, the average force tracking error percentage is reduced slightly across all measurements with the increased amplifier's sampling rates: Figures 5.20, 5.21 and 5.22. The average position tracking tracking error percentage is reduced significantly when interacting with the 5 N and 20 N mechanical force meters, but increased notably when interacting with the 10 N mechanical force meter.

Across all these measurements, using the increased amplifier's sampling rates in the position-measured force architecture increases the average force tracking error percentage of using the standard sampling rates by 0.4 %, but reduces the average position tracking error percentage by 7.1 %.

Furthermore, all measurements in this experiment show a significant increase in noise in the force measurement on the task side, which is also observed in the LMS filter's plots: Figures 5.23, 5.24 and 5.25. While the filter is replaced by the basic noise estimation when handle is interacted with, its performance in terms of attenuation of the paddle movements is drastically reduced compared to the performance with the standard sampling rate (experiment 5.4); it contains a lot more noise in the force estimate.

Discussion: The increase in force tracking error when interacting with a fixed object originates from the large peaks that occur each time the fixed object is hit by the handle (e.g. around 4.5 s and 6 s in Figure 5.19). Since these peaks were also observed when switching the load-cells and amplifiers of the setups (making the setup on the human side the setup on the task side and vice versa), it is very likely to be a consequence of the increased sampling rate; this sampling

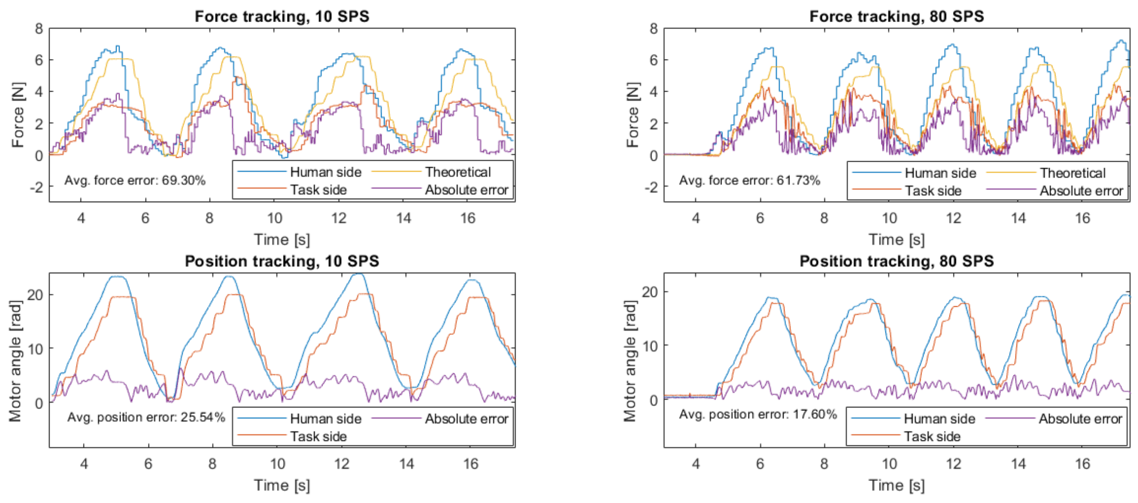


Figure 5.20: Comparing the transparency of the position-measured force architecture with HX711 sampling rates of 10 SPS and 80 SPS and $K_{f,d} = 0.1$ [N] in terms of force and position tracking when interacting with a 5 N mechanical force meter (50 N/m).

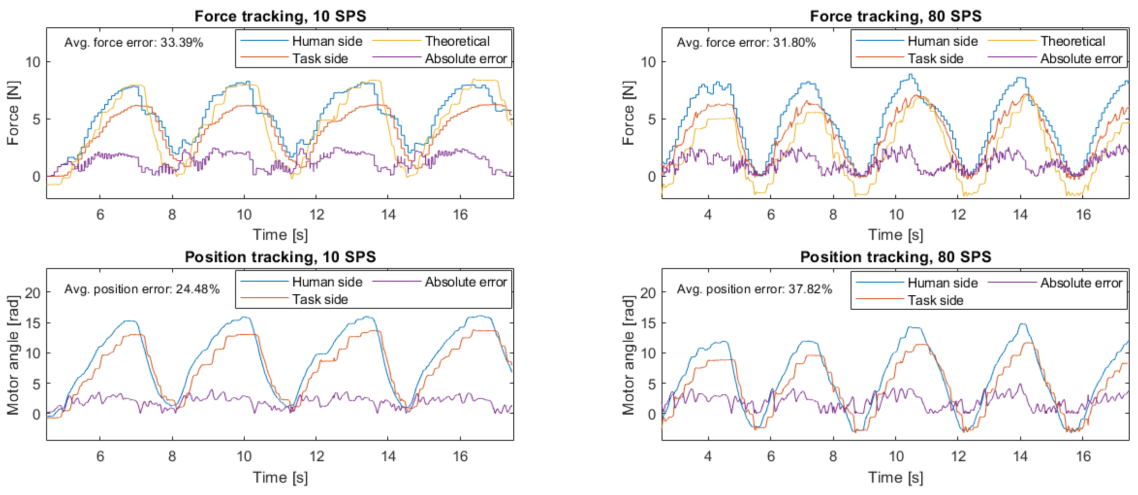


Figure 5.21: Comparing the transparency of the position-measured force architecture with HX711 sampling rates of 10 SPS and 80 SPS and $K_{f,d} = 0.1$ [N] in terms of force and position tracking when interacting with a 10 N mechanical force meter (100 N/m).

rate is 8 times higher than the standard sampling rate, so the standard sampling rate may have been too low to measure the short peaks that occur right after hitting the object.

