Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Deutsches Zentrum
für Luft- und Raumfahrt
German Aerospace Center

Adaptive Model Mediated Teleoperation Using Reinforcement Learning

Master thesis
in German Aeroscpace Center, DLR
Institute of Robotics and Mechatronics
Modular Dexterous Robotics (Modex) Lab

Universität Hamburg
MIN-Fakultät
Department Informatik

Author
Hadi Beik-Mohammadi
01.01.2020

Gutachter:	Prof. Dr. S. Wermter (UHH)
	Dr. Matthias Kerzel (UHH)
Betreuung:	Dr. Neal Y. Lii (DLR)

Hadi Beik-Mohammadi
Matrikelnummer: 6885407
Fritz-Bauer-Str 22
81249 München

Abstract

Due to similarities in learning techniques, Reinforcement Learning (RL) is the closest alternative to human-level intelligence. Teleoperation systems using RL can adapt to new environmental conditions and deal with high uncertainty due to long-time delays. In this thesis, we propose a method that takes advantage of RL capabilities to extend the human reach in dangerous remote environments. The proposed method utilizes the Model Mediated Teleoperation (MMT) concept in which the teleoperator interacts with a simulated setup that resembles the real environment. The simulation can provide instant haptic feedback where the data from the real environment are delayed. The proposed approach enables haptic feedback teleoperation of high-DOF dexterous robots under long time delays in a time-varying environment with high uncertainty.

In existence of time delay, when the data is received by the remote system the environment may change drastically, therefore, the attempt for task execution will fail. To prevent failure, an intelligence system is realized in two layers, the first layer utilizes the Dynamic Movement Primitives (DMP) which accounts for certain changes in the environment. DMPs can adjust the shape of a trajectory based on given criteria, for example, a new target position or avoiding a new obstacle. But in an uncertain environment, DMPs fail, therefore, the second layer of intelligence makes use of different reinforcement learning methods based on expectation-maximization, stochastic optimal control and policy gradient to guarantee the successful completion of the task.

Furthermore, To ensure the safety of the system, and speed up the learning process, each learning session for RL happens in multiple simulations of the remote system and environment, simultaneously.

The proposed approach was realized on DLR's haptic hand-arm user interface/exoskeleton, Exodex Adam. It has been used for the first time in this work as the master device to teleoperate a high-DOF dexterous robot. This slave device is an anthropomorphic hand-arm system combining a five-finger hand (FFH) attached to a custom configured DLR lightweight robot (LWR 4+) more closely fitting to the kinematics of the human arm. An augmented reality visualization implemented on the Microsoft Hololens fuses the slave device and virtual environment models to provide environment immersion for the teleoperator.

A preliminary user-study was carried out to help evaluate the human-robot interaction capabilities and performance of the system. Meanwhile, the RL approaches are evaluated separately in two different levels of difficulty; with and without uncertainty in perceived object position.

The results from the unweighted NASA Task load Index (NASA TLX) and System Usability Score (SUS) questionnaires show a low workload (27) and above-average perceived usability (71). The learning results show all RL methods can find a solution for all challenges in a limited time. Meanwhile, the method based on stochastic optimal control has a better performance. The results also show DMPs to be effective at adapting to new conditions where there is no uncertainty involved.

Zusammenfassung

Aufgrund von Ähnlichkeiten in den Lerntechniken ist Reinforcement Learning (RL) die nächstliegende Alternative zu menschlicher Intelligenz. Teleoperationssysteme, die RL verwenden, können sich an neue Umgebungsbedingungen anpassen und mit hoher Unsicherheit aufgrund von langen Zeitverzögerungen umgehen. In dieser Arbeit schlagen wir eine Methode vor, die sich die Fähigkeiten von RL zunutze macht, um die menschliche Reichweite in gefährlichen, abgelegenen Umgebungen zu erweitern. Die vorgeschlagene Methode nutzt das Konzept der Model Mediated Teleoperation (MMT), bei der der Teleoperator mit einer simulierten Umgebung interagiert, die der realen Umgebung ähnelt. Die simulierte Umgebung kann ein sofortiges haptisches Feedback liefern, während das Feedback aus der realen Umgebung verzögert ist. Der vorgeschlagene Ansatz ermöglicht die haptisches Feedback für Teleoperation von Robotern mit vielen Freiheitsgraden (DOF) unter langen Zeitverzögerungen in einer zeitvarianten Umgebung mit hoher Unsicherheit.
Bei einer Zeitverzögerung, wenn die Daten vom entfernten System empfangen werden, kann sich die Umgebung drastisch ändern, weshalb der Versuch der Aufgabenausführung fehlschlagen kann. Um ein Scheitern zu verhindern, ist ein intelligentes System in zwei Ebenen realisiert, wobei die erste Ebene Dynamic Movement Primitives (DMP) verwendet, die bestimmte Änderungen in der Umgebung berücksichtigen un konpensieren können. DMPs können die Form einer Trajektorie nach vorgegebenen Kriterien anpassen, z.B. eine neue Zielposition oder die Vermeidung eines neuen Hindernisses. Aber in einem unsicheren Umfeld versagen DMPs, daher nutzt die zweite Ebene verschiedene Methoden des Reinforcement Learnings, die auf Erwartungs-Maximierung, stochastischer optimaler Steuerung und Policy Gradient basieren, um die erfolgreiche Bewältigung der Aufgabe zu garantieren.
Darüber hinaus, um die Sicherheit des Systems zu gewährleisten und den Lernprozess zu beschleunigen, findet jede Lernepisode für RL in mehreren Simulationen des entfernten Systems und der Umgebung gleichzeitig statt. Der vorgeschlagene Ansatz wurde auf der haptischen Hand-Arm-Benutzeroberfläche/Exoskelett des DLR, Exodex Adam, realisiert. Er wurde in dieser Arbeit erstmals als Mastergerät zur Teleoperation eines hoch-DOF-fähigen Roboters eingesetzt. Bei diesem Slave-Gerät handelt es sich um ein anthropomorphes Hand-Arm-System, bei dem eine Fünf-Finger-Hand (FFH) an einem speziell konfigurierten DLR-Leichtbauroboter (LWR 4+) angebracht ist, der sich eng an die Kinematik des menschlichen Arms anlehnt. Eine auf dem Microsoft Hololens implementierte Augmented-Reality-Visualisierung verschmilzt das Slave-Gerät und virtuelle Umgebungsmodelle, um dem Teleoperator eine Immersion in die Umgebung zu ermöglichen.
Eine vorläufige Benutzerstudie wurde durchgeführt, um die Fähigkeiten und die Leistung des Systems in Bezug auf die Mensch-Roboter-Interaktion zu bewerten. Zugleich werden die RL-Ansätze getrennt in zwei verschiedenen Schwierigkeitsgraden bewertet; mit und ohne Unsicherheit in der wahrgenommenen Objektposition.
Die Ergebnisse aus den ungewichteten Fragebögen NASA Task Load Index (NASA

TLX) und System Usability Score (SUS) zeigen eine geringe Arbeitsbelastung (27) und eine überdurchschnittlich hohe wahrgenommene Benutzerfreundlichkeit (71). Die Lernergebnisse zeigen, dass alle RL-Methoden in kurzer Zeit eine Lösung für alle Herausforderungen finden können. Die auf stochastischer optimaler Steuerung basierende Methode erziehlt dabei die beste Performance. Die Ergebnisse zeigen auch, dass sich DMPs effektiv an neue Bedingungen anpassen können, wenn keine Unsicherheit besteht.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

In long-distance teleoperation, signals can take minutes to reach the teleoperator. The time delay introduces inconsistency or mismatch between master and slave that falsifies the provided feedbacks. For instance, the teleoperator receives the haptic feedback when the slave robot has already collided with an object in the remote environment. The reaction to faulty feedback might put the robot at a higher risk. Additionally, the visual information provided to the operator refers to the past which is no longer relevant and correct. In a grasping scenario, if during data transmission the object moves the teleoperation fails. Therefore, providing real-time data from the remote environment is undesirable.
Approaches such as passivity-based control [49, 4, 48, 19] and predictive display [9, 7] can reduce the inconsistency but the problems appear when the time delay increases or becomes variable. A unique way of approaching the time delay problem is introduced by Model Mediated Teleoperation (MMT) [53, 85]. MMT uses a copy or a simulation of the remote environment at the master-side to provide instant feedback for the teleoperator. It has been used in several areas, for example surgical teleoperation [40], space robot teleoperation [51].
MMT requires an accurate model of the remote environment and slave device. Modeling a non-linear time-variant system so-called remote environment is not an easy task. The MMT approaches attempt to estimate/predict/approximate the environment model using different approaches such as neural networks [82] which suffer from complexity and lack of determinism.

1.2 Objective and Approach

To tackle time delay issues, an approach with characteristics such as instant visual and haptic feedback as well as intelligence to adapt to new environmental conditions is expected. The slave side intelligence must compensate for errors and mistakes in the demonstration, therefore, discarding the need for an accurate model of the remote environment on the master-side. Accordingly, providing in-

stant haptic and visual feedback does not require the precise future state of the environment and a simple approximation is sufficient.
The approach used in this thesis adopts the architecture from the MMT scheme and combines it with Dynamic Movement Primitives (DMP) [64] and model-free Reinforcement Learning (RL) [78, 76, 61]. The slave system uses DMPs to account for certain changes in the model by transforming the trajectory into a non-linear dynamical system. The new form of the trajectory can be modified on the spot, the changes such as the shape as well as the start/target of the trajectory can be readjusted almost instantly. To increase the system intelligence the slave should be able to adapt to uncertainties in the model that cannot be considered before execution. The policy search RL methods are designed to explore and search for a viable solution in a limited time. Therefore, RL agents can complement DMPs and ensure that the task is completed successfully. Three RL methods are evaluated, PoWER based on expectation maximization, eNAC based on policy gradient, and PI^2 based on stochastic optimal control.

1.3 Major Contribution

This study examines two main hypothesis:

- Integrating Dynamic Movement Primitives and Reinforcement Learning methods into a Model Mediated Teleoperation architecture can effectively compensate for the changes of the trajectory in a dynamic environment.
- Goal learning using Reinforcement Learning can effectively account for the human error and uncertainty that are presented in a Model Mediated Teleoperation architecture.

This study includes the design and implementation of the following:

- A complete Model Mediated Teleoperation pipeline using the DLR Exodex Adam hand-arm haptic interface.
- A physics simulation for the slave, master devices and the remote environment.
- An augmented reality visualization of the teleoperation scenario.

1.4 Thesis Structure

In chapter 2 "Background and Related work", we begin by presenting a brief but essential background on RL and MMT. In chapter 3 "Methodology", we proceed to describe the approach and detail description of proposed MMT architecture. In chapter 4 "Implementation and Evaluation", we describe the design and implantation of necessary modules to provide a framework for experiments. In chapter 5, "Results and Discussion" the results are illustrated and a detail explanation is provided. Finally, in Chapter 6, "Conclusion and Future work", we give a conclusion and discuss future work.

Chapter 2

Background and Related Work

2.1 Teleoperation for object manipulation and remote task execution

Teleoperation provides a framework for an operator to interact with a remote environment using an intermediate device. The intermediate device is combined of two interconnected parts so-called master and slave. Using the master device, the operator remotely controls the slave through a communication channel. Although the slave is assumed to be passive, due to the bilateral control scheme it affects the master device by position or force feedback. The bilateral control scheme provides feedback to the operator which augments the remote environment to be perceived by the operator's senses such as haptic, visual, and auditory.
Grasping and object manipulation can be realized in different ways, the robot configurations can be pre-recorded and used as a look-up table. The corresponding configurations can be selected based on the closest object position to the current position [46]. On the other hand, to reduce the inconsistency between configurations end-to-end deep learning with pre-trained neural networks [45] can be used instead of hard-coded lookup tables. Furthermore, online deep reinforcement learning can be used as an online solution where the agent learns to reach for grasp by interacting with the environment [17, 18, 6, 26, 27]. The idea of learning from demonstration (LfD) can increase the stability and performance of the grasping and in-hand manipulation [76, 41].
In practice, a teleoperation scenario may take place in different levels of abstraction, automation, and shared autonomy. Using a task-driven approach with gesture recognition, it has been demonstrated that in-hand manipulation can be effectively carried out for all six possible DOF, on free and partially constrained objects [33]. DLR and ESA's METERON SUPVIS Justin project [66, 68, 67] was a teleoperation mission with high-level abstraction and complete task autonomy where an astronaut in orbit commands a robot on the ground to execute a task. The task also can be a combination of primitive tasks such as pulling, pushing, grasping, in-hand-manipulation and placing an object. The astronauts were briefed about

how to demonstrate the task in a high-level graphical user interface but none of them were informed about how the tasks were deployed on the low-level motion planners. In this case, the task performance relies mainly on slave's intelligence while low-level manipulation tasks rely on the operator's skills, prior knowledge, and sufficient sensory feedback.
On the other hand, teleoperators continue to demand immersive user experience with haptic feedback of force reflection to feel the object and environment through telerobotics. Various work has been carried out to study force reflection in-hand telepresence, including an exoskeleton system that uses neural network [14].
Furthermore, previous work [32] has evaluated and shown the effectiveness of haptic feedback in increasing in-hand teleoperation performance, as compared to other feedback conditions. By considering the viscosity of the environment, a telepresence system can be extended to interact in a mixed media environment of fluid, gas, and solids up to the fingertips [69].
The bilateral teleoperation architecture usually contains 2-channel (e.g. position-force) [16] or 4-channel (position-force-position-force) [12] which ensures the consistency between two models and the safety of the slave. For example, if during a task an object in the remote environment hinders the slave movement, instant force feedback can inform the operator to prevent colliding with an object or stop penetrating to a hard surface. Instant force feedback requires high bandwidth communication channels with no time delay.
But, teleoperation missions usually involve transmitting data with time delay. The duration of the time delay depends on distance, communication bandwidth, network buffering, processing on relay stations or other factors. Therefore, the feedback may reach the operator with long time delays. The latency varies in different applications which can last a few milliseconds up to hundreds of seconds. For example, interplanetary teleoperation from earth to mars takes between 13 to 22 minutes depending on the planets' relative positions. Time-delay can cause telepresence to become unstable, particularly for high-DOF, long-delay systems. Kontur-2 [5], METERON Haptics-1 and Interact experiments[34] have all tackled this with increasing levels of success from 20 msec to 800msec+ timed delays.
As figure 2.3 shows the level of abstraction and the time delay are important factors to categorize different teleoperation schemes. The high bandwidth force feedback methods which provide high-quality instant force feedback usually use the force or position values as the feedback without any intermediate transformation. The passivity based approaches are shown in figure 2.2 focus on the lossless transmission of energy and stability of a passive system where an energy variable (e.g. velocity) gets transformed into wave-variables bidirectionally [49, 4, 48, 19]. But, passivity like most approaches suffers from instability where the time delay is too long and variable and/or when the remote environment changes throughout the time, stochastically.
The third common approaches use a predictive display to predict the visual clues to compensate for the time delay [7]. Authors in [9] use an estimator to predict the state of an unmanned vehicle by using the dynamics of the vehicle. The estimator uses a feed-forward and feed-back way simultaneously, to compensate

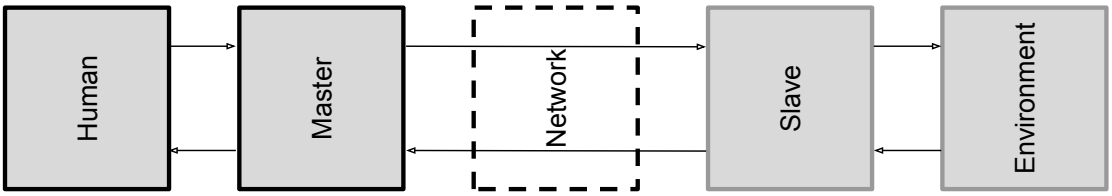


Figure 2.1: The figure shows the general structure of a telemanipulation system with five elements; human operator, a master device, communication channel (network), slave device and remote environment. A human operator using the master device controls/commands the slave device to interact with/adjust the remote environment. This operation usually occurs through a network (e.g. satellites, relay stations, etc) and is usually slowed down due to bandwidth, network buffering, processing on relay stations or other reasons [82]

.

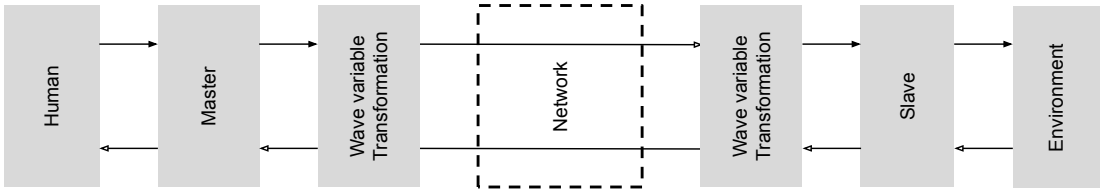


Figure 2.2: The figure illustrates a simplified architecture for Passivity based teleoperation using wave variables. In this approach, the forces and positions are transformed into wave-variables and send to the slave side and transformed back to the forces and positions before being deployed on the slave device. The same procedure is applied for force and position feedback to the master device.

for the drifting of the vehicle since the estimator does not consider the terrain shape. Using the estimated state the predictive display manipulated the raw video for the operator. The predictive display has been studied in different areas such as surgery [60] or space robotics [55].

2.1.1 Model Mediated Control (MMT)

One of the first foundations of control in the presence of time delay was introduced by Otto Smith, so-called Smith predictor [72]. As figure 2.4 shows this predictor compensates for the inconsistency caused by the time delay, by a closed-loop transfer function (the blue loop). The transfer function contains a predictor that resembles the real plant. Considering the transfer function without time delay defined as,

$$K(z) = \frac{D(z)G(z)}{1 + D(z)G(z)}, \tag{2.1}$$

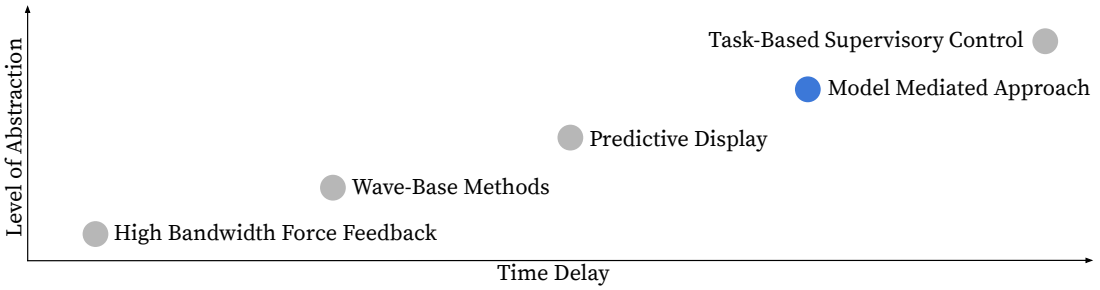


Figure 2.3: The figure shows a different level of abstraction in teleoperation inspired by [43]. The vertical axis shows the level of abstraction and the horizontal axis shows the time delay. But, by increasing the abstraction and use task-based supervisory control the operator instead of using motion command, can deploy complicated high-level commands such as "Grasp the Red Box". Due to high time delay, the feed-backs are usually augmented based on a prediction/approximation of the environment and the slave device.