The position tracking error percentage during interaction with a fixed object is greatly dependent on how much force the operator applies to the handle and how much torque the motors can provide (which is temperature-dependent), and should therefore not be regarded as highly representative of the position tracking performance, as discussed in experiments 5.2 and 5.3.

The increased noise in the force measurements corresponds with the findings of experiment 4.5. This added noise is increased in power due to the amplifiers' modification (experiment 4.5) and not able to be replicated by the LMS adaptive filter in its noise estimate and thereby attenuated in the force estimate, as explained in experiment 5.1.

Finally, it should be noted that the load-cell amplifier on the human side was not modified to provide 80 SPS. This decision was made to avoid the risk of damaging this (last) load-cell amplifier, as one had already been damaged during the modification. Consequently, the measured

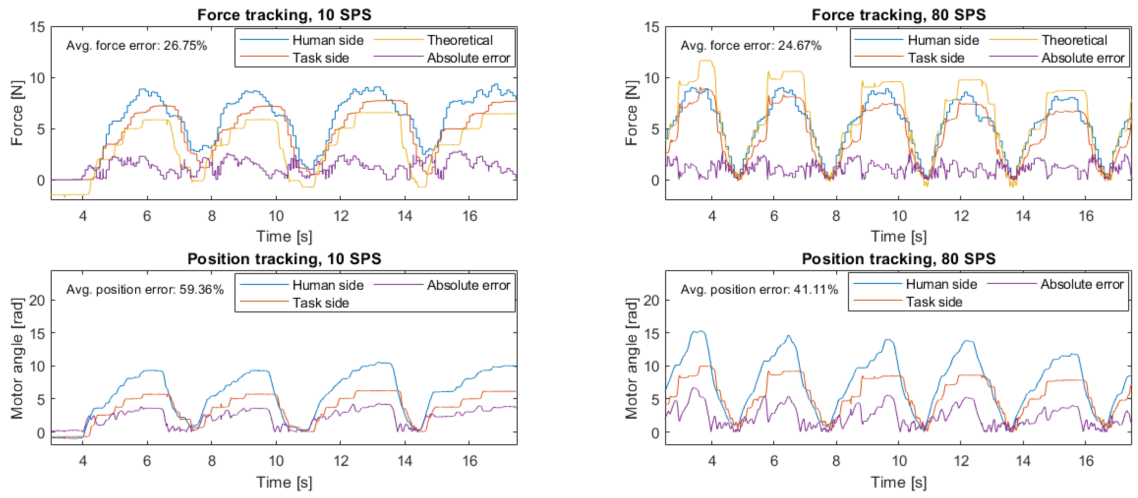


Figure 5.22: Comparing the transparency of the position-measured force architecture with HX711 sampling rates of 10 SPS and 80 SPS and $K_{f,d} = 0.1$ [N] in terms of force and position tracking when interacting with a 20 N mechanical force meter (200 N/m).

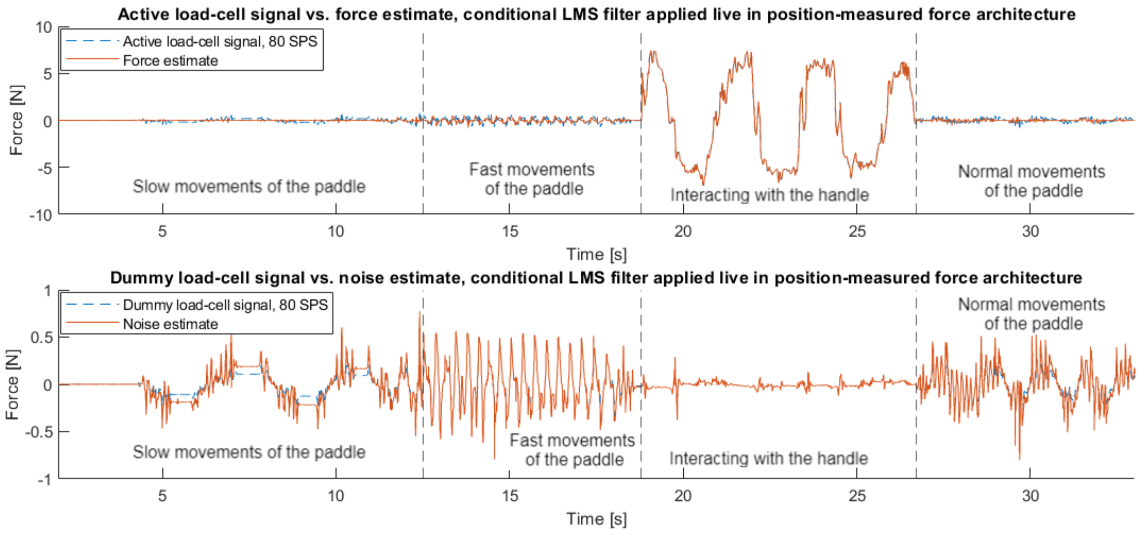


Figure 5.23: Using the LMS adaptive filter conditionally with $\mu = 1 \cdot 10^{-5}$ and $M = 2$ in the position-measured force architecture implemented live on the Arduino Uno with modified amplifiers, conducted in the robotics lab. During the slow, normal and fast movements of the paddle, the handle was not interacted with.

force and position tracking error percentages for the increased sampling rate of 80 SPS may be marginally inaccurate.

Conclusion: When modifying the load-cell amplifiers to increase their sampling rate to 80 SPS, the position tracking performance slightly increases, but the force tracking performance remains similar and the force measurement contains a lot more noise. Since this noise is environmental noise (increased in power due to the higher amplifiers' sampling rate), the LMS filter used as noise canceller is not able to attenuate it in the force estimate.

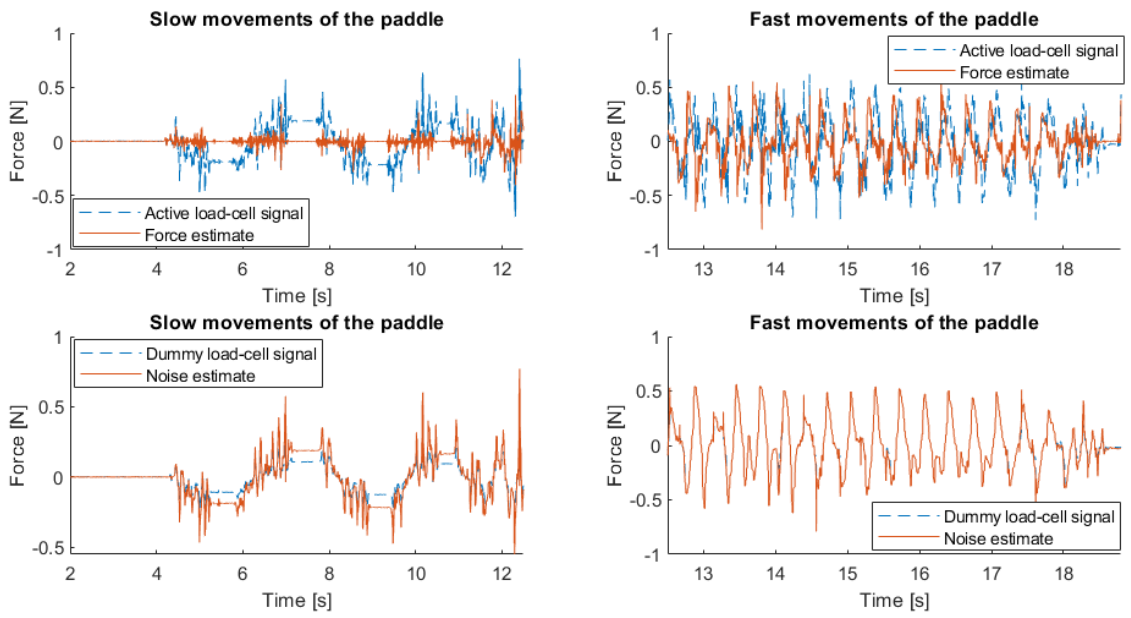


Figure 5.24: Using the LMS adaptive filter conditionally with $\mu = 1 \cdot 10^{-5}$ and $M = 2$ in the position-measured force architecture implemented live on the Arduino Uno with modified amplifiers, conducted in the robotics lab. During these measurements, no force was applied at the handle (active load-cell).