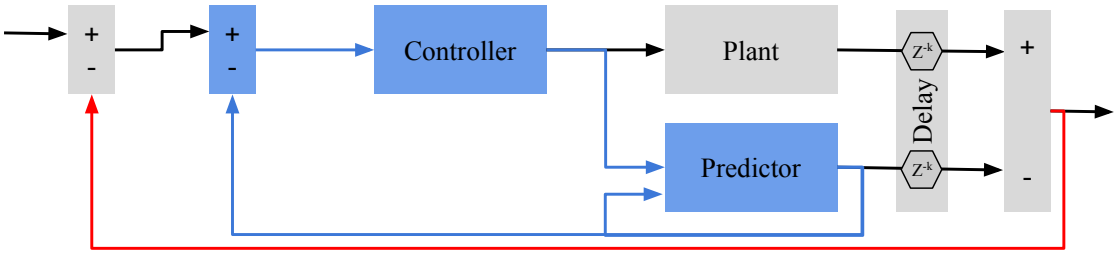


Figure 2.4: The figure shows an extended version of the smith predictor that compensates the time delay of the plant by replacing the predictor inspired by [82]. The predictor estimates the output of the plant given the input generated by the controller (blue loop). Furthermore, the outer loop of the architecture is responsible to compensate for the model mismatch between the plant and the predictor at each time step.

where D(z) and G(z) are the controller and the plant, correspondingly. The time delay can be integrated in the transfer function,

$$\hat{K}(z) = \frac{\hat{D}(z)G(z)z^{-k}}{1 + \hat{D}(z)G(z)z^{-k}}. \tag{2.2}$$

By designing the controller $\hat{D}$ in a way that $\hat{K}(z) = z^{-k}K(z)$ the output of the predicted controllers matches the real system with a delay of k[82]. The figure 2.4 shows an extended version of the smith predictor in which a second loop (red loop) is added to compensate for the modeling errors in the predictor. Without the second loop, the errors in the predictor never get fixed and it might lead to accumulated error in the predicted model. In conclusion, since the approach stabilizes the linear time-invariant systems [85], it is not beneficial for the nonlinear dynamic system which performs tasks in a time-variant environment.

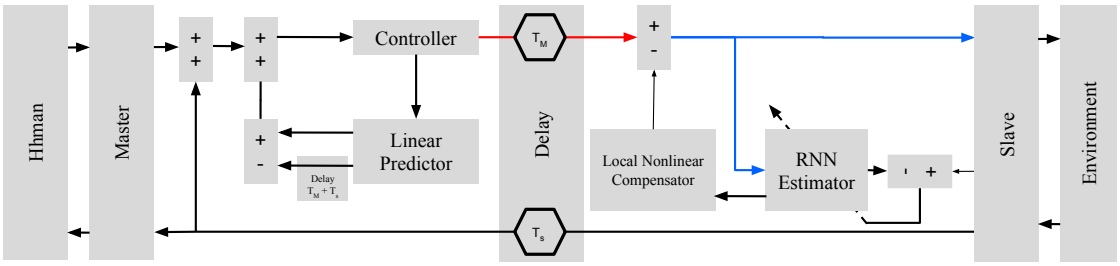


Figure 2.5: The figure shows the architecture of a teleoperation system where an RNN estimates the behavior of the non-linear plant [82]. The local non-linear compensator eliminates the non-linear part of the delayed controller output. The linear model is deployed on the master side to provide instant feedback for the operator.

The MMT approaches use an intermediate model of the remote environment to provide instantaneous sensory feedback for the operator. The intermediate model predicts the real environment in which the operator can instantly react, therefore the accuracy of the model guarantees the safety of the slave device and remote environment. In [54], an approach is introduced to synchronize the slave and the intermediate model while keeping master and slave responsive. The experiment was successfully deployed and evaluated on one DOF slave device.
In [44], second sensory feedback is provided for the master where a laser proximity sensor predicts the future contacts. Therefore, the collisions are easily detected using the secondary sensor instead of the unnecessary synchronization of the complete model. The approach has been successfully deployed and evaluated in one DOF. On the other hand, in [2], a hybrid approach is proposed where a combination of classical Kelvin-Voigt model and the nonlinear Hunt-Crossley model is used to estimate the remote environment with six DOF teleoperation system under negligible time delay. As the figure 2.5 shows an RNN based approach where the neural network approach is used to extend the smith predictor to non-linear systems [23]. The architecture divides the nonlinear model of the slave-environment into three parts,

- a linear predictor which works as a smith predictor and simulates the linear behavior as a predictive controller on the master side,
- an RNN estimator to approximate the dynamics of the non-linear plant,
- a non-linear compensator to remove the non-linear effects from the controller output.

The communication channels are assumed to have two different delays from the master to slave T_M and from slave to master T_S. An unknown system is modeled using RNN,

$$\dot{x}(t) = A_s x(t) + bW^T \phi(x(t)) + bu(t) + bd(t) + \epsilon(t), \tag{2.3}$$

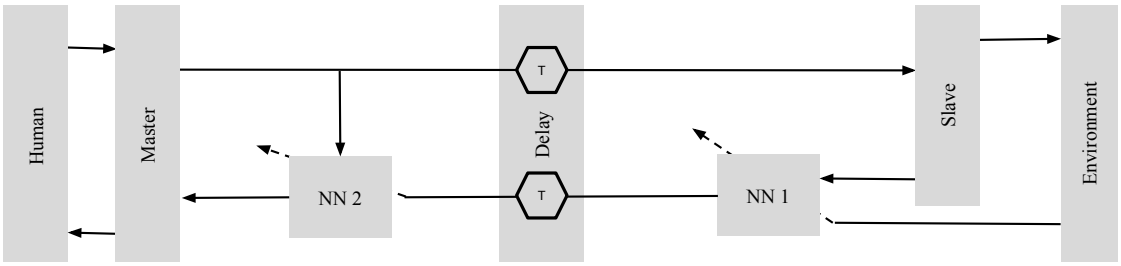


Figure 2.6: The figure shows the teleoperation architecture proposed by [71] where, two neural networks are deployed to estimate the force feed-back from the remote environment

where $W^T\phi(x(t))$ is the non-linear compensation that should be estimated by the RNN, and the locally compensated non-linearity,

$$u(t) = \tau(t - T) - \hat{W}^T\phi(x(t)). \tag{2.4}$$

Where $\tau(t - T)$ is the output of the controller based on the linear predictor after time delay T (red arrow in figure 2.5). Before the controller output enters the plant, the compensator adds the non-linear terms (blue arrow).
One might also consider replacing the linear predictor with a neural network model, the authors in [71] have used two neural networks as shown in the figure 2.6, NN1 estimates the force feedback f_e from the slave environment and the input of the network is the $x, \dot{x}, \ddot{x}$. Then the weights of the NN1 are transferred to NN2 to be used as a locally simulated environment for the master controller.
Different studies have investigated the machine learning approaches to deal with the time delay [59, 21, 15].

2.2 Reinforcement Learning

Reinforcement learning (RL) refers to a branch of machine learning that utilizes a bio-inspired reward-punishment technique. The RL agent uses the Markov Decision Process (MDP) to execute the actions in which it has received or likely to receive a reward. The reward is task-dependent and can be instantaneous or delayed. Consequently, the MDP determines a time trade-off to maximize the potential future reward. As a result, the RL agent may ignore small rewards and execute several actions with no return to receive a bigger reward at the end. The figure 2.7 depicts the agent-environment interaction used in MDP where the RL agent interacts with an environment e where the actions are followed by their associated positive or negative returns. The MDP is defined by its action-space $a \in A$, state-space $s \in S$, transition dynamics $P(s_{t+1}|s_t, a_t)$, reward function $r(s_t, a_t)$ and its initial state probability $\mu_0(s)$. The MDP principle assumes the information about the current action a_t and state s_t is enough to estimate the next state s_{t+1} and/or future reward r_{t+1} [77].

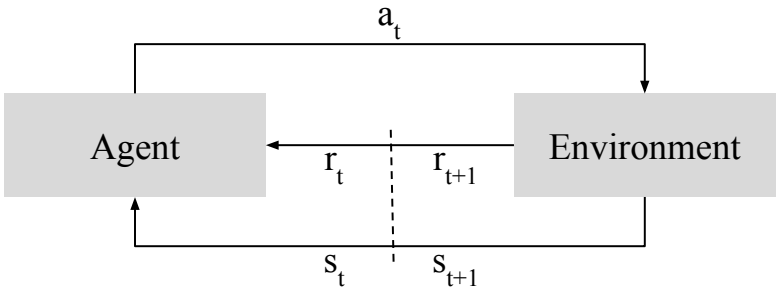


Figure 2.7: The figure illustrates the Markov Decision Process (MDP) agent-environment interaction. (inspired from [77]).

Using future cumulative return J_π the RL agent attempts to estimate the optimal policy. The optimal policy

$$\pi^* = \underset{\pi}{argmax} J_\pi, \tag{2.5}$$

defines the best-known action a_t based on the current state s_t where

$$J_\pi = \sum_{t=0}^{H} R_t. \tag{2.6}$$

The R_t indicates the reward at time t and H is the MDP horizon. Depending on the task's temporal behaviour such as repetition or continuity, the cumulative expected reward is associated with different time horizons [25]. In rhythmic tasks where an agent has to repeat a task over and over such as walking or balancing [63, 31] the horizon is infinite which changes the future cumulative reward [11]

$$J_\pi = \underset{\pi}{argmax} \sum_{0}^{\infty} \gamma^t r_t. \tag{2.7}$$

The γ is the discount factor to relinquish the effect of distant future rewards. On the other hand, in the non repetitive tasks (stroke based tasks) [30, 47] the horizon is finite

$$J_\pi = \underset{\pi}{argmax} \sum_{0}^{T} r_t. \tag{2.8}$$

2.2.1 Value-based reinforcement learning

In value-based RL, instead of estimating the optimal policy, the value of each action is estimated via function approximation. The value of action indicates the potential reward of that action given any state s at time t,

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma V'(s_{t+1}|s_t, a_t). \tag{2.9}$$

Therefore, final optimal policy π^* is defined using the value function Q,

$$\pi^*(s_t) = \underset{a_t}{argmax} Q(s_t, a_t). \tag{2.10}$$

Furthermore, The generalized policy function can be derived from,

$$\pi(s,a) = \frac{e^{-Q(s,a)}}{\sum_a e^{-Q(s,a)}}. \tag{2.11}$$

The value function based methods normally cover the whole state-space, therefore, scaling up to high-dimensional state/action-spaces is problematic. It also suffers from error propagation due to sudden large changes in the policy which is not convenient for real robotic systems [47]. The value function approaches usually perform on discrete action-space, and to extend their functionality to support continuous action-spaces a separate policy function should be integrated.
In literature, the policy and value functions are called actor and critic, correspondingly. Therefore, methods that utilize the value and policy function at the same time are called actor-critic methods. For example, Deep Deterministic Policy Gradient (DDPG) is an actor-critic method that operates in continuous action and state-space [38]. In DDPG, the critic function is trained by the returns from the environment and the actor function is improved based on sampling and evaluating the value function. But, deep reinforcement learning suffers from the curse of dimensionality [8], meaning by increasing the dimensionality the volume of task-space increases exponentially. Therefore, there need to be extremely more explorations to cover the whole state-space.

2.2.2 Policy Search Methods

The policy search methods use different ways to encode trajectories into parametric policies $a \sim \pi(a \mid s;\theta))$ with θ as the parameter vector. Then, local exploration in parameter space $\theta_i \sim N(\theta \mid \mu_\theta, \Sigma_\theta)$ is used to find locally optimal solutions like $\theta_{new} = \theta_{old} + \alpha\frac{dJ_\theta}{d\theta}$. The policy search approaches are mostly based on three main concepts:

1. **Policy Gradient (PG)**: This category of approaches use the gradient descent to find the steepest path to the optimal policy. policy gradient algorithms explore the action-space and update the policy based on the returns from executed actions. The eNAC algorithm described in 2.2.7 utilizes the same concept to find an optimal path.

2. **Expectation Maximization (EM)**: This approach tries to iteratively find the maximum likelihood of parameter θ by maximizing the lower bound defined as,

$$\theta_{t+1} = \underset{\theta'}{argmax} L_\theta(\theta') \tag{2.12}$$

   To maximize the function the gradient of the function in respect to θ can be used

$$\partial_{\theta'} L_\theta(\theta') = E\left\{\sum_{t=1}^{T} \partial_{\theta'} \log\pi(a_t \mid s_t, t) Q^\pi(s,a,t)\right\}, \tag{2.13}$$

where Q is the value function defined as,

$$Q^{\pi}(s,a,t) = E\left\{\sum_{z=t}^{T} r(s_z, a_z, s_{z+1}, z) \mid s_z = s, a_z = a\right\}. \tag{2.14}$$

To find the maximum, the gradient with respect to θ should be equal to zero as

$$E\left\{\sum_{t=1}^{T} \partial_{\theta'} \log\pi(a_t \mid s_t, t) Q^{\pi}(s,a,t)\right\} = 0. \tag{2.15}$$

Therefore, an stochastic policy needs to be defined as

$$a = \theta^T \phi(s,t) + \epsilon(\phi(s,t)), \tag{2.16}$$

where ϕ and θ are basis functions and parameters correspondingly. The plain concept leads to an algorithm so called eWPR which is the base of PoWER algorithm described in 2.2.8.

3. **Stochastic Optimal Control (SOC)**: Unlike PG and EM, the SOC based methods attempt to simply minimize a cost function. A control system in SOC is defined as

$$\dot{x}_t = f(x_t) + G(x_t)(u_t + \epsilon_t) = f_t + G(u_t + \epsilon_t), \tag{2.17}$$

where x_t is the state, f(x_t) defines the passive dynamics, G(x_t) is the control matrix, and u_t is the control vector. ϵ_t is the noise which is sampled from a zero mean Gaussian distribution with a covariance Σ.

The cost of a trajectory is defined as

$$R(\tau_i) = \phi_{t_N} + \int_{t_i}^{t_N} r_t d_t. \tag{2.18}$$

where the ϕ_{t_N} defines the final cost and r_t defines the immediate reward at time t. The value function based on the cost function R is defined as

$$V(x_{t_i}) = \underset{u_{t_i:t_N}}{min} E_{\tau_i}[R(\tau_i)], \tag{2.19}$$

where u_t is the optimal control at time t. The value function V can be minimized using the solution of Hamilton-Jacobi-Bellman (HJB) equation as

$$\partial_t V_t = q_t + (\nabla_x V_t)^T f_t - \frac{1}{2}(\nabla_x V_t)^T G_t R^{-1} G_t^T (\nabla_x V_t) + \frac{1}{2} trace((\nabla_{xx} V_t) G_t \Sigma_\epsilon G_t^T), \tag{2.20}$$

and,

$$u_{t_i} = R^{-1} G_{t_i}^T (\nabla_{x_{t_i}} V_{t_i}). \tag{2.21}$$

The PI^2 algorithm described in 2.2.9 uses the nonlinear second order partial differential equation 2.20 to Iteratively improve the policy.

In this chapter, three algorithms are introduced, episodic Natural Actor Critic (eNAC) based on PG [57], Policy Learning by Weighting Exploration with the Returns (PoWER) based on EM, and Policy Improvement with Path Integrals (PI^2) based on SOC. All the approaches that are introduced require encoded trajectory in a form of policy parameters. To do so, deep neural networks [31] or movement primitives [78] can be used to extract the policy parameters and decrease the dimension of the state-space and compress the prior knowledge intuitively.

2.2.3 Dynamic Movement Primitives

Representing a trajectory using its kinematics building blocks, such as position y_t at time step t and its first and second time derivatives, velocity $\vec{v} = \frac{d\vec{y}_t}{dt}$ and acceleration $\vec{a} = \frac{d^2\vec{y}_t}{dt^2}$ is trivial. Therefore a recorded trajectory can be deployed by a robot to reproduce the same movements. Although the trajectory reconstruction would be almost lossless and produces a precise motion, it is not flexible to changes such as introducing a new target position or new obstacles. Therefore a flexible representation that dynamically changes the shape of the trajectory, based on new conditions is required. Using dynamic movement primitives (DMP) [64], any trajectory can be encoded as a dynamical system that can be reconstructed under totally different conditions to reproduce trajectories with different shapes. A DMP is a nonlinear dynamical system that represents a trajectory in one dimension,

$$\frac{1}{\tau}\ddot{y} = \underline{\alpha(\beta(g-y)-\dot{y})} + f \tag{2.22}$$

$$f = \psi_t^T \theta, \tag{2.23}$$

where $[y, \dot{y}, \ddot{y}]$ are position, velocity and acceleration at time t, and the term τ is the non-zero positive time constant to accelerate ($\tau > 1$) or decelerate ($0 < \tau < 1$) the motion while keeping the trajectory consistent. The equation 2.22 is actually combination of linear (underlined part) and nonlinear systems. The linear part is a simple spring damper system, and, the nonlinear part so-called forcing function defines the shape of the trajectory using a set of kernels (e.g. Gaussian functions) ψ_t^T so-called basis functions and shape parameters θ. The jth basis function is defined as,

$$[\psi_t]_j = \frac{w_j(s_t)}{\sum_{K=1}^{p} w_k(s_t)} s_t(g - x_0) \tag{2.24}$$

$$\frac{1}{\tau}\dot{s}_t = -\alpha s_t, \tag{2.25}$$

where s is a linear dynamical system representing the time also called canonical system. The canonical system guarantees s_t reaches zero while x_t converges to the goal g, therefore removing the effect of the forcing function 2.24 when the target is reached. Furthermore, the spatial scaling $(g - x_0)$ is responsible to adapt the

length of the trajectory when a new goal is introduced. In this study, each basis function w is defined to be Gaussian function [64],

$$w_j = e^{-\frac{h_j(s_t - c_j)^2}{2}}, \tag{2.26}$$

where h_j and c_j are the variance and mean correspondingly. As equation 2.26 defines $(s_t - c_j)$, the canonical system s_t has been used to determine the kernel centers (Gaussian mean) making sure the kernels are distributed equally throughout s. To guarantee that the basis functions spread over the state-space equally, the variance of Gaussian kernels also should be optimized. To spread the ψ activations through s the variance is defined as,

$$h_i = \frac{N_{BF}}{c_i}. \tag{2.27}$$

2.2.4 Learning the DMP

When the trajectory is demonstrated, the next step is to learn the trajectory by determining the weights θ in a way that the reconstructed trajectory imitates the demonstrated trajectory as close as possible. The forcing function is the core of the learning, where the basis functions guarantee smooth movements meanwhile the weights promise the flexibility in the Spatio-temporal shape of the trajectory. Due to available training data (demonstration) and linearity of the parameters, the initial parameters are generated using supervised learning. To encapsulate the trajectory in a form of forces the forcing function 2.22 is rearranged as,

$$f = \tau\ddot{y} - \alpha_z(\beta_z(g - y) - \tau\dot{y}). \tag{2.28}$$

The demonstrated trajectory $[y_{demo}(t), \dot{y}_{demo}(t), \ddot{y}_{demo}(t)]$ describes a discrete movement where the samples are taken at time step $t \in [0, 1, ..., T]$. The initial point of the trajectory is defined by $y_{demo}(0)$ and the goal is defined as $y_{demo}(T)$. The final forces f_{target} are calculated using the equation 2.28,

$$f_{\text{target}} = \tau^2\ddot{y}_{demo} - \alpha_z(\beta_z(g - y_{demo}) - \tau\dot{y}_{demo}), \tag{2.29}$$

where the effects of the point attractor system is eliminated. Finally, the supervised learning uses a cost function to determine the numerical distance between demonstrated and reconstructed trajectory to minimize the discrepancies between f and f_{target}. In this thesis, the locally weighted regression (LWR) [65] is used to minimize locally weighted quadratic error J by defining the control parameter ψ,

$$J_i = \sum_{t=1}^{P} \psi_i(t)(f_{target}(t) - w_i\xi(t))^2, \tag{2.30}$$

where $\xi(t) = x(t)(g - y(0))$. The weighted linear regression problem has a solution that allows one shot learning as,

$$w_i = \frac{s^T\Gamma_i f_{target}}{s^T\Gamma_i s}, \tag{2.31}$$

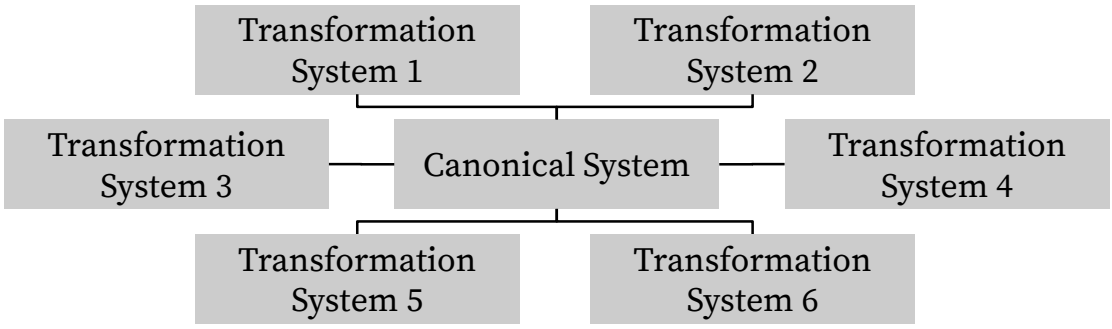


Figure 2.8: The figure shows multiple transformation systems sharing a canonical system as a central clock to synchronise multiple DMPs.

where,

$$s = \begin{pmatrix} \xi(1) \\ \xi(2) \\ ... \\ \xi(3) \end{pmatrix} \Gamma_i = \begin{pmatrix} \psi_i(1) & & & \\ & \psi_i(2) & & \\ & & ... & \\ & & & \psi_i(P) \end{pmatrix} f_{target} = \begin{pmatrix} f_{target}(1) \\ f_{target}(2) \\ ... \\ f_{target}(P). \end{pmatrix} \quad (2.32)$$

One shot learning can increase the performance where there is no need for incremental learning.

2.2.5 Extending DMPs to multiple degrees of freedom

A single DMP can reproduce a trajectory in one degree of freedom (DOF). Nowadays the industrial robots have redundancy in their kinematic chain which means they have more than six DOF. Therefore, multiple DMPs are required to cover the complete trajectory. Every DMP executes the trajectory in one dimension where each dimension must be temporally coupled with the others to prevent inconsistency in the execution. There exist three methods to extend the multiple DMPs [41]:

1. Each DMP uses its transformation and canonical system. But, due to lack of synchronization between DMPs, they diverge through time and cause inconsistency.

2. Each DMP uses its transformation and canonical system and a coupling term to synchronize them. Although coupling terms can extend the capabilities of DMPs they increase the complexity.

3. Each DMP uses its transformation but the canonical system is shared. Using a shared canonical system as a central clock can guarantee stable coordination between all DMPs.

2.2.6 Model-Free Policy search

Model-free policy search approaches directly update the policy using the return value. The general function approximation loop in model-free policy search algorithm is defined as below:

1. Exploration: perturbing the current policy π to generate a set of random trajectories τ_j
2. Evaluation: using a reward function to asses the quality of the perturbed trajectory
3. Update: Using the evaluation results to incorporate different features of different trajectories

This loop demands lots of data to converge to a global maximum but in compare to model-based methods, it is more stable and steady. Also, it requires designing a task-dependent customized reward function which is usually a complicated task. The policy gradient or natural gradient [58], expectation-maximization [28] and path integral [78] methods that have been mentioned are categorised under this category.
The main difference in most model-free policy search methods lies in how they explore the state-space which identifies their control policy. Some algorithms define the exploration as parameter space perturbation and they follow a deterministic control policy. They are called episode-based due to the modification in the parameters before the execution of the trajectory. The step-based approaches, employ a stochastic control policy and add the perturbation to the action-space at each time step.

2.2.7 Episodic Natural Actor-Critic (eNAC)

Policy gradient methods utilize the gradient of the cost function to find the steepest direction in cost-space to minimize the future return. This process requires exploration and exploitation in action-space and consequently calculating the corresponding parameter,

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J_{\theta_k}, \tag{2.33}$$

where θ is the parameter vector and $\nabla_\theta J_{\theta_k}$ is the gradient of cost function with respect to changes in parameter vector,

$$\nabla_\theta J_{\theta_k} = \frac{\partial J_\theta}{\partial \theta}. \tag{2.34}$$

The gradient determines the changes of the cost J due to changes in parameters θ. Therefore, using the gradient matrix the agent finds the steepest path toward the local/global minimum/maximum of the cost function. The general form of a cost function can be defined as,

$$J_\theta = \int p(\tau;\theta) R(\tau) d\tau, \tag{2.35}$$

where, $p(\tau;\theta)$ is expectation over the trajectory distribution,

$$p(\tau;\theta) = p(s_1)\prod_{t=1}^{T}\pi(a_t \mid s_t;\theta)p(s_{t+1} \mid s_t, a_t), \tag{2.36}$$

and $R(\tau)$ determines the reward for each trajectory

$$R(\tau) = \sum_{t=1}^{T} r_t. \tag{2.37}$$

Using the log-ratio transformation the equation 2.35 can be modified as,

$$\nabla_\theta J_\theta = \sum_{i=1}^{N} \nabla_\theta \log p(\tau_i;\theta)R(\tau_i). \tag{2.38}$$

By combing 2.38 and 2.36 the REINFORCE policy gradient [84] is derived,

$$\nabla_\theta J_\theta = \sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \log \pi(a_t^i \mid s_t^i;\theta)R(\tau). \tag{2.39}$$

The REINFORCE algorithm suffers from the high unintended divergence between a new and old policy which eNAC has solved it by using the natural gradient instead of policy gradient. To obtain the eNAC algorithm first the natural gradient should be defined. The natural gradient uses the Fisher information matrix (FIM) as the metric. FIM approximates the Kullback Leibler (KL) divergence of two distribution that defines the distance between them; for example $KL(p_\theta + \Delta_\theta \parallel p_\theta)$ shows the deviation in distribution by adding $\Delta\theta$ to p_θ. It is called an information matrix due to contained information about the effect of each parameter on the properties of the distribution,

$$G(\theta) = \mathbb{E}_p[\nabla_\theta \log p_\theta(x)\nabla_\theta \log p_\theta(x)^T]. \tag{2.40}$$

The natural gradient of the cost function is defined as below:

$$\nabla_\theta^{NG} J_\theta = argmax_{\Delta\theta}\Delta\theta^T\nabla_\theta J \approx G(\theta)^{-1}\nabla_\theta J \tag{2.41}$$

By replacing the policy gradient with a natural gradient in 2.33 the eNAC update equation is achieved,

$$\theta_{k+1} = \theta_k + \alpha\nabla_\theta^{NG} J_{\theta_k}. \tag{2.42}$$

The equations 2.39 and 2.37 show the gradient is defined by the reward of the whole trajectory. since adding the previous rewards into the calculations make the system unstable the effect of past rewards should be removed. Therefore, to generate new policies using future rewards a baseline is introduced to equation 2.39,

$$\nabla_\theta J_\theta = \sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \log \pi(a_t^i \mid s_t^i;\theta)(Q^i - b(s^i)), \tag{2.43}$$

where,

$$Q = \sum_{n=t}^{T-1} r_n + r_T, \tag{2.44}$$

and,

$$Q_t(s_t, a_t) = r(s_t, a_t) + V_{t+y}^{\pi}(s_t + 1). \tag{2.45}$$

Assuming $f_w(s_t, a_t) \approx (Q^i - b(s^i))$, Q can be determined by approximating the function $f_w(s, a)$,

$$V_t(s_t) + f_w(s_t, a_t) = r(s_t, a_t) + V_{t+y}^{\pi}(s_t + 1), \tag{2.46}$$

where the function approximation is defined as,

$$f_w(s_t, a_t) = \psi(s_t, a_t)^T w. \tag{2.47}$$

Based on compatible function approximation which assumes,

$$\psi(s, a) = \nabla_\theta \log \pi(a \mid s), \tag{2.48}$$

and by inserting $\psi(s, a)$ into equation 2.46 the following equation is obtained,

$$\underbrace{V^{\pi}(s_1)}_{J} + \underbrace{\left(\sum_{t=1}^{T-1} \nabla_\theta \log \pi(a_t \mid s_t; \theta) \right)}_{\psi^T} w = \sum_{t=1}^{T} r(s_t, a_t). \tag{2.49}$$

To find proper weights w the linear regression is used in two steps,

- Critic step:

$$\begin{bmatrix} w \\ J \end{bmatrix} = (\Psi^T \Psi)^{-1} \Psi^T R$$

  where

$$\Psi = \begin{bmatrix} \psi_1 & \psi_2 & ... & \psi_N \\ 1 & 1 & ... & 1 \end{bmatrix}^T \quad R = \begin{bmatrix} R_1 & R_2^T & ... & R_N^T \end{bmatrix}^T$$

- Actor step:

$$\theta_{t+1} = \theta_t + \alpha_t w_t$$

2.2.8 Policy Learning by Weighting Exploration with the Returns (PoWER)

PoWER is a robust policy perturbation algorithm based on expectation maximization [61]. The algorithm uses an exploration method. Unlike gradient-based algorithms, perturbation happens in the parameter space. The exploration in the parameter-space leads to smoother trajectories but unobservable actions while perturbation in action-space may lead to

1. large variance in parameter-space that accumulates over time [57, 61, 28],
2. neutralized consequent actions due to independent perturbations and causing the system to act as a low-pass filter [28],
3. damaging the hardware due to instantaneous changes in the movement direction without considering the dynamics of the system [28].

Due to this exploration technique, this algorithm is well-suited for DMPs where the shape of the trajectory is encoded as a set of parameters. Furthermore, the returns are considered as an improper probability distribution where the costs are always positive. The cost for trajectory k at step i,

$$S_{t_i}^k = \sum_{j=i}^{N} r_j^k, \tag{2.50}$$

where r is the return value. The parameter update is calculated as below,

$$\delta\theta \approx \frac{\left(\sum_{k=1}^{K}\sum_{i=1}^{N} M_{t_i}\epsilon_{t_i}^k S_{t_i}^k\right)}{\left(\sum_{k=1}^{K}\sum_{i=1}^{N} M_{t_i} S_{t_i}^k\right)}, \tag{2.51}$$

where M is the projection matrix and, ϵ determines the perturbation.

$$M_{t_i} = \frac{\psi_{t_i}\psi_{t_i}^T}{\psi_{t_i}^T\Sigma\psi_{t_i}} \tag{2.52}$$

The parameter update is defined as,

$$\theta = \theta + \delta\theta \tag{2.53}$$

2.2.9 Policy Improvement using Path Integrals(PI^2)

PI^2 is a local policy improvement algorithm that instead of using expectation maximization it uses the stochastic optimal control concept. The main goal of PI^2 is based on minimizing a cost function J by the locally perturbing policy. The policy perturbation $\theta + \epsilon$ is achieved using a normal distribution $N(0, \Sigma)$.

The parameter Σ standard deviation of the normal distribution is the only hyper-parameter in this algorithm. In order to explain the PI^2 algorithm in detail first the equation 2.22 modified,

$$\frac{1}{\tau}\ddot{y} = \alpha(\beta(g-y) - \dot{y}) + \psi_t^T(\theta + \epsilon). \tag{2.54}$$

By creating N different roll-outs of the perturbed trajectory τ_j and evaluating them using a cost function the trajectories that are close enough to the demonstrated trajectory but have less cost can be found. the cost functions as,

$$J(\tau_i) = \phi_{t_N} + \int_{t_i}^{t_N} (r_t + \frac{1}{2}\theta_t^T R\theta_t)dt \tag{2.55}$$

Where ϕ_{t_N} is the final cost, r_t is an immediate cost and, $\frac{1}{2}\theta_t^T R\theta_t$ is the immediate control cost. The discrete form of the finite-horizon cost function is calculated for each roll-out and at each time step as,

$$S(\{\tau_i\}_k) = \{\phi_{t_N}\}_k + \sum_{j=i}^{N-1} \left\{ r_{t_j} + \frac{1}{2}\Theta_{t_j}^T R\Theta_{t_j} \right\}_k \tag{2.56}$$

where $S(\{\tau_i\}_k)$ is the cost for trajectory k at step i and Θ is the perturbed policy parameters as,

$$\Theta_{t_j} = \theta + M_{t_j}\epsilon_{t_j} \tag{2.57}$$

Here M is the projection matrix to project the Gaussian noise into the parameter space defined as,

$$M_{t_j} = \frac{R^{-1}g_{tj}g_{tj}^T}{g_{tj}^T R^{-1} g_{tj}} \tag{2.58}$$

where g is the matrix of the basis functions. Finally the exponentiated probability for each roll out is calculated. This probability defines the importance of perturbation noises ϵ in that particular trajectory. The probability is calculated as,

$$P(\{\tau_i\}_k) = \frac{e^{\frac{-1}{\lambda}S(\{\tau_i\}_k)}}{\sum_{l=1}^{K}\left[e^{\frac{-1}{\lambda}S(\{\tau_i\}_l)}\right]}. \tag{2.59}$$

The parameter λ regulates the sensitivity of the exponentiated cost. the λ can be remoed from the equation by,

$$e^{-\frac{1}{\lambda}S(\tau_i)} = e^{\frac{-h(S(\tau_i) - minS(\tau_i))}{maxS(\tau_i) - minS(\tau_i)}}, \tag{2.60}$$

where h is a constant in which is defined 10 as suggested in [78]. Using the probability weighted average over K roll-outs, the parameter update $\delta\theta_{t_i}$ for each time step i can be computed as,

$$\delta\theta_{t_i} = \sum_{k=1}^{K} \left[P(\{\tau_i\}_k) M_{t_i} \{\epsilon_{t_i}\}_k\right], \tag{2.61}$$

where trajectories with lower cost and higher probability, contribute more to the parameter update [76]. Similarly, the averaging is calculated over time as,

$$[\delta\theta]_j = \frac{\sum_{i=0}^{N-1}(N-i)w_{j,t_i}\left[\theta_{t_i}\right]_j)}{\sum_{i=0}^{N-1}(N-i)w_{j,t_i}}. \tag{2.62}$$

In the time average the activation of the basis functions are used to increase the effect of the parameter updates with higher activation. The coefficient $(N-i)$ is to weight each parameter update at time i based on the number of steps left on the trajectory, this creates a higher impact from the points closer to the start of the trajectory. To obtain the new parameter the parameter updates is added to the old one as,

$$\theta_{\text{New}} = \theta_{\text{Old}} + \delta\theta. \tag{2.63}$$

Chapter 3

Methodology

3.1 Modular Architecture

In this section the architecture of the teleoperation system is presented. The final architecture is designed in a modular way to facilitate the analysis and testing process. The figure 3.1 illustrates the overall architecture of the MMT system. The network so-called delay in the figure splits the architecture into two main parts the slave and master sides. Each module (blocks in the figure) is assigned to execute a separate task that is defined as below:

- Master side:
  - **Human**: The teleoperator who demonstrates the trajectory using the master device
  - **Master**: The haptic user interface which perceives the operator's hand movements and gives the haptic feedback to the operator.
  - **IK**: The inverse kinematics block that calculates the human hand posture using haptic interface contact points in the work-space.
  - **HPE**: Hand Pose Estimation calculated slave hand configuration based on human hand posture
  - **Hand/LWR Simulink simulation**: Single instance of Light Weight Robot (LWR) and Five Finger Hand (FFH) simulations. This simulator using the environment generates haptic feedback for the Master device.
  - **Environment**: Singe instance of physics simulation combining environment with the 3D model of LWR and FFH. This simulation does not simulate the kinematics and dynamics of LWR and FFH, and it just simulates the interaction forces with the environment.
  - **DMP**: Dynamic Movement Primitive transformation in the master side to encode the trajectory to a set of policy parameters.
  - **AR**: Augmented reality module which provides a stereo visual representation as a separate environment simulation without considering the physical properties.

- Slave side:
  - **DMP**: Dynamic Movement Primitive transformation on the slave side to decode the policy parameters to the final trajectory.
  - **Reinforcement Learning**: Improves the trajectory in the case when DMP fails to complete a task.
  - **Hand/LWR simulation**: Multiple instances of LWR robot and FFH slave hand simulator. This simulator continuously interacts with the DMP and Reinforcement Learning blocks for trajectory evaluation.
  - **Environment (blue blocks)**: multiple instances of physics simulation of the remote environments and LWR and FFH. Each instance interacts with the one LWR and FFH Simulink simulator to update the kinematics and dynamics of the physics model. Moreover, this simulation gets updated frequently to increase the feasibility of the final trajectory.
  - **Slave** The real LWR and FFH.
  - **Environment**: The real remote environment.

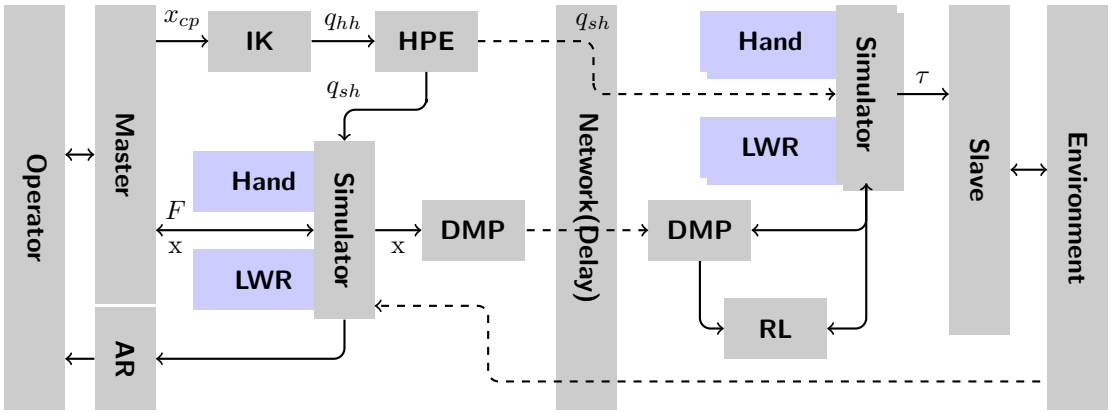


Figure 3.1: The figure illustrates the overall architecture of the system. The network also called the delay in the figure splits the architecture into two main parts the slave and master side. The *Human* teleoperator interacts with the *Master* device. In the Master side *LWR/Hand simulation* using physics simulation, *Environment* provides instant haptic feedback and also information for the Augmented Reality unit *AR*. The Inverse Kinematic *IK* block calculates the human hand posture given the contact points on the master device. Consequently, the slave Hand Posture Estimator *HPE* approximates the corresponding slave hand configuration. The *DMP* in the master side encodes the trajectory to basis functions while the *DMP* in the slave side encodes them back, and evaluates and reconstructs the new trajectory based on the new environmental situation given by *Hand/LWR* and *Environment* simulator. IN the case of failure the *Reinforcement Learning* unit improves the trajectory and finally, deploys it on the *Slave* robot in the remote *Environment*

3.2 Trajectory Encoding

Transferring knowledge between structurally different systems requires transformation and usually results in information loss. Adding more transformations leads to high degrees of adaptation and eventually higher performance and lower loss. Similarly in teleoperation, in most cases, slave and master have fundamental differences that aggravate a lossless performance. Besides knowledge transfer, communication causes a higher loss of information and also lower performance. To evaluate the knowledge transfer and trajectory reconstruction different properties are taken into account as follows:

1. **The flexibility of reconstruction and degree of adaptation to new conditions:** due to the time difference between demonstration (master side) and execution (slave side), the environment may change, the differences might be caused by the slave robot or an external factor (e.g. slipping, drifting). Therefore, this dissimilarity should be taken into consideration for new trajectory generation, the potential differences that are taken into account are:

   - The starting point of the trajectory
   - The target point of the trajectory
     - Reachability of the target
     - Uncertainty in target position
   - Obstacles

2. **Dimensionality of encoded space:** The minimum number of parameters that are required to encode a trajectory. A low number of parameters leads to,

   - less processing overhead
   - less network traffic
   - faster optimization

   To ensure the transformation loss does not affect the system eventually, an optimal trade-off needs to be found as a very high number of parameters leads to complexity and a very low number of parameters leads to inconsistency and excessive loss.