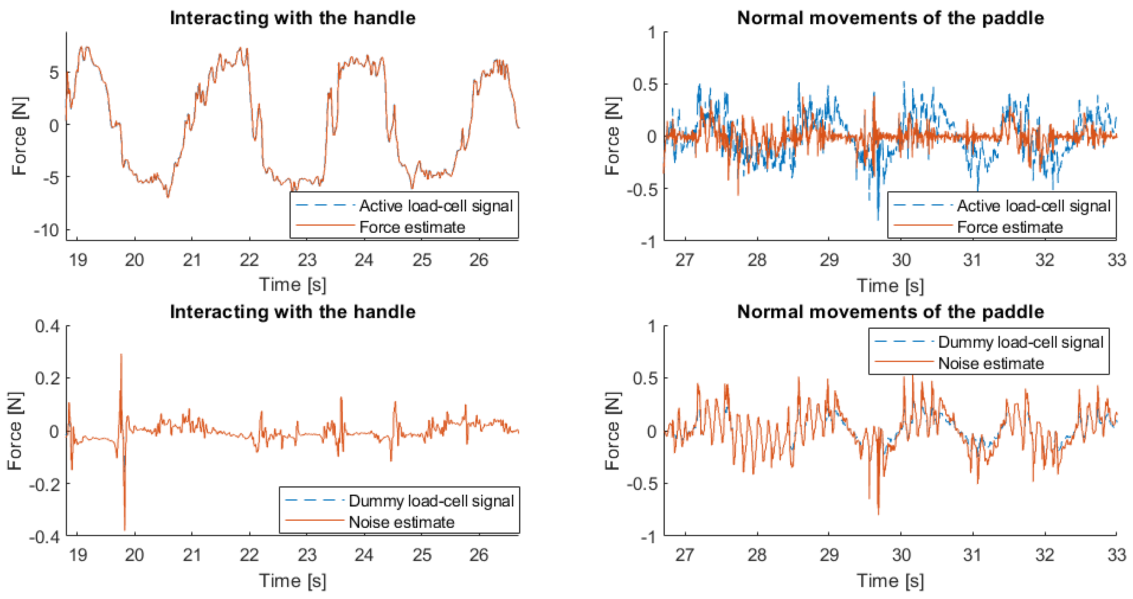


Figure 5.25: Using the LMS adaptive filter conditionally with $\mu = 1 \cdot 10^{-5}$ and $M = 2$ in the position-measured force architecture implemented live on the Arduino Uno with modified amplifiers, conducted in the robotics lab. During the normal movements of the paddle, no force was applied at the handle (active load-cell).

6 Conclusions and Recommendations

First of all, in this thesis, the quality of the force measurement has been improved significantly. While the linearity and accuracy of the load-cells were found to be sufficient, the force measurement was notably susceptible to environmental noise, the effects of gravity and inertia, and (electromagnetic) interference during teleoperation. An adaptive filter using the LMS algorithm, implemented as a noise canceller was determined to be the optimal solution for attenuating these disturbances and noise sources with varying characteristics at the desired loop frequency of 500 Hz, but this filter also distorted the clean signal: the force applied by the operator at the setup's handle (on the task side). Hence, when a force is applied at the handle, the filter has to be temporarily disabled and replaced by a basic noise estimate: subtracting the dummy-load cell signal from the active load-cell signal. With this approach, the most common disturbances and noise sources in the force measurement are attenuated significantly without distortion of the clean signal, except for short and intense disturbances.

Short disturbances are only attenuated when they are recurring, and environmental noise (in the robotics lab) is not attenuated at all. The sub-optimal performance in these cases can be attributed in part to the relatively "slow" (but computationally light) LMS algorithm, but it is mostly caused by delay between the load-cells' measurements originating from the low sampling rate of the amplifiers. When the delay between the load-cells' measurement is reduced with an increased amplifiers' sampling rate, these disturbances may also be sufficiently attenuated by the basic noise estimate. However, this should be verified with different load-cell amplifiers, as the modification of the HX711 amplifiers (to provide 80 SPS) significantly increased the noise in the force estimate.

To implement the position-measured force and 4-channel architectures, the motors' torque should be controlled. Since using the current sensor of the motor shield could not be justified due its questionable accuracy and the required extra (feedback) controller and filter with inherent delay, $K_{f,d}$ was determined to convert the measured force at the load-cell into a PWM duty cycle for the motor(s) and vice versa, thereby predicting the torque instead of controlling it. This constant was found to depend greatly on the setup's temperature and differ slightly per setup, so it was based on the measurements that were performed when the DIY kit was at its most likely temperature. Nonetheless, the dependency of $K_{f,d}$ on temperature and its deviation between setups is found to be detrimental to the force transparency and stability of the position-measured force architecture and should be even more detrimental to the (overall) transparency and stability of the 4-channel architecture.

This demonstrates the problem of using motors (and motor drivers) with inconsistent behaviour for a DIY kit that is to be assembled by the user; since the $K_{f,d}$ should represent all motors equally well, there can be a noticeable discrepancy in the force applied to the setup on the task side and the force reflected to the operator via the setup on the human side. Furthermore, even when the motors' behaviours are similar, the motor shields may limit their currents to different extends when their temperature increases, as discussed in experiment 3.2. Since this type of motor can provide way more torque than required to tilt and move the setup's base, a similarly priced motor with a lower maximum torque output (and cogging torque) but consistent and specified motor constant would benefit a future iteration of the DIY setup. Furthermore, using a motor with a lower maximum torque output should reduce its current draw (assuming the ESR is increased), which reduces the load on the motor shields and consequently the risk of them overheating and limiting the current outputs.