3. **Safe exploration and learning:** The reconstructed trajectory must be as safe as the demonstrated trajectory by the supervising teleoperator. Since learning mainly relies on exploration in parameter or action-space, intuitive representation of the trajectory helps to keep the exploration safe.

The trajectory can be encoded in the form of neural network weights, Bézier curves and Dynamic movement primitives (DMP), etc.. This thesis mainly focuses on utilizing DMP transformations. Every DMP encodes a trajectory in one DOF

therefore for multiple DOF, multiple DMPS are required. DMPs can learn to reconstruct a trajectory with different temporal and spatial features from the base trajectory. The reconstruction is happening with one-shot learning. Given the position $[x_0, x_1, ..., x_N]$, velocities $[\dot{x_0}, \dot{x_1}, ..., \dot{x_N}]$ and acceleration $[\dot{x_0}, \dot{x_1}, ..., \dot{x_N}]$, the equation 2.31 generates DMP parameters θ. The goal x_N or the start of the trajectory x_N can be modified based on the new situation on the slave side. The dimensionality of space of each trajectory is six, three for position and three for orientation. As a result, six DMPs are used to encode every demonstration.

$$\begin{bmatrix} x & y & z & \alpha & \beta & \gamma \end{bmatrix} \tag{3.1}$$

The roll, pitch, and yaw of the end-effector are then converted to a rotation matrix by a set of transformations to avoid singularities. The hand joints are not considered in the DMP transformation due to the high number of DOFs (21 DOF) which is outside of the scope of the thesis.
The figure 3.1 shows our proposed architecture, as can be seen, the slave robot gets the trajectory from two sources, DMP, and Reinforcement learning units. The DMP in the master side encodes the trajectory using the following steps[64]:

1. Calculate the derivative of position trajectory to access velocity and acceleration
2. Calculate the f_{target} using equation 2.29

$$f_{\text{target}} = \tau^2 \ddot{y}_{demo} - \alpha_z(\beta_z(g - y_{demo}) - \tau \dot{y}_{demo})$$

3. Calculate the basis functions ψ using equation 2.24 which is shown in figure 3.2.

$$[\psi_t]_j = \frac{w_j(s_t)}{\sum_{K=1}^{p} w_k(s_t)} s_t(g - x_0)$$

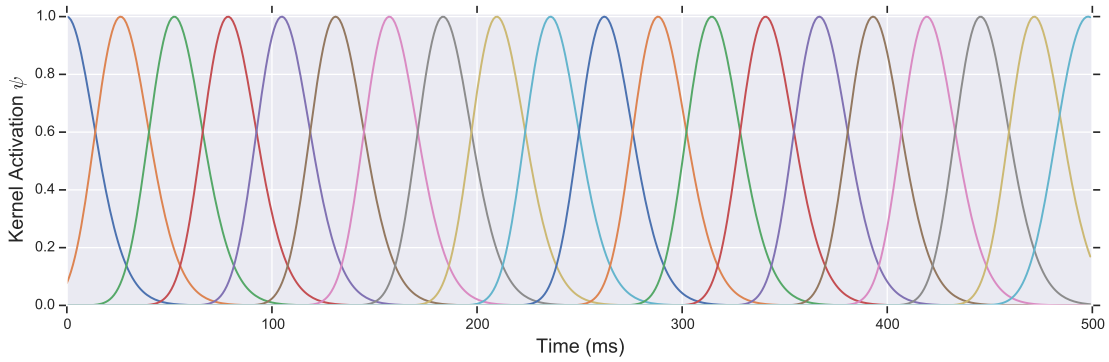


Figure 3.2: The figure shows the activation of each basis function (Gaussian Kernel) at every time step.

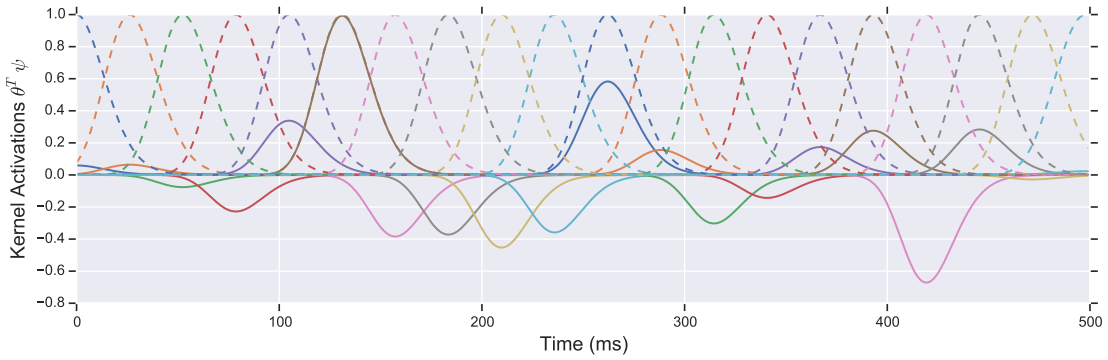


Figure 3.3: The figure shows the activation of kernels after applying the learned parameters.

4. Calculate the parameters θ using linear weighted regression equation 2.31

$$w_i = \frac{s^T \Gamma_i f_{target}}{s^T \Gamma_i s},$$

The DMP block on the slave side decodes and rebuilds the trajectory given the updated values from the slave robot and the environment with the following steps:

1. Calculating the forcing function f using equation 2.23, an example of final results after this step is depicted in figure 3.3

$$f = \psi_t^T \theta,$$

2. Reconstruct the trajectory using equation 2.22. This step is incremental and generates the trajectory in a loop with the length of the trajectory, the initial value of the trajectory y_0, is arbitrary and can be different from the demonstration (e.g. the current position of the end effector). the same procedure for the goal g can be adapted to the new position of the target. An example of a reconstructed trajectory in comparison with the main trajectory is depicted in figure 3.4

$$\frac{1}{\tau}\ddot{y} = \underline{\alpha(\beta(g - y) - \dot{y})} + f$$

The reconstructed trajectory gets evaluated using simulation, to see if the trajectory can fulfill the task requirements. This thesis mainly focuses on grasping different objects. The objects have been selected to evaluate the system's performance in grasping objects with different types of geometry (e.g. centric, parallel, arbitrary). The evaluation conditions are defined in the simulation as described in 4.3. If the reconstructed trajectory can grasp the object in the simulation environment, then the real slave robot deploys the trajectory, otherwise, the DMP unit activates the reinforcement learning to adapt and improve the trajectory until a successful grasp is achieved.

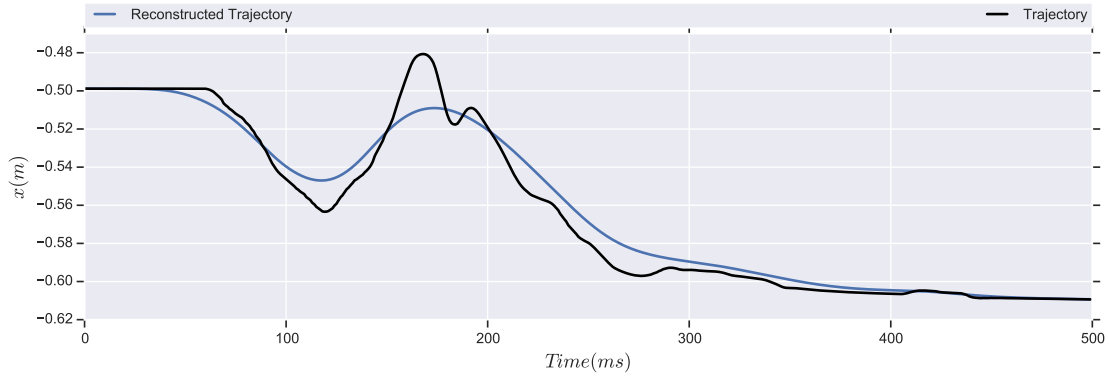


Figure 3.4: The figure shows a reconstructed trajectory generated by a DMP. The black trajectory shows the demonstration by the operator. Due to the low number of kernels, the reconstructed trajectory using DMP has a rough estimation of the demonstrated one, therefore losing some features.

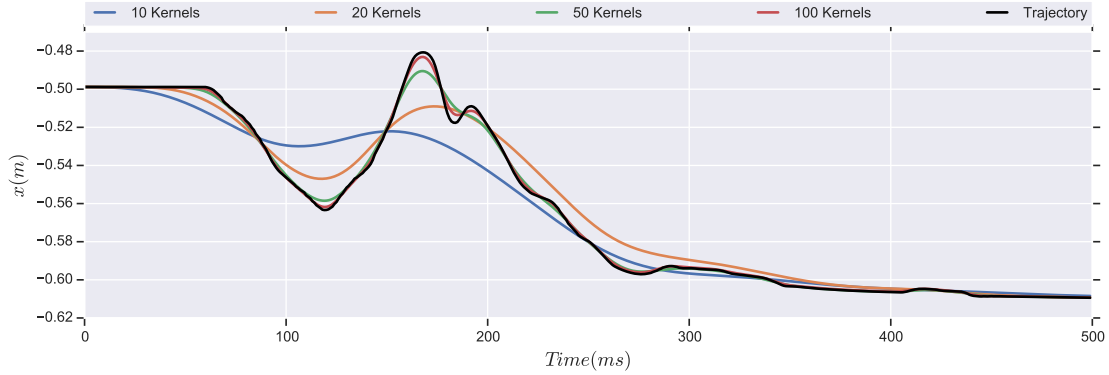


Figure 3.5: The figure shows the trajectories generated using 10, 20, 50, 100 kernels. Increasing the number of kernels improves the accuracy of the trajectory by adding more features throughout the trajectory. on the other hand, the unnecessary high number of kernels increases the learning time and complexity of the trajectory representation.

3.3 Reinforcement learning

Although DMPs can adapt an old trajectory to new conditions, they may fail due to reasons such as the approaching angle for grasping with an arbitrary structured asymmetric end-effector (e.g. anthropomorphic hand) while for a manipulator with symmetric end-effector configuration the approach angle does not make any difference. There are also other factors such as the accuracy of the controller to follow a generated trajectory as close as possible, in this case, failure happens when the controller diverges incrementally from the intended trajectory. The controller accuracy relies on the quality of the hardware, for example, a joint with high inertia or friction cannot follow controller commands due to physical restrictions.
Furthermore, uncertainty in the object position is a common problem and causes

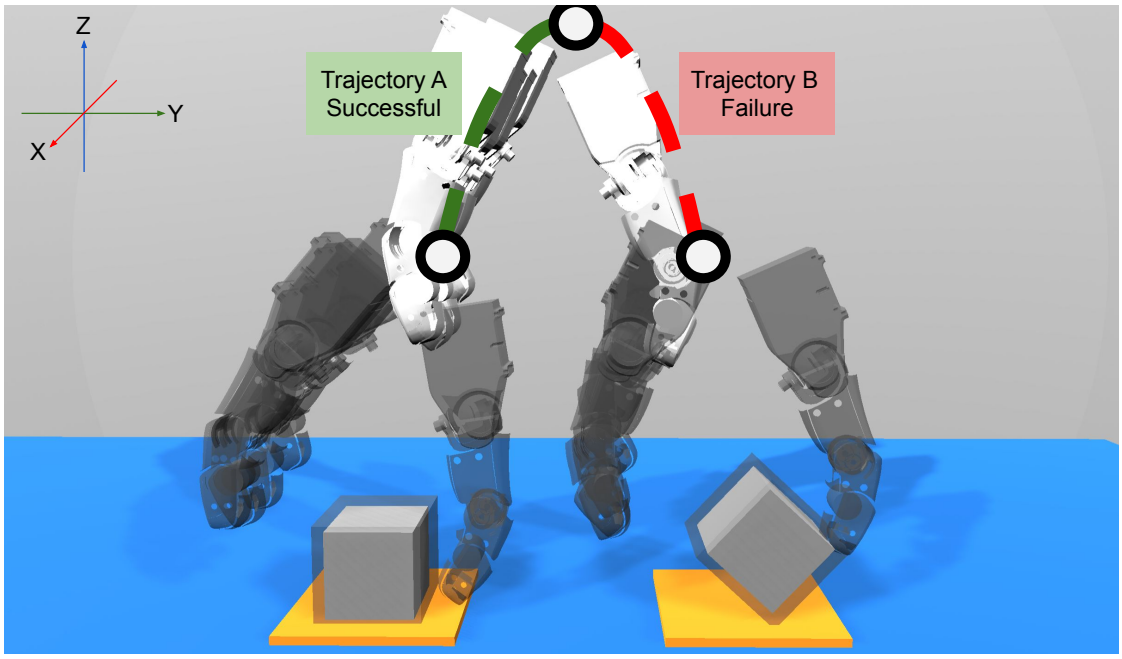


Figure 3.6: The figure shows two trajectories with the same shape but with different target positions relative to the initial positioning. The problem appears to be due to the asymmetric shape of the end effector.

collision and inconsistency in grasping and eventually failure of the task. Hence, a new demonstration might be costly and also result in the same problem, therefore it is necessary to adapt the same trajectory to new conditions. There are several ways to improve a trajectory, such as policy search, path integrals, deep reinforcement learning, etc. In this thesis, the comparison between fast and robust algorithms such as Policy Improvement using Path Integrals(PI^2), Policy Learning by Weighting Exploration with the Returns (PoWER) and Episodic Natural Actor-Critic (eNAC) is presented.

PoWER and PI^2 are policy perturbation methods that add noise to the original trajectory in the parameter space $\pi_{\theta+\epsilon}$ to generate more trajectories that are similar to the original one but have slightly different features. These exploratory trajectories are essential for RL, they visit new parts of the task-space to find an optimal solution if it exists. Unlike deep RL, policy search methods usually have a limited amount of time to find a solution for one specific problem, therefore, the approach does not consider generalization. The figure 3.7 shows a detail view of the RL block mentioned in figure 3.1, which outlines the general structure of policy perturbation methods. Both PI^2 and PoWER algorithms have a similar exploration behavior, but the differences are in the calculation of parameter update $d\theta$. The parameter update in PI^2 is calculated in a way that provides more freedom for designing the cost function. Whereas the cost function in the PoWER algorithm has to be an improper probability distribution, therefore, the returns should always be positive. The use of proper probability distribution may be advantageous, Where

the rewards should be positive and add up to one [28].
The PI^2 parameter update is calculated in five steps,

1. Determining the cost to go S for each roll-out at each time step using the equation 2.56

$$S(\{\tau_i\}_k) = \{\phi_{t_N}\}_k + \sum_{j=i}^{N-1} \left\{ r_{t_j} + \frac{1}{2}\Theta_{t_j}^T R \Theta_{t_j} \right\}_k .$$

2. Calculate the probability p of each roll-out using the exponential form of S using equation 2.59. Higher probability indicates a higher contribution of the roll-out in the final update $\delta\theta$.

$$P(\{\tau_i\}_k) = \frac{e^{\frac{-1}{\lambda} S(\{\tau_i\}_k)}}{\sum_{l=1}^{K} \left[e^{\frac{-1}{\lambda} S(\{\tau_i\}_l)} \right]}$$

3. Averaging over roll-outs using equation **??**. The weighted average is due to the importance of different trajectories, here the trajectories with the lower cost which have a higher probability, have a higher influence on the final parameter update.

$$\delta\theta_{t_i} = \sum_{k=1}^{K} [P(\{\tau_i\}_k) M_{t_i} \{\epsilon_{t_i}\}_k]$$

4. Average over time-steps using equation 2.62. The final $\delta\theta$ at time step i is weighted based on remaining points on the trajectory. It means the points closer to the start of the trajectory has more influence on the shape since they affect the bigger part of trajectory [76].

$$[\delta\theta]_j = \frac{\sum_{i=0}^{N-1} (N-i) w_{j,t_i} [\theta_{t_i}]_j)}{\sum_{i=0}^{N-1} (N-i) w_{j,t_i}}$$

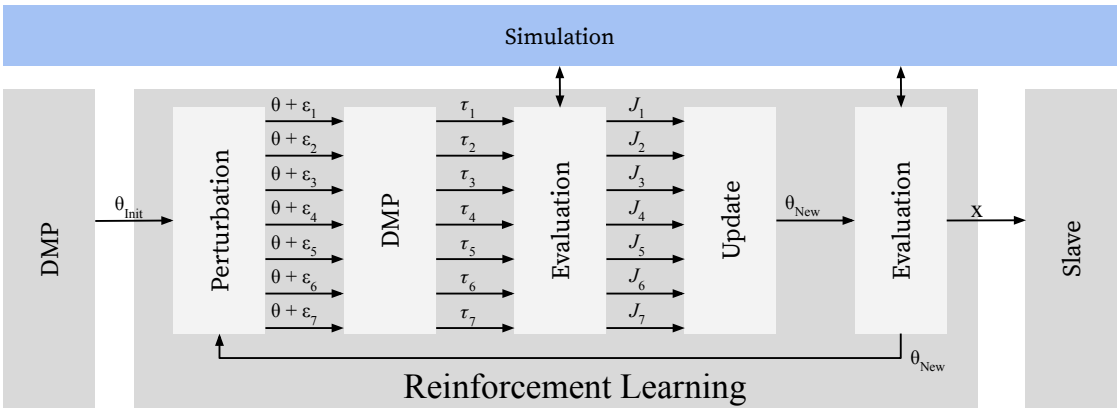


Figure 3.7: The figure shows the general structure of policy parameter perturbation methods inspired from [76].

5. Update the parameters using equation 2.63

$$\theta_{new} = \theta_{old} + \delta\theta$$

But the PoWER algorithm uses a simpler method and update the parameter θ in three steps:

1. Determining the cost to go S for each roll-out at each time step using the equation 2.50
$$s_{t_i}^k = \sum_{j=i}^{N} r_j^k$$
2. Average over time-steps and roll-outs in a single step using the equation 2.51.
$$\delta\theta \approx \frac{\sum_{k=1}^{K}\sum_{i=1}^{N} M_{t_i}\epsilon_{t_i}^k S_{t_i}^k}{\sum_{k=1}^{K}\sum_{i=1}^{N} M_{t_i} S_{t_i}^k}$$
3. Update the parameter using the equation 2.53.
$$\theta_{new} = \theta_{old} + \delta\theta$$

The eNAC algorithm is an actor-critic policy gradient which uses the natural gradient to find the steepest path to the optimal answer. It is called episodic since the critic is evaluated at the end of trajectory execution, but the algorithm is step-based since the actions are perturbed each frame. Perturbing actions has disadvantages [75] such as,

- Introducing independent random small movements to a smooth trajectory increases the jerkiness,
- The system might behave as a low-pass filter and filter-out the high-frequency perturbations by averaging.
- Instant changes in robot motion without considering the inertia might damage the robot,
- The variance of the parameter vector might increase dangerously.