While the position-measured force architecture with the $K_{f,d}$ determined from the measurements significantly outperformed the position-computed force architecture in terms of overall transparency, the force tracking error was not reduced to anticipated extend. With a revised (increased) $K_{f,d}$, the force tracking performance did match the expectations, roughly halving this error while maintaining the low position tracking error. The modification of the amplifiers to provide 80 SPS did not improve the transparency in these experiments significantly. To further improve the transparency, a PD controller could be implemented instead of a proportional (P) controller. A PD controller adds a damping factor (proportional to the rate of change of the error) to reduce the oscillation resulting from high proportional gains, thereby increasing the proportional gain that can be achieved before instability starts to occur.

The overall teleoperation system was found to be stable during all experiments, but started to present oscillatory behaviour when both setups were interacted with by a single operator. When the amplifiers were modified to provide 80 SPS, this behaviour was less noticeable; the stability seemed to be improved. Hence, an increased sampling rate for the force measurement is definitely beneficial to the system's transparency and stability, but modifying the HX711 to increase the sampling rate is not the optimal solution because of the added noise in the force estimate.

Due to time constraints, the implementation of the 4-channel architecture and Processing interface has not been completed. While the Processing interface would have been a useful tool to show the effect of the proportional controller gain K_p and added time delay on the realised position-measured force architecture's transparency and stability, it was considered less crucial for the completion of this assignment. It is most unfortunate that the implementation of the 4-channel architecture was not completed, as the literature review suggested that this architecture should provide the best transparency for this DIY kit. However, this architecture is heavily reliant on the force measurements of both setups, so the performance with the load-cell amplifiers' low sampling rate may not meet these high expectations.

6.1 Answering the main research questions

To conclude, the (two) main research questions stated in Section 1.6 are to be answered, firstly:

"How can the position-measured force and 4-channel architectures be realised on the Avatars.Report DIY at Home lab setup; what are the critical design aspects and how can these be addressed using the available hardware and software resources?"

The position-measured force and 4-channel architectures can be realised on the DIY setup by determining and using a constant $K_{f,d}$ to predict the force at a setup's handle for a certain PWM duty cycle and vice versa. The critical design aspects are not merely achieving the highest transparency and stability with these control architectures, but also achieving this performance consistently across all DIY kits consisting of this (in some ways flawed) hardware, even though the hardware's parameters such as the motor constant may differ.

And secondly:

"How can a low-cost force sensor be effectively utilised for force sensing in a teleoperation system?"

To effectively utilise a low-cost force sensor (load-cell) for force sensing in a teleoperation system, an adaptive filter using a computationally light algorithm like LMS implemented as noise canceller is key. With the use of a "dummy" load-cell to which none of the relevant force is applied, this adaptive filter should be able to remove the noise from the force measurement of the "active" load-cell. However, the noise and relevant force may be too correlated, such that the adaptive filter distorts the relevant force signal. In this case, the adaptive filter should be disabled when force is applied at the handle, and the dummy load-cell signal should simply be subtracted from the active load-cell signal.

Bibliography

[1] R. C. Goertz, "Master-Slave Manipulator," 1949. [Online]. Available: https://www.osti.gov/biblio/1054625

[2] G. V. A. G. Asanka Perera and A. M. H. S. Abeykoon, "Review on bilateral teleoperation with force, position, power and impedance scaling," in *7th International Conference on Information and Automation for Sustainability*, 2014, pp. 1–7.

[3] A. Martins Oliveira and I. Bezerra Queiroz de Araujo, "A Brief Overview of Teleoperation and Its Applications," in *2023 15th IEEE International Conference on Industry Applications (INDUSCON)*, 2023, pp. 601–602.

[4] "Introduction to telemanipulation – Avatars.Report," [Online; accessed 17. May 2024]. [Online]. Available: https://avatars.report/introduction-to-telemanipulation

[5] X. Xu, B. Cizmeci, C. Schuwerk, and E. Steinbach, "Model-Mediated Teleoperation: Toward Stable and Transparent Teleoperation Systems," *IEEE Access*, vol. 4, pp. 425–449, 2016.

[6] D. Lawrence, "Stability and transparency in bilateral teleoperation," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 5, pp. 624–637, 1993. [Online]. Available: https://ieeexplore.ieee.org/document/258054

[7] B. Hannaford, "A design framework for teleoperators with kinesthetic feedback," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 4, pp. 426–434, 1989.

[8] R. Lieftink, S. Falcone, C. van der Walt, J. van Erp, and D. Dresscher, "A Pragmatic Approach to Bi-Directional Impedance Reflection Telemanipulation Control: Design and User Study," *IEEE Robotics and Automation Letters*, vol. 9, no. 4, pp. 3617–3624, 2024.

[9] G. V. A. G. Asanka Perera and A. M. H. S. Abeykoon, "Review on bilateral teleoperation with force, position, power and impedance scaling," in *7th International Conference on Information and Automation for Sustainability*, 2014, pp. 1–7.

[10] "DIY @ Home setup – Avatars.Report," [Online; accessed 17. May 2024]. [Online]. Available: https://avatars.report/diy-home-setup

[11] J. Kim, P. H. Chang, and H.-S. Park, "Two-Channel Transparency-Optimized Control Architectures in Bilateral Teleoperation With Time Delay," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 1, pp. 40–51, 2013.