3.3.1 Learning under Uncertainty

Due to the uncertainty in object detection, the RL agent must be able to learn the new trajectory goal while optimizing the trajectory shape. The authors in [76] introduce an approach to extend the capability of PI^2 to adapt the target of the trajectory. The approach uses the cost of the whole trajectory $S(\{\tau_0\}_k)$ since the goal does not change throughout the trajectory execution. Given the trajectory cost S the probability of each trajectory can be calculated,

$$P(\{\tau_0\}_k) = \frac{e^{\frac{-1}{\lambda}S(\{\tau_0\}_k)}}{\sum_{l=1}^{K}\left[e^{\frac{-1}{\lambda}S(\{\tau_0\}_l)}\right]} \tag{3.2}$$

Where P determines the contribution of each trajectory for the next update. To be exact, P defines the value of each trajectory based on the cost function. Then the probabilities are used to calculated the goal parameters,

$$\delta g = \sum_{k=1}^{K} [P(\{\tau_0\}_k) \{\epsilon^g\}_k], \tag{3.3}$$

where $\{\epsilon^g\}_k$ is the goal exploration related to the trajectory k. The final update of the goal is defined as below,

$$g = g + \delta g \tag{3.4}$$

The haptic feedback as secondary sensory feedback is used to compensate for the vision inaccuracy.

Formulation

Exploration Rate Σ

As PI^2 and PoWER approaches suggest, the perturbation step introduces exploration into the parameter space. The exploration may cause serious disruptions in the agent's behavior and unpredictable movement which can adversely affect the slave robot and the remote environment simultaneously. Due to the counter-intuitive definition of the parameters θ which define the final shape of the trajectory by modifying the DMPs, an intuitive understanding of the applied noise (exploration) Σ is required before performing actions on the slave. A high exploration rate in parameter perturbation methods may highly diverge the trajectory and cause physical damages to the robot.
On the other hand, the eNAC algorithm uses the action-space to explore the new policies therefore the exploration rate Σ is an intuitive measure of distance. The disadvantage of the perturbation in the action-space is the jerky movement. Low exploration rate leads to a smoother trajectory while a high exploration rate may cause damage to the robot by big instance changes in the movement direction. In parameter and action-space perturbation methods, a trade-off can be found that speeds up learning while keeping the slave device and remote environment safe.
Furthermore, initially, the RL agent requires higher exploration to increase the chance to find a solution within the time limit. For that reason, when a solution has been found the rate of the exploration must decrease to stabilize the convergence and finalize the trajectory shape. Therefore, the exploration rate Σ decreases at each time step using the following equation:

$$\gamma = max(\frac{\text{Update}_{\text{max}} - i}{\text{Update}_{\text{max}}}, 0.1) \tag{3.5}$$

$$\Sigma_i = \gamma \Sigma_{\text{init}} \tag{3.6}$$

Where i is the current time step in the trajectory and $\text{Update}_{\text{max}}$ is the maximum number of the updates. The decay rate γ is responsible for linearly decreasing

the exploration rate. The decay rate has a minimum value of 0.1 due to need for exploration throughout the learning process. On the other hand, the exploration rate of the goal defines the perturbation of the target position. Therefore high goal exploration leads to failure due to diverging quickly before having a chance to touch the object. On the other hand, a low exploration rate results in late convergence, therefore the trade-off should be defined properly depending on the model uncertainty. The learning exploration rate for each approach was defined as experimentally in the simulation. The final exploration rates for each approach is defined as below:

	PI^2	PoWER	eNAC	Goal
Σ	300	300	0.01	0.04

Learning Rate α

In the critic step of the eNAC algorithm, a learning rate is deployed to control the final influence of parameter updates on the final trajectory. A small learning rate leads to late convergence while a high learning rate results in sudden changes in the trajectory and failure due to high divergence from the initial trajectory.

Cost Function

The cost function in policy search algorithms defines whether the agent should get a reward. It also defines the scale of the reward depending on the quality of the action. A cost function interprets several aspects of action in one single value with different proportions depending on the purpose of the learning. For example, the cost function can be a combination of,

- Acceleration of the end-effector,
- Velocity of the end-effector,
- Distance from the target,
- The scale of the added noise,
- The number of frames that fingertips are inside the Diaphragm (High number of frames indicate the trajectory target is well placed in relative to object),
- Number of fingers involved in the grasp (High number of frames indicate a better grasp),
- The object displacement during grasp.

Different combinations of these parameters have a different interpretation and they result in different trajectories. For example, integrating the acceleration into the cost function results in a less jerky trajectory [76]. Another example, by adding the

object displacement during the grasp makes the end effector not touching during its approach. consequently, considering any mentioned property results in a new trajectory where that property is minimized. Moreover, The simulation involves characteristics that are unrealistic like continuous access to the precise object position. In a real environment, the object can be obscured and hard to detect and locate, therefore, using cost items like the object position is not plausible. As a final solution, after experiments, two properties are selected; acceleration and noise scale for the control and trajectory cost. and for the final cost, the number of involved fingers in grasping is included. The cost function is defined as below [76],

$$J(\tau_i) = \Phi_{t_N} + \int_{t_i}^{t_N} (10^{-11}(\ddot{x_t}^2 + \frac{1}{2}\theta_t^T R \theta_t))dt. \tag{3.7}$$

where Φ is the final reward indicating the success of the grasp. The quality of the grasp is determined by the number of fingers involved in the grasp $N_{Fingers}$

$$\Phi(\tau_i) = 1 - \frac{N_{Fingers}}{5}. \tag{3.8}$$

The Φ becomes zero when all fingers are involved in the grasp but depending on the object size the maximum number of fingers might differ. In the next chapters, the penetration calculation in the physics engine is explained.

Roll-out and Update number

The number of roll-outs determines the number of perturbed instances of the trajectory which must be generated and evaluated for the next update. And the number of updates defines the maximum number of steps in which at each step one set of roll-outs are generated. Therefore the total number of trajectories generated is defined as Roll-out number $\times$ Update number. The higher number of roll-outs k or trajectory perturbation leads to robust learning and slower convergence [76]. 100 updates with Seven roll-outs have been used in the experiments due to hardware limitations. To increase the learning stability the best two trajectories are kept in the memory to be used in the next update. Although, increasing the roll-out number increases the robustness against noise but it results in slower convergence.

Chapter 4

Implementation and Validation

To provide a framework to evaluate the system, different modules are designed and implemented. The concept behind these modules are not the focus of this thesis but they are necessary to evaluate the study. The main focus of this chapter is the link between the hardware and software, controllers, and detail description of the modules in figure 3.1 that were not investigated in the previous chapter. Also, the hardware architecture including manipulators, robotic hands, visualization devices, and the middleware are explained with a brief description of the experiments and the preliminary user-study procedure.

4.1 Master-Slave Command

High-level task demonstration such as grasping in a teleoperation scenario requires a highly compliant interface that provides weightless movement throughout the entire work-space [56]. In addition to compliance, proper haptic feedback helps the user to understand the physical properties of the remote environment by touching the objects and feeling different surfaces. In haptic rendering, compliance plays an important role. Just like a highly noised image, small inertia or discrepancies in haptic feedback may confuse or mislead the operator. Therefore, using a gravity compensation module which provides a weightless physical interface by counterbalancing the weight of the manipulator. Furthermore, other factors such as inertia or friction that may affect the quality of compliant feedback are taken into account. The Lagrangian formulation of an uncontrolled robot is defined as below:

$$F = M(q)\ddot{x_d} + c(q, \dot{q}) + g(q) + h(q, \dot{q}) \tag{4.1}$$

Where M, c, g, f are inertia, coriolis and centrifugal, gravity and nonlinear terms and the parameter x_d defines the target position of the end-effector. The compliance can be extended to each robot finger. To do so, a torque controller has been deployed on the robot hand which provides freedom in the movement at the finger level. Consequently, the in-hand (e.g. gestures) and human arm movements were followed by the robot fingers and the manipulator, correspondingly.

On the slave side, the robot requires a position to follow the master, this position so-called anchor can be any point on the master device, such as the estimated human hand position or the position of the end-effector. The anchor is a virtual link that the slave robot uses to replicate the master movements. Using the master's end-effector as an anchor without considering the end-effector (robot hand) causes inconsistency in the model because the displacement of the anchor depends on the distance from the previous joint. To ensure low position tracking error, the anchor point is defined by estimating the pose between the Carpometacarpal and Inter-Metacarpal(IMC) joints in the human palm as shown in figure 4.3. To do so, the contact points 1 and 2 (red and yellow rings) which are physically connected to latter joints, therefore, the average position of the contact points defines the anchor (blue ring). To follow the anchor point by slave robot, a position tracking controller with high accuracy is necessary. There are two solutions, using inverse kinematics that calculates the joint angles based on a given absolute position in the work-space or an impedance controller which uses a spring-damp system to follow a relative target in work-space. Although inverse kinematics guarantees high accuracy, it imposes high stiffness to the joints. On the other hand, the impedance controller supports variable stiffness in Cartesian space which provides a safe environment for the human-robot interaction.
The impedance controller (figure 4.1) uses the displacement of the anchor ($x_t - x_{t-1}$) at each time step t. Considering the Lagrangian formulation the impedance controller,

$$F = K_x(x_d - x) + K_d(\dot{x_d} - \dot{x}) + M(q)\ddot{x_d} + c(q, \dot{q}) + g(q) + h(q, \dot{q}), \tag{4.2}$$

where K_x and K_d are the stiffness and damping matrices,

$$K_x^P = \begin{bmatrix} 800 & 0.0 & 0.0 \\ 0.0 & 800 & 0.0 \\ 0.0 & 0.0 & 800 \end{bmatrix}, K_x^R = \begin{bmatrix} 9.6 & 0.0 & 0.0 \\ 0.0 & 9.6 & 0.0 \\ 0.0 & 0.0 & 9.6 \end{bmatrix}, K_d = \begin{bmatrix} 9.9 & 1.0 \\ 9.9 & 1.0 \\ 9.9 & 1.0 \end{bmatrix}.$$

To achieve final torque on the motors the Jacobian of the manipulator is used as below [3, 83]:

$$\tau_d = J^T(q)F \tag{4.3}$$

It is worth to mention that as figure 4.2 shows due to the opposing posture of the teleoperator relative to the interface a transformation is required which rotates the haptic interface -90 degrees around the z-axis. The reason that the robot needs -90 degrees rotation instead of -180 degrees, is due to the initial -90 degrees rotation of the end effector (FFH) relative to its base.

4.2 Hand Posture Estimation (HPE)

As figure 4.4 shows, the slave, master and human hands have a different structure. A pipeline of Inverse Kinematic (IK) [56] and joint-to-joint mapping has been used to estimate the human hand posture using the slave hand. The IK is used since

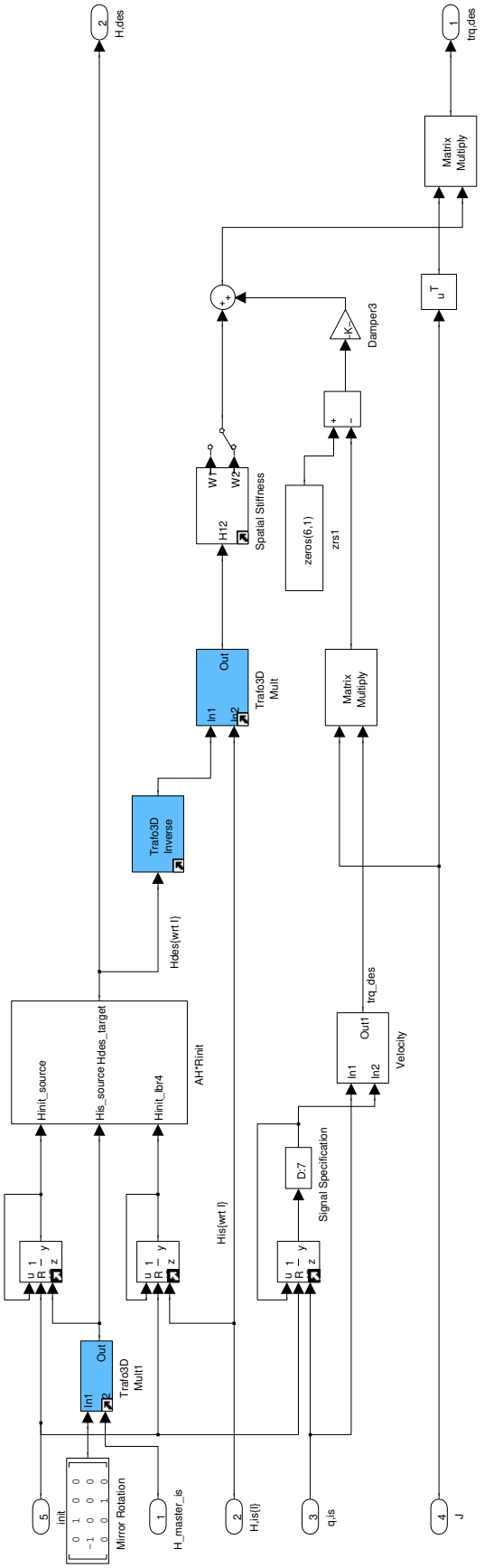


Figure 4.1: The figure shows the Simulink model related to the LWR impedance controller. The *Mirror transformation*, transforms the anchor pose (*H_master_is*) while the deviation calculator ($AH * R_{init}$) generates the slave end-effector pose (*Hdes*). The changes are transformed into spacial stiffness and added to the damping matrices. The damper uses the velocity and the Jacobian of the manipulator. The final torque (trq_{des}) is deployed on the motors.

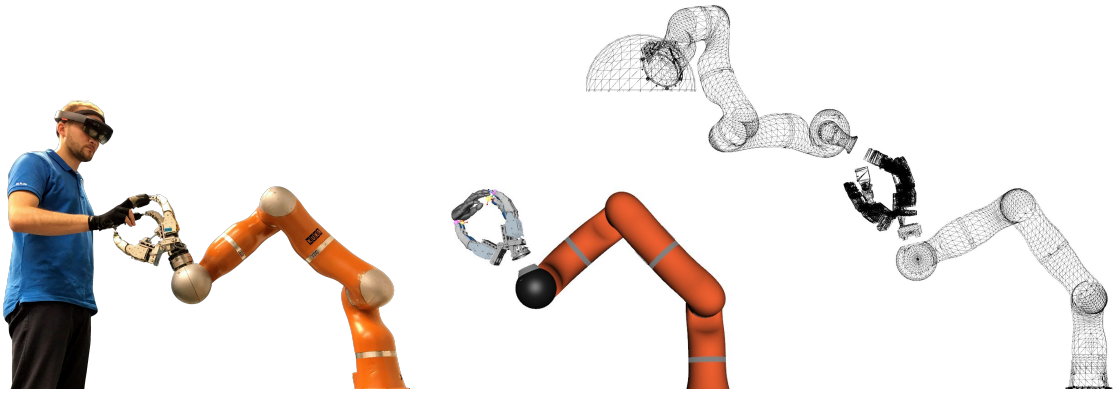


Figure 4.2: The figure shows how the operator hand model is evolved in the two steps to be used as the robot hand. On the left, the operator is attached to the haptic interface, then, the contact points of the hand and interface are calculated using the forward kinematics. The calculated positions are used to estimate the human hand posture using the inverse kinematics of the human hand which is extracted from MRI models. Finally, the estimated human hand configuration is used for a joint to joint approximation to calculate the associated robot joint angles.

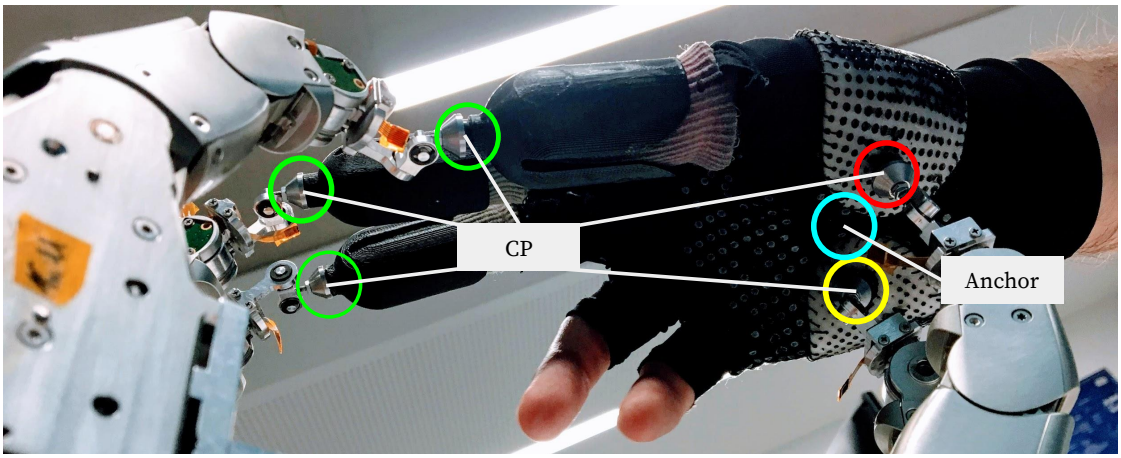


Figure 4.3: The figure shows the operator's hand attached to the Exodex Adam haptic interface. The green rings show the fingertip attaching point while the red and yellow show the palm attach points. The blue ring defines the anchor point that the slave robot uses to follow the operator hand movement.

there is no direct way to read human joint values. The IK utilizes the contact points calculated using forward kinematic (fingertips of master robot hand). Then, the joint-to-joint mapping uses the human hand joint configuration to compute slave robot joint values.

The fmincon is used as the optimization method which finds the minimum of a constrained nonlinear multivariable function. The cost function takes into account the Euclidean distance between an arbitrary number of points on two curves. Each

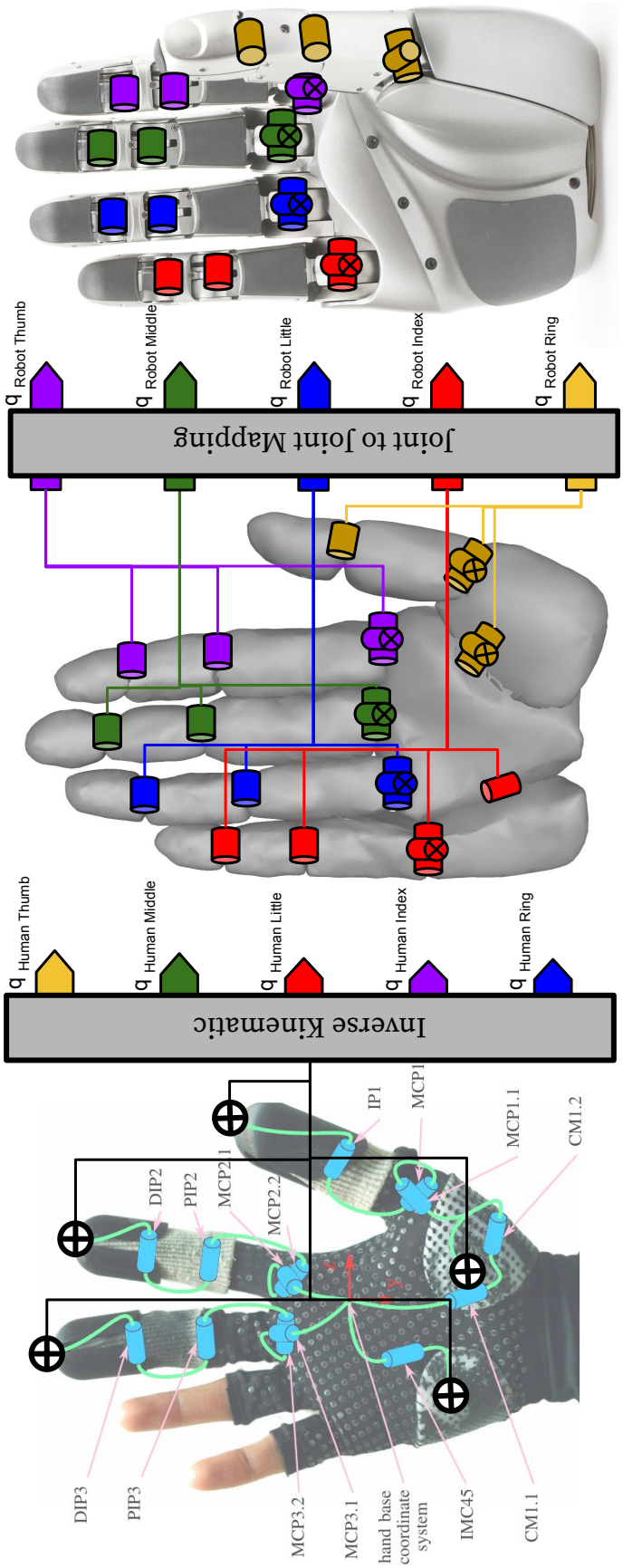


Figure 4.4: The figure shows from left to right first the human hand with the glove and finger caps [56], the MRI generated hand model, and the Five Finger Hand (FFH). The contact points then are used to estimate the human hand MRI configuration using the inverse kinematic. Finally, the estimated human hand configuration is used for a joint to joint mapping to calculate the associated robot joint angles.

curve passes through each joint on both sides (human MRI hand (Master) and robot hand (Slave)). Therefore, the cost function is a similarity measure between two fingers (human MRI model finger and its associated finger on the slave robot). The inputs of the cost function consist of 8-10 constants depending on the finger. To find the distance between two curves first a transformation must be calculated that matches two curves. Finally the evaluation is done by calculating the distances in the finger's range of motion. The configuration of each joint (Slave) has been defined by a set of linear equations. Each joint is assumed to be coupled with 2-3 different joints on the human side. The equations for the index finger are given as follows

$$q1 = \alpha_1 MCP_{2.1} + \beta_1 MCP_{2.2} + \gamma_1 PIP_2$$

$$q2 = \alpha_2 MCP_{2.1} + \beta_2 MCP_{2.2} + \gamma_2 PIP_2$$

$$q3 = \alpha_3 PIP_2 + \beta_3 DIP_2$$

$$q4 = \alpha_3 PIP_2 + \beta_3 DIP_2 = q_3$$

$$q_{index} = [q_1, q_2, q3, q_4]$$

Where MCP, PIP and DIP are Metacarpophalangeal, proximal Interphalangeal and distal Interphalangeal joint values correspondingly. And, q_1, q_2, q_3, q_4 are the slave robot hand joint values and α, β and γ are constants that are being optimized. As you see there are eight different values to be optimized. For example, the first joint of the robot q1 is affected by the first three joints of the human hand MRI Model. This is an example of optimization results

$$q1 = (0.6894 MCP_{2.1}) + (0.0140 MCP_{2.2}) + (0.0246 PIP_2)$$

As expected the first joint (abduction of robot hand (Slave) index finger) is highly affected by the abduction of the Metacarpophalangeal joints, and, other joints have less effect. All permutations of three values [min, 0, max] are applied to each joint separately. Finally, the sum of all distances is used as the return value of the cost function.
It means that the cost function indicates how close the curves were not just in one configuration but in 81 different configurations. The min and max joint values are given in [10]. Finally, the slave hand joint values are calculated and transmitted to the slave hand simulator and from there to the physics simulator. Using the configuration of the slave hand and a mesh model the simulator calculates the penetration explained in 4.3 and recognizes the grasping. The slave robot joint values are sent to the master side simulation where the physical model of the slave robot and hand is deployed. To synchronize the visualization for the operator the physics simulation also sends another copy to the AR interface.

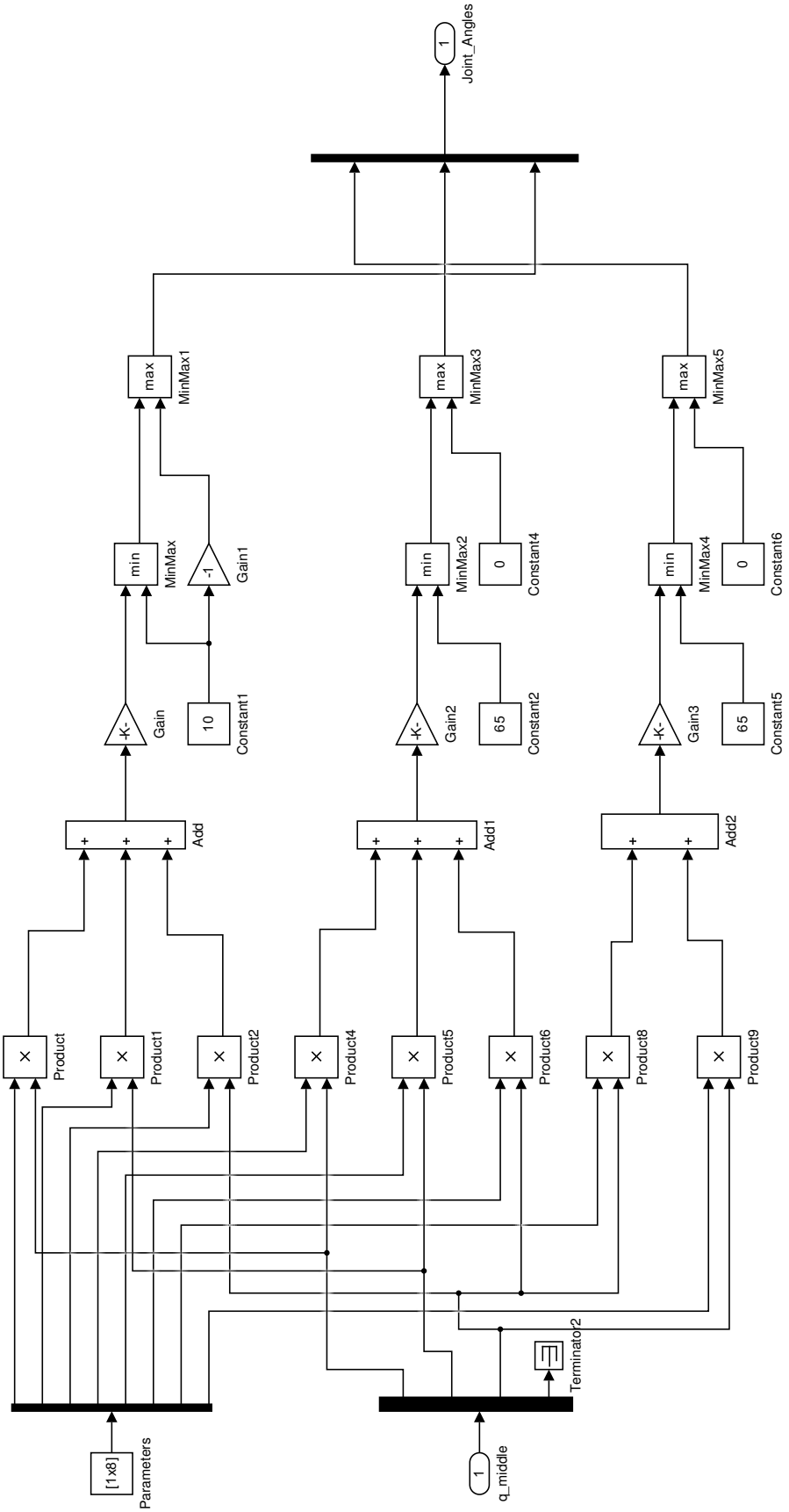


Figure 4.5: The figure shows the Simulink model for joint-to-joint mapping of the middle finger from a human hand to the robot hand. Parameters are the learned coefficients from the optimization and constant1, constant2, constant3, constant4, and constant5 are the joint limits. The gains, gain2 and gain3 are the feed-forward gains to manipulate the angles in case of inconsistency due to changes in the model.

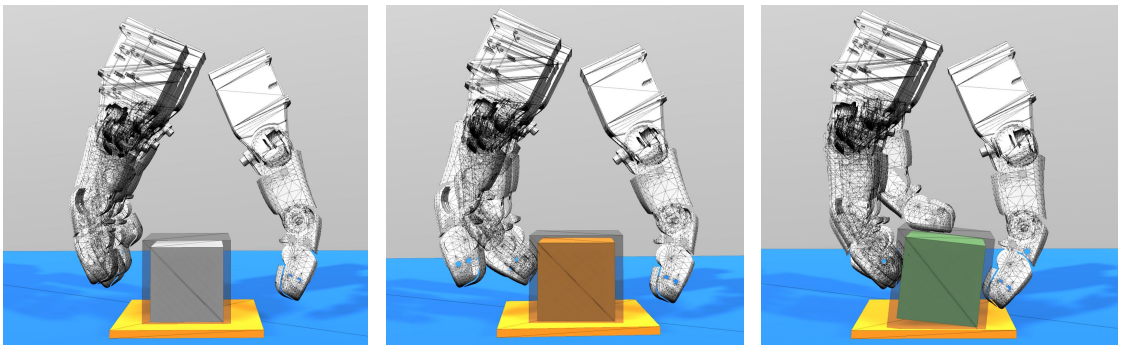


Figure 4.6: The figure shows different stages of grasping a cube. The left image shows the normal condition, the center image shows when the middle and ring finger activate the diaphragm. The right image shows the situation where the object is successfully grasped.

4.3 Grasping and Force Calculation in Simulation

Since MMT extensively uses a simulation environment, therefore using a simulation with a proper physics engine is imperative. The chosen physics engine must have the ability to

- Calculate penetrations on complex objects with convex hulls.
- Simulate rigid body dynamic behavior.
- Process in Real-Time.

Unity [81] is a cross-platform game engine with a powerful built-in physics engine. Due to its ability to facilitate designing three-dimensional, two-dimensional, augmented reality and virtual reality games it has been widely used in the industry. Although Unity is a powerful tool to simulate complicated objects with dynamic behavior but due to limited processing power the capabilities are restricted. For example, Unity does not support non-convex rigid body collision detection due to computation overhead and low demand in the market. An object is convex when there is no line inside the object that intersects with its surface mesh. Consequently, Unity has a default approach to deal with non-convex objects which is to approximate a new convex-mesh so-called the collider.
Unity does not support penetration calculation for soft objects, for example, human skin or elastic materials. Although, unity supports simulation for different tangential forces for instance friction the noise coming from the master robot makes the surface friction impractical. Surface friction is necessary for grasping, this tangential force is the main reason that the object stays within the hand during the movements. In the simulation, surface friction fails because of constant attach and detach of the fingertips and the object surface. This fluctuation causes the object to slide out of the hand right after picking up off the ground.

Other simulation software support tangential forces and penetration detection but they either suffer from the same problem (e.g. Unreal engine [62], Mujoco [80], VREP [13], and Gazebo [29]) or they were not commercially accessible for this work. To solve this problem using Unity a new approach is proposed so-called the diaphragm. The diaphragm guarantees a smooth attachment without any contact breakage. The breakage happens when one object mesh enters another object mesh without being able to apply any force. In unity, two colliders break into each other where one of them (e.g. robot finger mesh) is controlled kinematically. In Unity, the objects can be controlled in two different ways, kinematic and non-kinematic. The non-kinematic control uses forces such as gravity, friction, arbitrary forces and torques generated via built-in joints. On the other hand, the kinematic control changes the position and rotation of the object directly.
Although the diaphragm facilitates the grasping, due to high noise in object position, it may slip out of the hand, to address this issue an external controller is deployed on the object position. This external controller makes the grasp more persistent by using a kinematic control and eliminating all the forces including gravity, therefore, moving the object relative to the robot hand. There are several conditions to ensure grasping is complete,

- **Force equilibrium:** This condition ensures the forces applied from different fingers are facing toward each other. As figure 4.7 shows, to calculate the equilibrium the angle between two force vectors, the force vector from the thumb (F_{Thumb}) and the average force vector (F_{Opp}) from other involved fingers is compared to a threshold. To ease the grasping the threshold can be increased.

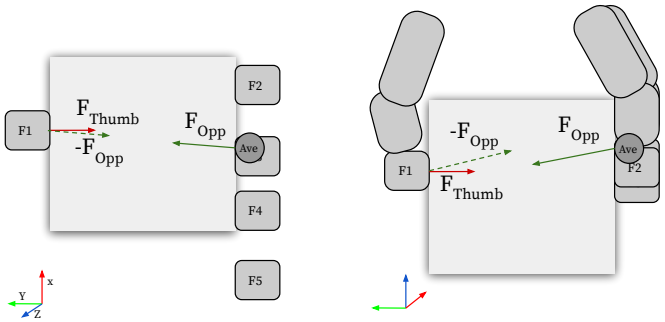


Figure 4.7: This figure shows the force equilibrium in 2D

- **Stability:** To increase the stability of the grasp the involved fingers are checked to have constant contact with the abject mesh for at least ten consequent frames. This ensures if the object slips away from a finger, that finger should not be involved. So, grasping is not complete unless the object is in a stable situation inside the hand.

As figure 4.8 shows, the depth and normal vector of the penetration calculated by the physics engine are sent to the Exodex Adam Simulink model. And, the

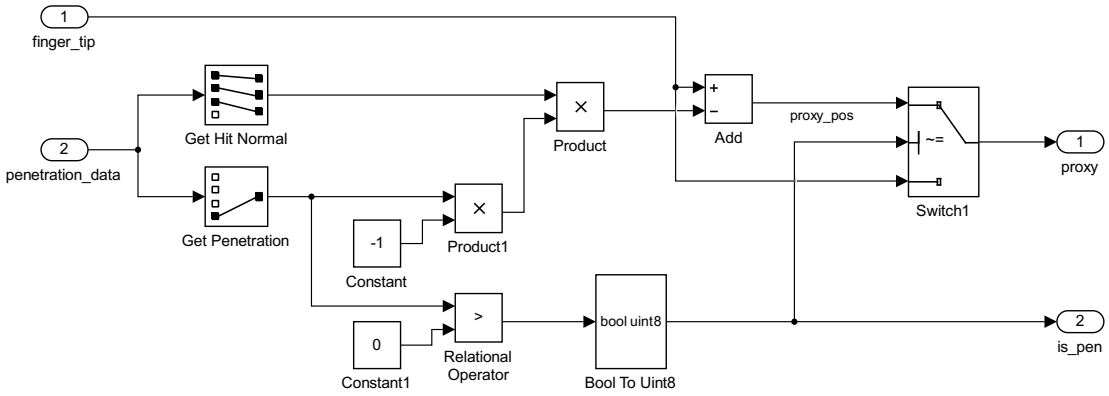


Figure 4.8: The figure shows the Simulink model for arbitrary object penetration detection. This model also calculates the pose of the fingertip on the surface of the object if necessary.

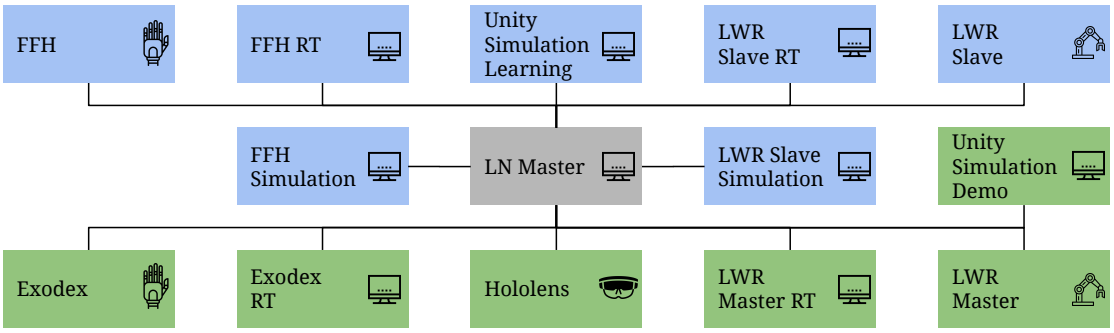
Simulink model uses the god object approach [50] to calculate the proper proxy position to guide the slave hand mesh outside of the object mesh, therefore, providing haptic feedback for the operator. As figure 4.8 shows the proxy position defines

$$x_{\text{proxy}} = \begin{cases} x_{fingertip} - (\vec{pen}\,\|pen\|)) & pen > 0 \\ x_{fingertip} & pen \leqslant 0 \end{cases} \tag{4.4}$$

The proxy position x_{proxy} defines the closest position to the surface of an object from each fingertip. Then this position is used as an attractor point for a spring-damper system which leads the fingertip to that position. Finally, the forces are converted to torques using the Jacobian of each finger and then sent to the motors. Due to the simplification in the haptic interface as it is shown in figure 4.4 the middle, ring and little fingers are derived using the middle finger position. Therefore the forces generated for these fingers are fed back to the middle finger in the haptic interface. And the contact points connected to the palm mesh, generate the force for the manipulator to generate a crisp haptic feeling for the arm.

Diaphragm

The diaphragm is a novel idea which uses an arbitrary virtual object that surrounds the target object to adapt contact forces between the hand and the objects. It has the same shape as the object but 20 percent bigger in size with a transparent appearance. Once one of the robot fingers enter the diaphragm, the physics engine takes over the low-level control. Consequently, the delay between the robot controller and the physics engine is eliminated and the penetrations can be detected in the early stage. The diaphragm also introduces a color hint, when it is activated the object changes its color to yellow. Then, the teleoperator knows when to keep the hand steady until the process of grasping is complete. Once the grasping conditions are satisfied the object becomes green and the object is ready to pick up.



4.3.1 Gesture recognition

To finish a task the teleoperator needs to release the object in a designated area. Therefore, an external signal (e.g. hand gesture) is required. The gestures can be recognized by analyzing the geometry of the human hand using the relative positions of the contact points. The Cartesian range of motion on each finger is measured by calculating the difference between two different gestures, power grasp and rest (open hand). Then the range of motion is used to recognize which fingers are bent, therefore, recognizing gestures such as pinch grasp, pointing, power grasp, and rest. The rest gesture is used as the designated gesture for releasing the object after a complete grasp.

4.4 Hardware

This section is an overview of the hardware architecture and how the hardware and tasks are distributed in the master and slave side and also how they communicate.

4.4.1 Architecture

Due to advantages such as easier debugging process and problem rooting, the system uses a star topology. The figure 4.4.1 shows the connections and nodes in the master side (green nodes) and slave side (blue nodes) that are defined as below:

- **LN Master**: This block regulates the communication between nodes as a middleware. It also manages the processes and the dataflow between them.
- **Slave Side**
  - **FFH**: This block contains the Five Finger Hand (FFH) hardware and low-level controller.
  - **FFH RT**: This block executes the FFH high-level controller (Simulink executable files) and also the communication to the FFH hardware.

  - **LWR Slave**: This block contains the Light Weight Robot (LWR) hardware and low-level controller.
  - **LWR Slave RT**: This block executes the LWR high-level controller (Simulink executable files) and also the communication to the LWR hardware.
  - **Unity Simulation - Learning**: The block contains processes related to physics simulation containing the environment, FFH, and LWR. It contains seven different instances of so-called workers that function in parallel to speed up the learning process.
  - **FFH Simulation**: The block utilizes seven independent FFH Simulink models running in parallel to provide high-level control for physics simulation.
  - **LWR Slave Simulation**: This block comprises seven independent LWR Simulink models running in parallel to provide high-level control for physics simulation.

- Master Side:
  - **LWR Master**: This block contains the LWR hardware and low-level controller.
  - **Exodex Adam**: This block contains the Exodex Adam hand interface hardware and low-level controller.
  - **Hololens**: This block contains a Microsoft Hololens and a simulation of the environment and the slave device. It provides visual stereo representation for the user.
  - **Exodex Adam RT**: This block executes the Exodex Adam hand interface high-level controller (Simulink executable files) and also the communication to the hand interface hardware.
  - **LWR Master RT**: This block executes the LWR high-level controller (Simulink executable files) and also the communication to the LWR hardware.