[12] H. Jian, S. Zheng, C. Zhao, H. Li, and S. Wang, "Adaptive Sliding Mode Control for Random Teleoperation System with Time-Varying Delay," in *2021 40th Chinese Control Conference (CCC)*, 2021, pp. 2280–2285.

[13] W. Li, B. Zhang, and Y. Zhang, "Neural networks adaptive control for bilateral teleoperation system with uncertain dynamics and kinematics," in *2017 Chinese Automation Congress (CAC)*, 2017, pp. 1217–1222.

[14] A. Sharif Ahmadian, "Chapter 7 - Numerical Modeling and Simulation," in *Numerical Models for Submerged Breakwaters*, A. Sharif Ahmadian, Ed. Boston: Butterworth-Heinemann, 2016, pp. 109–126. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128024133000079

[15] I. G. Polushin, "A Generalized Scattering Framework for Teleoperation with Communication Delays **The research was supported by the Discovery Grants Program of the Natural Sciences and Engineering Research Council (NSERC) of Canada through grant RGPIN-2015-05753." *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 10 064–10 069, 2020, 21st IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896320334911

[16] Z. Chen, F. Huang, W. Sun, and W. Song, “An Improved Wave-Variable Based Four-Channel Control Design in Bilateral Teleoperation System for Time-Delay Compensation,” *IEEE Access*, vol. PP, pp. 1–1, 02 2018.

[17] U. Tümerdem, “A Study on the L2 Stability and Transparency of Three Channel Control Architectures in Bilateral Teleoperation under Time Delays,” *International Journal of Advances in Engineering and Pure Sciences*, vol. 33, no. 3, p. 455–466, 2021.

[18] “Device dynamics – Avatars.Report,” [Online; accessed 17. May 2024]. [Online]. Available: https://avatars.report/challenges-device-dynamics

[19] J. Orloff, “12.2: Nyquist Criterion for Stability,” *Massachusetts Institute of Technology*. [Online]. Available: https://math.libretexts.org/Bookshelves/Analysis/Complex_Variables_with_Applications_(Orloff)/12%3A_Argument_Principle/12.02%3A_Nyquist_Criterion_for_Stability

[20] “Position-Force Architecture – Avatars.Report,” [Online; accessed 17. May 2024]. [Online]. Available: https://avatars.report/position-force-architecture/

[21] M. Panzirsch, T. Hulin, J. Artigas, C. Ott, and M. Ferre, “Integrating Measured Force Feedback in Passive Multilateral Teleoperation,” vol. 9774, 07 2016, pp. 316–326. [Online]. Available: https://oa.upm.es/46691/1/INVE_MEM_2016_253472.pdf

[22] H. Lau and L. Wai, “Implementation of position–force and position–position teleoperator controllers with cable-driven mechanisms,” *Robotics and Computer-Integrated Manufacturing*, vol. 21, no. 2, pp. 145–152, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584504000730

[23] N. Marcassus, A. Chriette, and M. Gautier, “Theoretical and experimental overview of bilateral teleoperation control laws,” in *2006 14th Mediterranean Conference on Control and Automation*, 2006, pp. 1–6.

[24] J. F. Broenink, *Introduction to Physical Systems Modelling with Bond Graphs*. University of Twente, Dept EE, Control Laboratory, 1999.

[25] R. Beerens, D. Heck, A. Saccon, and H. Nijmeijer, “The Effect of Controller Design on Delayed Bilateral Teleoperation Performance: An Experimental Comparison,” *IEEE Transactions on Control Systems Technology*, vol. 28, no. 5, pp. 1727–1740, 2020.

[26] I. Aliaga, A. Rubio, and E. Sanchez, “Experimental quantitative comparison of different control architectures for master-slave teleoperation,” *IEEE Transactions on Control Systems Technology*, vol. 12, no. 1, pp. 2–11, 2004.

[27] J. Holtz, “Pulsewidth modulation for electronic power conversion,” *Proceedings of the IEEE*, vol. 82, no. 8, pp. 1194–1214, 1994.

[28] “Simulate pulse width modulation (PWM) output from hardware - Simulink - MathWorks,” [Online; accessed 3. Jun. 2024]. [Online]. Available: https://nl.mathworks.com/help/soc/ref/pwminterface.html

[29] D. Dresscher, “Applying the control theory with PWM steer,” [Online; accessed 15. Jun. 2024]. [Online]. Available: https://docs.google.com/document/d/1kInoAYRpwnOTtB-SHGQ_CThJarsBH_DJBVMTsG3yUkU/edit#heading=h.8ojpwygdrox0

[30] Z. Li, X. Li, J. Lin, Y. Pang, D. Yang, L. Zhong, and J. Guo, “Design and Application of Multidimensional Force/Torque Sensors in Surgical Robots: A Review,” *IEEE Sensors Journal*, vol. 23, no. 12, pp. 12 441–12 454, 2023.