4.4.2 Exodex Adam Haptic Interface

Light Weight Robot (LWR)

Nowadays industries utilize robots to increase efficiency and speed while keeping high precision. The first versions of industrial robots were designed to be used in an isolated environment such as cages due to safety issues for humans. The industrial robots increase precision by increasing the stiffness of the movements and therefore less drag and drift from the target configuration. Human-robot interaction and collaboration can add high intelligence and accuracy to task performance. Therefore industrial robots capable of providing safety for human operators have

been designed. The KUKA DLR-LWR 4+ robot is one of the best currently available robots that provide safety for HRI by integrating Force/Torque sensor to each joint. Therefore the robot can detect the small collisions at the joint level and can prevent them by stopping the robot in real-time. The LWR has seven Degrees of freedom (DOF) which provide redundancy for the controller. The redundancy is a result of having more DOF than the dimensionality of space (three translational and three rotational) and means for each accessible pose in the task-space there is an infinite number of configurations.

Simulink Model

The LWR Simulink model that interacts with the robot directly and in real-time is initially implemented by Thomas Hülin for the DLR HUG project [22]. The figure 4.9 shows the Simulink model for LWR 4+ manipulator arm, the major components are defined as below:

- **Control and GUI**: The settings related to switching between the control modes (Position, Torque and cartesian), the activation of the telemanipulation. Also, resetting the anchor position is defined in this block.
- **High-Level Control**: This block receives and processes the forces coming from the Exodex Adam Simulink model. It also modifies high-level parameters for different control modes, for example, joint limits and null-space forces.
- **Robot**: The forces are converted to torques to be deployed on the robot. Also, the torques for joint limit protection and torques to avoid knotting are calculated here. The telemanipulation block 4.1 is located in this block. The gravity compensation also located in this block plus the dynamic and kinematic calculations for the robot. A homogeneous transformation is calculated for each joint from the base to the last joint. The plant containing the interface to the real robot and its identical simulation is located in this block.
- **Visualization**: The useful and meaningful data such as torques, joint values, etc are visualized using scopes in this block. These data are also published on the network by Links and Nodes middleware.

Hand Interface

Exodex Adam is a novel haptic interface designed by the MODEX lab at the German Aerospace Center [35, 37, 36]. This setup combines an LWR with a custom design end-effector. The end-effector combines five torque-controlled robotic fingers where each finger has an anthropomorphic structure with three DOF with two coupled joints. The end-effector of each finger has a gimbal and a magnetic clutch

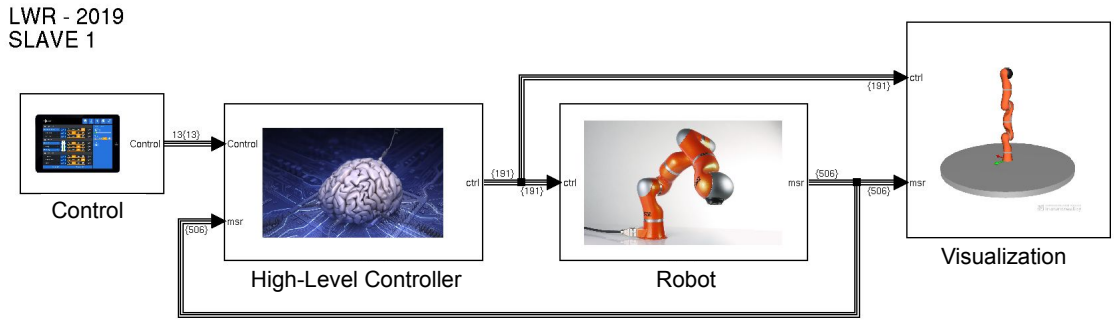


Figure 4.9: The figure depicts the Simulink model of the LWR of the Exodex Adam setup. The model consists of four modules, Control, High-level controller, Plant, and Visualization.

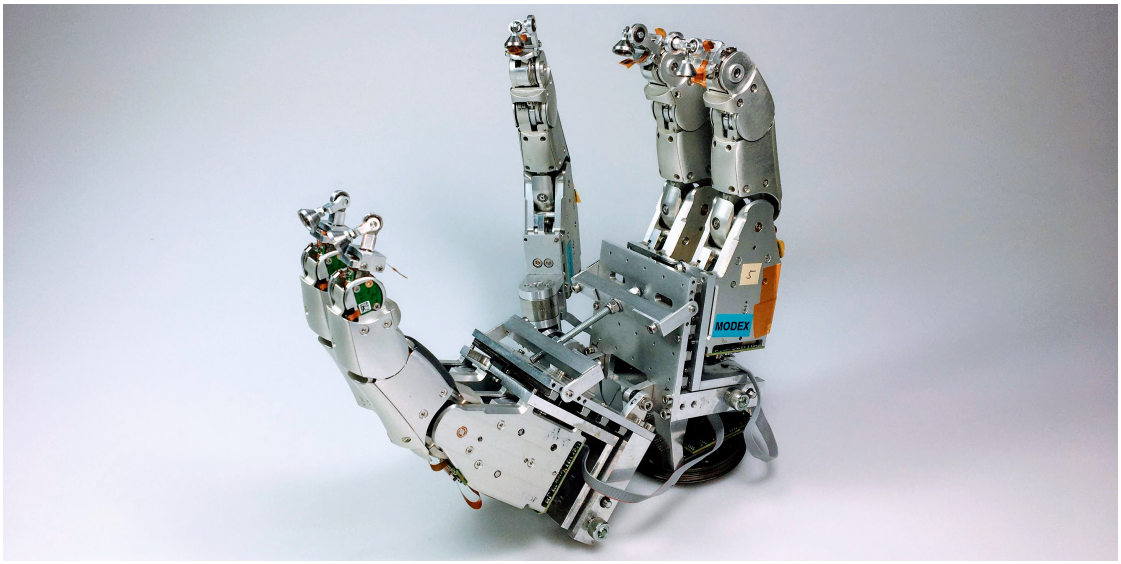


Figure 4.10: The figure depicts the hand interface of the Exodex Adam setup. The hand involves five fingers that attach to the human hand using magnetic clutches.

that attaches to the glove worn by a human hand. The magnet allows fast detachment in the case of an emergency. The Simulink model for the Exodex Adam end-effector runs in different real-time PC and runs the control loop and haptic feedback.

Simulink Model

The figure 4.11 shows a brief view of the Simulink model which controls each finger separately. There are four major modules defined as below:

- **Command**: This component defines the setting for each finger (e.g. joint limits and feed-forward gain). The control mode of each finger can be also selected between torque and position control. The calibration of joints also happens in this block.

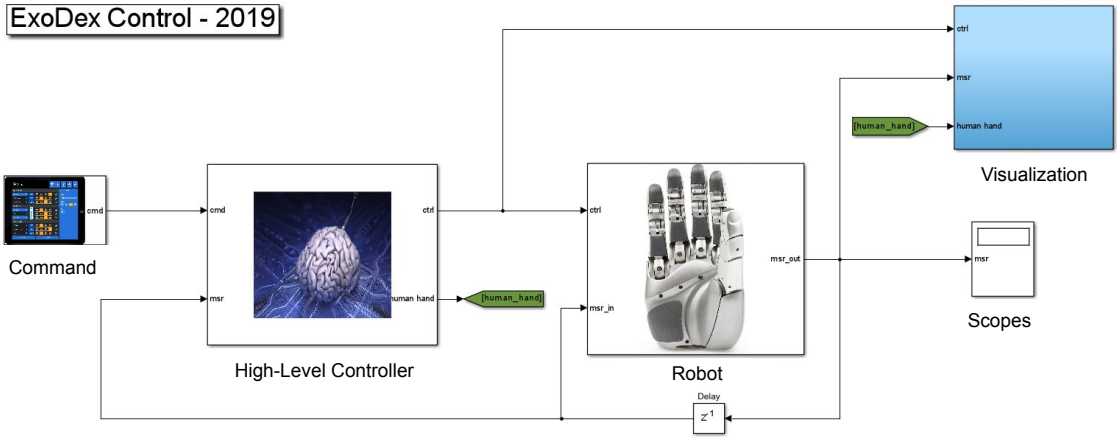


Figure 4.11: The figure depicts the Simulink model of the hand interface of Exodex Adam setup. The model consists of four modules, Command, High-level controller, Plant, and Visualization.

- **High-Level Controller (HLC)**: Here, the penetration data using the virtual environment are transformed into force feedback to be sent to the low-level controller. furthermore, the transformations of each joint and link are calculated using the complete DH parameters from the LWR base to the fingertips. The position of the fingertips is used for inverse kinematic calculations to determine the human hand configuration. the kinematic chain of the human hand (DH parameters) are given by Magnetic Resonance Imaging(MRI) found in previous studies at DLR [10, 73, 74]. Moreover, The properties of the virtual objects, for instance, the stiffness and friction are defined in this block.

- **Plant**: This component is responsible for direct communication between the Simulink model and the haptic interface through the RobotKernel block. The RobotKernel is a runtime-configurable hardware abstraction framework designed and implemented at DLR. It encapsulates device drives in dynamical loadable shared modules and provides generic interfaces to the hardware. To run Simulink in a simulator mode(Normal), the RobotKernel is replaced by a block with identical inputs and outputs that imitates the robot dynamic behavior. Here, the virtual environment forces calculated in the HLC are converted to torques and finally to pulse width modulation (PWM) signal to be deployed on the joints. Finally, the measured data from the hardware are processed, filtered and transformed into Cartesian form to be used in HLC.

- **Visualization**: The useful and meaningful data such as torques, joint values, etc are visualized using scopes in this block. These data are also published on the network by Links and Nodes middleware.

Control

The figure 4.12 shows the control loop for each finger based on [24] where the transfer function of the impedance felt by the user is defined as below:

$$\frac{u}{F} = \frac{1}{Z_u + \frac{Z_m}{1+K} + \frac{Z_e}{1+K}} \tag{4.5}$$

Where Z_m, Z_u, Z_e define the impedance of the haptic interface, user and the virtual environment, respectively. The parameter F determines all external forces and F_u defines the force exerted by the user. The Z_m and Z_e are reduced by the factor of $1 + K$ which means the exerted forces by the virtual environment are weakened. Furthermore, K is the gain that is used to intensify the exerted forces by the user to facilitate the movement of the haptic interface. Since the virtual environment interaction forces are defined by Z_e, these forces are replaced by the physics simulation in the architecture. As the figure shows the forces from the virtual environment F_e are opposing the forces applied by the user. Finally, the command forces F_cmd are generated to be sent to the haptic interface.

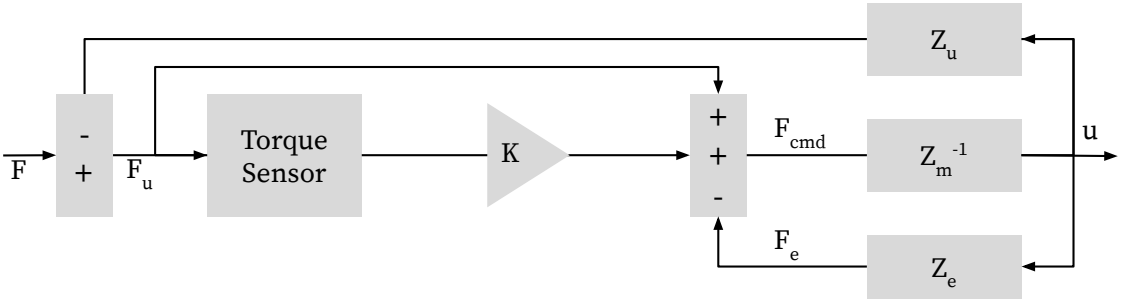


Figure 4.12: The figure 4.12 shows the control loop for each finger based on [24]. Z_m, Z_u, Z_e define the impedance of the haptic interface, user and the virtual environment, respectively.

Attachments and Configurations

The figure 4.13 shows the human hand is attached to the interface. The magnetic clutch coupled with each fingertip of the interface is the physical attaching point of the interface and the user. The human hand is covered by a glove combined of four parts, 3 finger caps each with a magnet on top for the thumb, index and middle fingers and one half-glove to cover the palm with two magnets. The glove does not have any sensors and it is only used for attachment purposes. The attaching points on the interface are connected to a three DOF gimbal that allows free rotation around the gimbal joint. There are eight non-actuated joints on the interface to modify the configuration of the fingers and the interface to match the user-specific requirements such as the hand size.

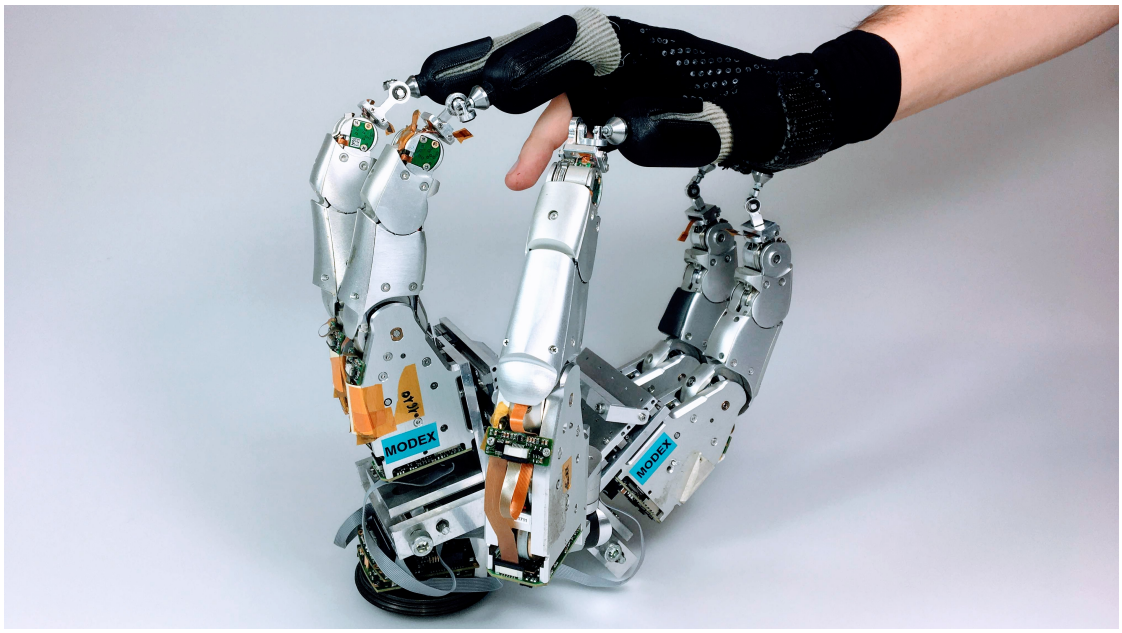


Figure 4.13: The figure shows the human hand attached to the interface. The magnetic clutch coupled with each fingertip of the interface is the physical attaching point of the interface and the user.

4.4.3 Augmented Reality

Microsoft Hololens One was selected as the AR platform due to the short supply of high-quality augmented reality goggles and also the limited number of options. This device provides a stereo representation of the virtual environment as an overlay on the operator's real surroundings. Therefore, the operator can interact with both environments at the same time. Since the device is not supported by links and nodes middleware a customized communication system has been developed for low delay communication setup using UDP protocol that translates the packages to be used in the AR device. Due to the onboard low-performance processor on the Hololens (Intel 32-bit (1GHz)), the heavy physics calculations are done in a separate powerful PC and the results are sent to the device as positions. The packets sent to the Hololens include

- The joint angles of the slave LWR and the fingers,
- The joint angles of the master LWR,
- The virtual object positions,
- The status of the grasp,
- The reference frame of the slave device.

Therefore, the Hololens is used for rendering the scene with no physics. The augmented 3D models of the table, objects, and robots are processed in Unity and anchored in the workspace using Vofuria. Vuforia is a software development kit that facilitates augmented reality application development in the Unity engine [1]. It uses an image (Image Target) with a high number of features to determine the frame of reference for the 3D models in the operator's point of view. Moreover, it uses multiple sensors to track the position of the headset when the image is not in the device range of view. When the image target is recognized the 3D virtual model appears on top of the real device. Once two environments are matched the master device and the slave LWR model is removed from the virtual environment leaving just the FFH model to narrow the operator's attention down to the important parts of the scene. The model has been placed in a way that when the operator's hand attaches into the master device the operator's hand matches the FFH augmented model. Therefore, the user can observe the influence of the actions on the robot hand. The lighting of the visualization also matches the lighting of the real environment since the master device is stationary, this improves the realistic view of the virtual environment and therefore helps the operator visually. Moreover, The diaphragm is visualized in the AR model and the user can interact with it in real-time.

Once one of the fingertips enters the diaphragm the object gets selected and the material color becomes yellow and the teleoperator realizes the diaphragm is successfully activated and needs to wait until the object becomes green and the grasping is complete.

4.4.4 Links and Nodes

If two machines run real-time processes the communication between them also must be real-time. DLR Institute of robotics and mechanics has designed a new middle-ware so-called Links and Nodes (LN) to manage the processes and the communication between them. Using LN local processes can share data using shared memory and two distant processes in two different PCs can share data using TCP or UDP depending on the type of request. The LN master is the central process of LN which runs on a separate PC. The LN master has a GUI as shown in the figure to facilitate the management of multiple processes at the same time. It uses the topic/service architecture similar to ROS to share data between two processes. The topic is a concept of sharing information that any process can read the data just by having the address of that specific topic but just one process can write on it at the time. And the services allow for sharing information without loss by using a handshake for requesting each packet. So for reading data with high frequency, it is better to use topics instead of services.

4.4.5 Slave Device

Master LWR, Slave LWR, The Differences

Due to the anthropomorphic configuration of the slave device to imitate the human arm, the LWR used in the slave side has joint offsets. The joint offsets are deployed to increase the work-space of the robot in the human-arm configuration when the first two joints imitate the shoulder and others play the elbow and wrist roles. The custom configuration keeps the joints away from the joint limits. furthermore, an extension to the last joint is deployed as a solution to the common singularity problem in LWR. The customized configuration is first used in Justin robot designed by DLR. Due to hardware similarities, the same Simulink model (figure 4.9) has been used but the DH parameters and the Jacobian block is borrowed from the Justin robot model. Another real-time PC is used for low and high-level controllers. The final model has been optimized and duplicated for seven different workers. The LN data ports are customized each instance of the Slave model, the ports connect the RL engine and physics simulator to the Simulink models.

Five Finger Hand (FFH)

The end-effector used in the slave device is a customized version of the DLR hand so-called FFH hand. The differences are in the position of the 5th finger which is opposing other fingers to facilitate grasping and in-hand manipulation.

4.5 Evaluation

The experiment includes multiple sub-tasks to evaluate the performance of proposed architecture in real scenarios. Each sub-task is divided into two main parts,

- **Demonstration**: The operator interacts using the haptic interface attempts to complete the assigned tasks in a simulated environment. This step continues until the first successful execution of the task by the operator is finished. Due to the high influence of the human factor in this step, a small user study with 10 participants has been conducted at DLR. The participants were all male and had experience with teleoperation devices. Due to the lack of the second system for comparison, the standard measures such as the system usability score (SUS) and NASA task load index (NASA-TLX) has been used. The user-study serves our purpose to test the system with the help of experts and gathering data to evaluate RL techniques that are used in robotics in the context of learning high-dimensional motor control tasks.

- **Learning**: After the demonstration by the operator, the DMP parameters were sent to the slave side and immediately translated to a Cartesian trajectory corresponding to the new object position. Then the trajectory was deployed on the simulation worker 1 to evaluate the new trajectory. The simulation determines whether the grasping was successful. In the case of

failure, the trajectory is sent back to the RL unit which adapts the trajectory to the new condition. This step includes trajectory deployment, learning, and evaluation of the architecture and the learning approaches in the simulation. The evaluation is conducted under different conditions,

- Placing the object in different distances from the initially taught one shows how often the task requires RL also shows the performance of the system without uncertainty.
- Applying uncertainty to the object position imitates the inaccuracy in the vision systems shows the performance of the system when using tactile feedback as sensory input.
- Grasping objects with different shapes shows the performance of the approach when facing different geometrical shapes.
- using different hyper-parameters shows whether the final results are compared based on the best trials and how changing parameter effects the behavior of the system.