[31] W. Hernandez, “Improving the Response of a Load Cell by Using Optimal Filtering,” *Sensors*, vol. 6, no. 7, pp. 697–711, 2006. [Online]. Available: https://www.mdpi.com/1424-8220/6/7/697

[32] K. Hoffmann, *Applying the wheatstone bridge circuit*. HBM Darmstadt, Germany, 1974. [Online]. Available: http://www.eln.teilam.gr/sites/default/files/Wheatstone%20bridge.

pdf

[33] L. Shi, X. Wei, and Y. Wei, "Comparative analysis on piezoelectric effect and electromagnetic effect for the further study of multi-piezoelectric effect," in *Proceedings of 2012 International Conference on Measurement, Information and Control*, vol. 2, 2012, pp. 907–911.

[34] D. G. Black, A. H. H. Hosseinabadi, and S. E. Salcudean, "6-DOF Force Sensing for the Master Tool Manipulator of the da Vinci Surgical System," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2264–2271, 2020.

[35] C. González, J. E. Solanes, A. Muñoz, L. Gracia, V. Girbés-Juan, and J. Tornero, "Advanced teleoperation and control system for industrial robots based on augmented virtuality and haptic feedback," *Journal of Manufacturing Systems*, vol. 59, pp. 283–298, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612521000492

[36] A. Bolopion and S. Régnier, "A Review of Haptic Feedback Teleoperation Systems for Micromanipulation and Microassembly," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 3, pp. 496–502, 2013.

[37] A. M. Ousaid, D. S. Haliyo, S. Régnier, and V. Hayward, "A Stable and Transparent Microscale Force Feedback Teleoperation System," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 5, pp. 2593–2603, 2015.

[38] R. V. Patel, S. F. Atashzar, and M. Tavakoli, "Haptic Feedback and Force-Based Teleoperation in Surgical Robotics," *Proceedings of the IEEE*, vol. 110, no. 7, pp. 1012–1027, 2022.

[39] S. A. M. Dehghan, H. R. Koofigar, and M. Ekramian, "Nonlinear bilateral control of 4-channel teleoperation systems using adaptive force estimator," in *2016 20th International Conference on System Theory, Control and Computing (ICSTCC)*, 2016, pp. 102–106.

[40] S. Haykin, *Adaptive Filter Theory*, ser. Prentice-Hall information and system sciences series. Prentice Hall, 2002. [Online]. Available: https://books.google.nl/books?id=leuzQgAACAAJ

[41] H. Quanzhen, G. Zhiyuan, G. Shouwei, S. Yong, and Z. Xiaojin, "Comparison of LMS and RLS Algorithm for Active Vibration Control of Smart Structures," in *2011 Third International Conference on Measuring Technology and Mechatronics Automation*, vol. 1, 2011, pp. 745–748.

[42] S. Safapourhajari, "Lecture 6 - Adaptive Filters II," Mar. 2022. [Online]. Available: https://canvas.utwente.nl/courses/14157/files/4001374?module_item_id=458033

[43] Massachusetts Institute of Technology, Department of Mechanical Engineering, "Introduction to Recursive-Least-Squares (RLS) Adaptive Filters," 2008. [Online]. Available: https://ocw.mit.edu/courses/2-161-signal-processing-continuous-and-discrete-fall-2008/0908e93cb4f31eac9150b14c1ffdf328_rls.pdf

[44] F. Bosman, "Hardware for DIY at Home Lab for Telemanipulation Control," 2023. [Online]. Available: https://cloud.ram.eemcs.utwente.nl/index.php/s/mwxQszQWbpHWMta

[45] "High-speed High-torque 775 DC Motor," [Online; accessed 21. May 2024]. [Online]. Available: https://www.benselectronics.nl/high-speed-high-torque-775-dc-motor.html

[46] "Arduino Motor Shield Rev3," [Online; accessed 21. May 2024]. [Online]. Available: https://store.arduino.cc/products/arduino-motor-shield-rev3

[47] M. Riddle, J. MacDermid, S. Robinson, M. Szekeres, L. Ferreira, and E. Lalone, "Evaluation of individual finger forces during activities of daily living in healthy individuals and those with hand arthritis," *Journal of Hand Therapy*, vol. 33, no. 2, pp. 188–197, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S089411302030079X

[48] M. Orta Martinez, C. M. Nunez, T. Liao, T. K. Morimoto, and A. M. Okamura, "Evolution and Analysis of Hapkit: An Open-Source Haptic Device for Educational Applications," *IEEE Transactions on Haptics*, vol. 13, no. 2, pp. 354–367, 2020.

[49] "AS5048: 14-bit Rotary Position Sensor with Digital Angle (Interface) and PWM Output," Jan. 2014, [Online; accessed 21. May 2024]. [Online]. Available: https://docs.rs-online.com/0657/A700000006921305.pdf

[50] F. van den Boom, "Software for DIY at Home Lab for Telemanipulation Control," 2023. [Online]. Available: https://cloud.ram.eemcs.utwente.nl/index.php/s/24gGc2radgpoaet

[51] "Arduino Uno Rev3," [Online; accessed 21. May 2024]. [Online]. Available: https://store.arduino.cc/products/arduino-uno-rev3?variant=35453920903319

[52] "Encoder - Arduino Reference," [Online; accessed 21. May 2024]. [Online]. Available: https://www.arduino.cc/reference/en/libraries/encoder

[53] "Arduino Motor Shield Rev3 Schematic," [Online; accessed 27. May 2024]. [Online]. Available: https://www.arduino.cc/en/uploads/Main/arduino_MotorShield_Rev3-schematic.pdf

[54] T. Instruments, "LMV3xx-N/-Q1 Single, Dual, and Quad General Purpose, Low-Voltage, Rail-to-Rail Output Operational Amplifiers," Aug. 2020. [Online]. Available: https://www.ti.com/lit/ds/symlink/lmv358-n.pdf?ts=1716722008621&ref_url=https%253A%252F%252Fwww.mouser.sk%252F

[55] "Current Sensing using Arduino Motor Shield L298N L298P - Robojax," [Online; accessed 27. May 2024]. [Online]. Available: http://robojax.com/learn/arduino/?vid=robojax-motor-shield-current-sensing

[56] "Digital signal processing and filtering – motor current sensing – Dustyn Robots," Jun. 2012, [Online; accessed 27. May 2024]. [Online]. Available: https://dustynrobots.com/academia/research/digital-signal-processing-and-filtering-motor-current-sensing

[57] A. MicroSystems, "ACS712 - Fully Integrated, Hall Effect-Based Linear Current Sensor IC with 2.1 kVRMS Isolation and a Low-Resistance Current Conductor," 2013. [Online]. Available: https://www.farnell.com/datasheets/1759100.pdf

[58] TheBoredRobot, "YouTube-Code / Post 22 - Real-Time Data Visualization and CSV Logging with Arduino and Python," Dec. 2023, [Online; accessed 30. May 2024]. [Online]. Available: https://github.com/TheBoredRobot/YouTube-Code/tree/main/Post%2022%20-%20Real-Time%20Data%20Visualization%20and%20CSV%20Logging%20with%20Arduino%20and%20Python

[59] I. Virgala and M. Kenderova, Peter Frankovsky, "Friction Effect Analysis of a DC Motor," *American Journal of Mechanical Engineering*, vol. 1, no. 1, pp. 1–5, 2013. [Online]. Available: http://pubs.sciepub.com/ajme/1/1/1

[60] A. Hughes and B. Drury, "Chapter 3 - D.C. motors," in *Electric Motors and Drives (Fifth Edition)*, fifth edition ed., A. Hughes and B. Drury, Eds. Newnes, 2019, pp. 89–129. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780081026151000039

[61] maxon, "RE 36 ø36 mm, Graphite Brushes, 70 Watt," Apr. 2005. [Online]. Available: https://docs.rs-online.com/ff9d/0900766b8002ac29.pdf

[62] Opencircuit, "2KG weeg sensor - Opencircuit," May 2024, [Online; accessed 10. May 2024]. [Online]. Available: https://opencircuit.nl/product/2kg-weeg-sensor

[63] ——, "2KG gewicht sensor handleiding," 2024, [Online; accessed 10. May 2024]. [Online]. Available: https://cdn.bodanius.com/media/1/fc41430_gewicht-sensor-handleiding.pdf

[64] G. Stano, A. Di Nisio, A. Lanzolla, and G. Percoco, "Additive manufacturing and characterization of a load cell with embedded strain gauges," *Precision Engineering*, vol. 62, pp. 113–120, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141635919305033

[65] AVIA Semiconductor, “HX711 24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales,” [Online; accessed 11. May 2024]. [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf

[66] R. Tillaart, “ GitHub - RobTillaart/HX711: Arduino library for HX711 24 bit ADC used for load cells and scales,” 2023. [Online]. Available: https://github.com/RobTillaart/HX711

[67] P. Machado, L. Fachini, V. Machado, R. Szmoski, and T. Antonini Alves, “Load cells calibration with a low cost data acquisition system,” *Revista Brasileira de Física Tecnológica Aplicada*, vol. 6, 04 2019.

[68] wespo, “LMS,” Jun. 2024, [Online; accessed 3. Jun. 2024]. [Online]. Available: https://github.com/wespo/LMS

[69] M. Mekhfioui, R. Elgouri, A. Satif, M. Moumouh, and H. Laâmari, “Implementation Of Least Mean Square Algorithm Using Arduino Simulink,” *International Journal of Scientific Technology Research*, vol. 9, pp. 664–667, 04 2020.

[70] I. T. Jolliffe and J. Cadima, “Principal component analysis: a review and recent developments,” *Philosophical Transactions of the Royal Society A*, vol. 374, no. 2065, p. 20150202, 2016.

[71] “SparkFun Load Cell Amplifier - HX711 - SEN-13879 - SparkFun Electronics,” [Online; accessed 24. Jun. 2024]. [Online]. Available: https://www.sparkfun.com/products/13879