The slave device was used to test the final learned trajectory and was not involved in the learning process. Learning in simulation guarantees the safety of the final trajectory for the slave device where any anomaly in the shape of the trajectory is chacked before deploying it on the slave device. The final part of the experiment is performed using different learning rates for policy gradient and, exploration rates for parameter perturbation methods to guarantee the fairness of the comparison by choosing the trials with the best performance.

4.5.1 Preliminary User-study Procedure

The user-study has been conducted to collect data and record trajectories to evaluate the overall architecture. It involves different tasks, including pick and place of geometrically different objects. Before the experiment, the participants were briefed verbally and written about the haptic interface, experiments, and details about safety procedures, and necessary information in the case of emergencies. The experiments were conducted under DLR safety regulations, All participants took the safety procedure course to work with lightweight robots. After signing the consent form the interface glove and the finger caps were chosen proportional to user hand size and comfort.
The necessary calibrations were done to reset the torque sensors before attaching the hand. The gesture recognition personalization was done based on the participant's hand size by asking the user to open the hand completely and close it completely. Next, the AR goggles were put on the users head by the experimenter and the quality and comfort were confirmed verbally. During the briefing session, information about the diaphragm, the expected forces, the meaning associated

with the color hints, and some guidelines about grasping were given to the participant. The participants were allowed to train on each task before the recording of the trajectory. Each task contained two parts, Pick and place, which they were separately recorded. The duration of performing each part was 10 seconds.

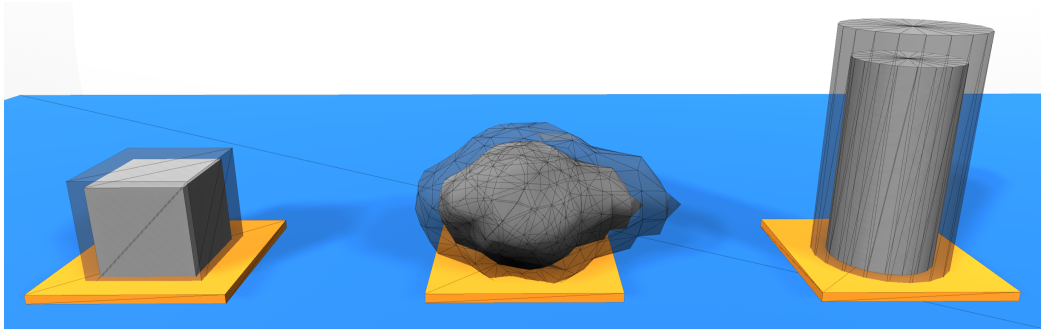
4.5.2 Tasks

The experiment involves four different tasks:

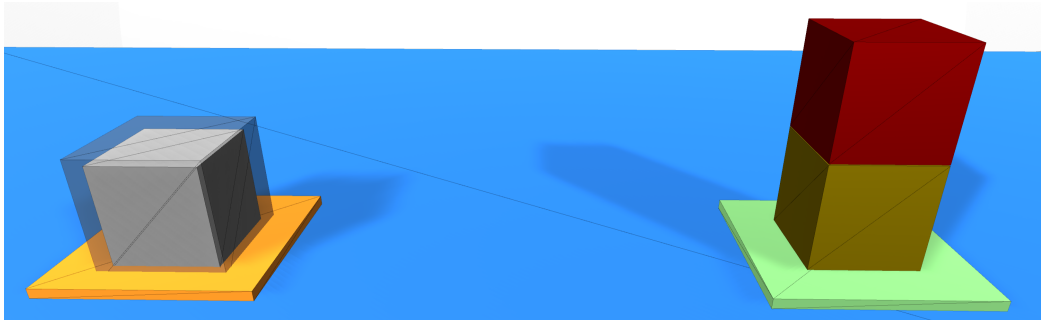
- Pick and place of a box (5cm x 5cm x 5cm) from the orange place holder to the green place holder to evaluate the performance of the system in grasping an object with parallel surfaces.

- Pick and place of a cylinder (10cm x 5cm) from the orange place holder to the green place holder to evaluate centric grasping

- Pick and place of a rock shape object from the orange place holder to the green place holder to evaluate the performance of the system in grasping objects with arbitrary shapes.



- Build a tower by picking the box from orange place holder and put it on the boxes on the other side to evaluate the performance of the system to deploy high-level tasks.



4.5.3 Questionnaire

System Usability Score (SUS)

The system usability score so-called SUS is a standard score that determines the usability of a product. The participant ranks the system immediately after performing all tasks in a subdivided scale of 20. The questionnaire includes different statements that the users choose the closest option to their experiences. The statements include

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

NASA Task Load Index (NASA TLX)

The NASA Task Load Index (NASA TLX) is a multidimensional subjective assessment tool to evaluate the workload of a task or system [20]. The participant ranks the system immediately after performing each task. The total workload is subdivided to five different tasks:

- **Mental Demand:** How much mental or perceptual activity was required(e.g. thinking, deciding calculation, remembering, looking, searching, etc)? Was the task easy or demanding, simple or complex?
- **Physical Demand:** How much physical activity was required (e.g. pushing, pulling, turning, controlling, activating, and, etc)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?
- **Temporal Demand:** How much time pressure did you feel due to the rate or pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?

- **Performance:** How successfully do you think you were in accomplishing the goals of the task set by the experimenter(or yourself)? How satisfied were you with your performance in accomplishing these goals
- **Frustration Level:** How insecure, discouraged, irritated, stressed and annoyed versus secure, gratified, content relaxed and complacent did you feel during the task?

Chapter 5

Results and Discussion

In this chapter several metrics are used to examine the performance of the system in different tasks, namely:

- **Trajectory Cost**: The continuous decrease in the cost of the trajectory during learning trials defines if the approach can minimize the cost in a limited time. The trajectory cost contains the quality of the grasp by taking into account the number of fingers involved and the smoothness of the trajectory by considering the acceleration. In this thesis, the cost and lost are used interchangeably,
- **Success rate**: The continuous increase in the success of different instances of a trajectory throughout the learning determines if the approach can improve the overall performance
- **Total update number**: The number of updates that an algorithm requires to successfully finish a task shows the performance of the method.
- **Trajectory shape**: The final trajectory shape indicates whether the changes were close to initially taught skill or if they were drastically different. The unnecessary changes in trajectory shape are considered as a disadvantage for the approach.

Finally, the results from user-study show how well the system performs when interacting with a human operator. The System Usability Score (SUS) and NASA Task Load Index (Nasa-TLX) of the system. This chapter also discusses the eligibility of the result in detail.

5.1 Grasp Without Uncertainty

The first set of results compare different algorithms in a scenario where the slave robot grasps a box. To evaluate the approaches the target pose has been changed incrementally in different directions (on the XY plane or table surface). The figure 5.2 compares the cost of trajectories generated by different methods in an extreme

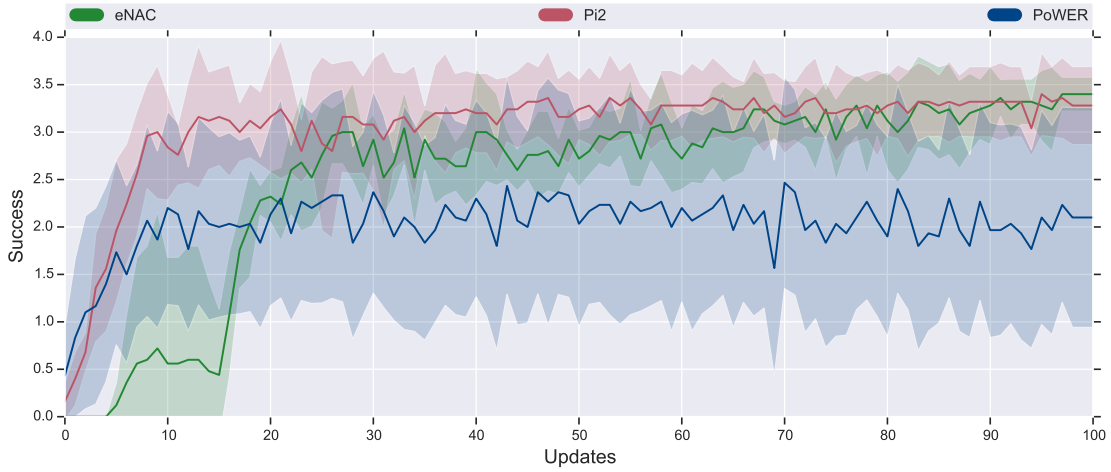


Figure 5.1: The figure shows the success rate after each update. The vertical axis indicates the number of successful roll-outs and the horizontal axis shows the update step. As it is shown, although the PoWER algorithm shows a sudden rise the steady-state stays around two which is lower than PI^2 which shows a milder increase in the success rate initially. On the other hand, eNAC has a slower increase and also drop but it achieves a constant increase in success rate after 10 updates which continue till the end.

condition where the object is located more than 40 cm away from the demonstrated target in both directions (40 cm in X and 10 cm in the Y direction).

The lines indicate the average cost of over five different learning sessions and the shadow around each line indicates the standard deviation. The initial cost relates to the trajectory generated by DMPs, and later, each algorithm attempts to reduce costs within a limited timeframe. The figure shows all the methods decrease the cost and stabilize the learning with a low cost where the grasping happens. The PI^2 and PoWER rapidly decrease the cost, but PI^2 shows higher stability by less fluctuation while reaching a lower value in its steady-state. The eNAC algorithm has a more gradual decrease and unusual fluctuations in the steady-state. The value of the steady-state of the eNAC algorithm shows the final cost is higher than the other two.

Furthermore, figure 5.1 compares the success rate of the three algorithms. Here, the lines indicate the average success rate over five different learning session and the shadow around each line indicates the standard deviation. The figure shows the PoWER algorithm rapidly increases the success rate but it reaches the learning steady-state sooner than the others. Although, the PI^2 algorithm shows significant growth with a high steady-state value and eNAC has a slight success increase and also a decrease after nine updates but it starts a steady increase after 15 updates which continue till the end and reaches the highest success. Therefore it shows the approach may need more time to reach better results than PI^2 and PoWER.

Figure 5.3 compares the number of required updates to achieve a successful grasp. Moreover, zero updates indicate the RL was not activated and the DMPs success-

fully compensate for the changes. The box charts indicate the results gathered from five different learning sessions for each algorithm.
As it is shown, the learning is required for the objects with more than 10 cm positional difference Δx where the median of the required updates is one. Here, all the approaches have similar performance, therefore, more challenging scenarios/tasks need to be analyzed. But when $\Delta x = 40$ cm, although PoWER needs a higher number of updates and the outliers show eNAC and PI^2 face occasional difficulties to find a solution. It is worth to mention $\Delta x > 40$ cm is out of the robot's reach, therefore, the positions beyond this limit have been excluded.
Besides, since it was already mentioned in the figure 3.6, the figure 5.4 confirms the changes in Y are harder to compensate for the learning since the y-direction affects the approaching angle and also the work-space limits are closer to the initial object position. The difficulty is due to the geometry of the robot hand, that changing in Y brings the object closer to the robot base, therefore, the opposing thumb might collide with the object before an actual grasp happens.
The figure shows when $\Delta Y = 10$ cm, RL approaches demand more learning updates than when $\Delta X = 40$ cm. Here, the PoWER algorithm has a better performance by having a smaller inner fence and lower median. The PI^2 has the same average but has a wide boundary. The eNAC has the same issue plus a higher median and also an outlier which shows the algorithm needed more than 50 updates to find a proper solution.
Figure 5.1 shows the final trajectories generated using different algorithms. Trajectories have been calculated with no perturbation after the last trial which led to a successful grasp.
The figure shows eNAC drastically changes the trajectory in T_X and T_Z while the

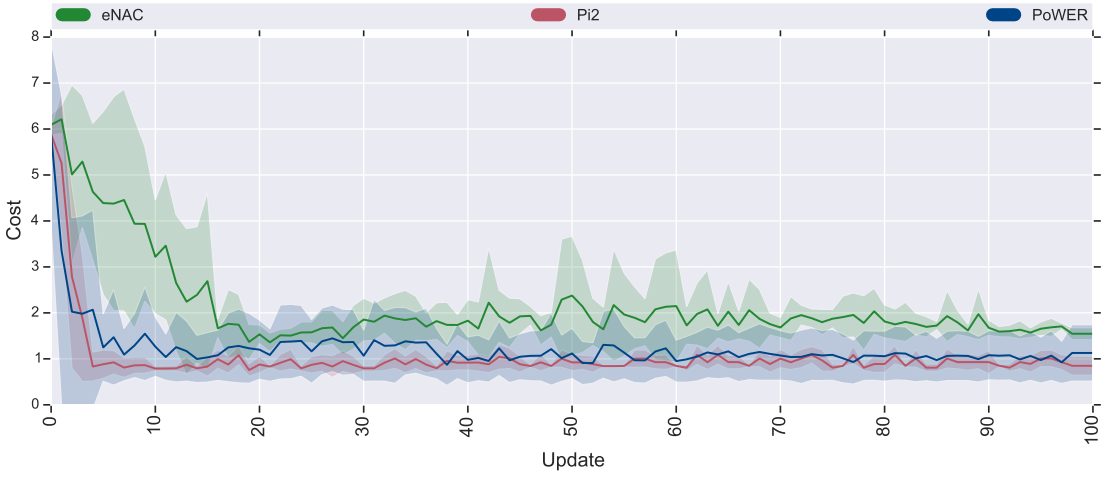


Figure 5.2: The figure shows the learning cost over 100 updates, each update contains seven roll-outs performed in the simulation. The targets related to this plot were placed about 50 cm away from the demonstration. The figure shows PI^2 and PoWER has a very steep decrease in the cost of the trajectory while eNAC has a slower convergence and less stability with high fluctuations.

changes by PoWER algorithm are applied on T_Z and R_X. It is observed that PI^2 has the lowest modification in all dimensions. The same results are achieved by [79].

The figure 5.1 shows the number of updates needed for the algorithm to be able to grasp the rock successfully. The results show the changes in X do not require learning as much as changes in Y and DMP transformations can compensate for the differences. But the anomaly in $\Delta X = 5$ cm is because the object is located on a place holder that the place holder stays still and the object moves so since the place holder is thick when the object is on the edge the object gets tilted and makes the object configuration different.

In this case, eNAC has the worst performance by requiring more than six update trials. In $\Delta X = 5$ cm and $\Delta X = 10$ cm also the higher bound is higher for eNAC in compare with PI^2 and PoWER.

To be able to show if the same results are achievable by the real robot, the same trajectory was deployed on the real and simulated robots, simultaneously. As it is shown in figure 5.8, the difference between the deployed trajectory at the end is less than five millimeters. But, the trajectories have small differences in the shape. At the start, the simulation drifted almost one cm, the trajectory has a shift at first which is due to initial drift in the simulation. Moreover, the real robot has a flatter plot when it reaches the highest value, and this flat curve coincidentally synced the shape of the two trajectories afterward. If the shape of the trajectory is to avoid an obstacle, for example, if an obstacle is placed in -0.42, the real robot will crash into it.

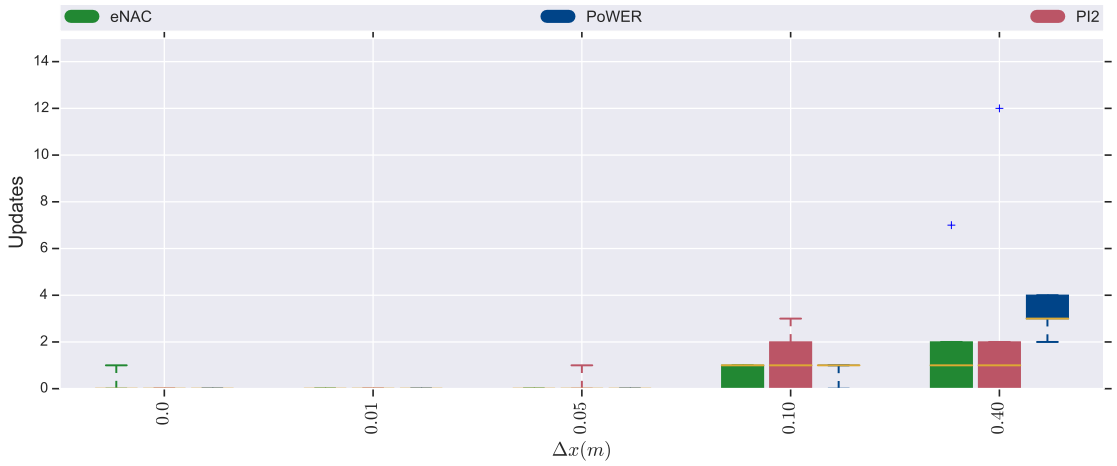


Figure 5.3: The figure shows the update number required for different approaches to find a solution. The horizontal axis indicates the distance from the demonstrated target and the vertical axis shows the number of updates. The positional differences are applied on X direction.

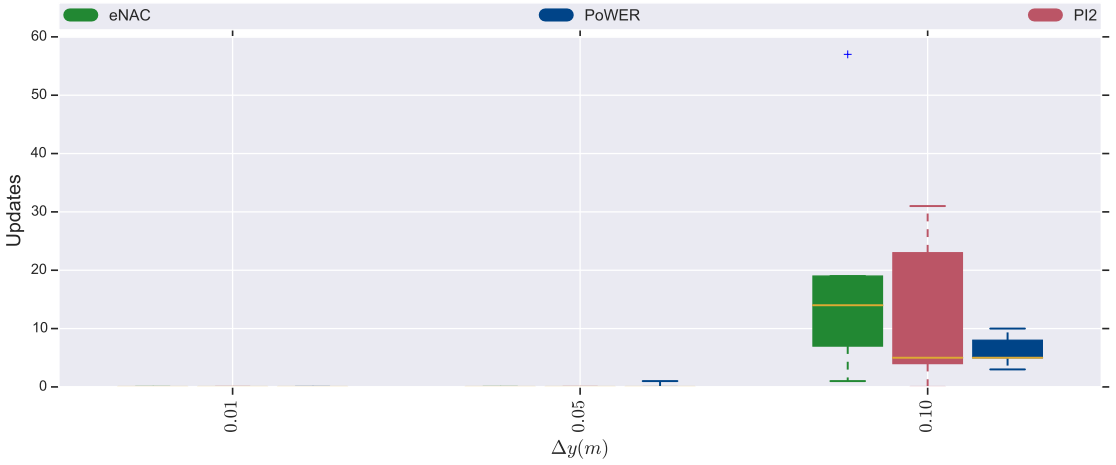


Figure 5.4: The figure shows the update number required for different approaches to find a solution. The horizontal axis indicates the distance from the demonstrated target and the vertical axis shows the number of updates. The positional differences are applied to the Y direction.

5.2 Grasping Under Uncertainty

The figure 5.2 compares the number of updates required to achieve a successful grasp under uncertainty. As it was mentioned before DMPs are powerful tools but as the results show they fail with small uncertainty in the object position. By increasing the uncertainty the number of required update trials increases as expected. The medians in the figure show the number of required updates decreases when the uncertainties reach six cm. The drop might be due to the goal exploration rate $\Sigma_g = 4$ cm so the end-effector does not hit the object since the object is not in the exploration range, initially. Therefore, smaller exploration rates were evaluated, but the results were not satisfying and the RL agent mostly failed to reach the object in 100 trials. The failure is due to the low probability to find the object since the haptic feedback is used as a reward so as long as the object is not touched by at least one of the fingers the agent does not receive any reward. For example, When the uncertainty is six cm but the exploration rate is just two cm the agent needs at least three trials toward the object with maximum step size to gain reward from touching the object which is quite unlikely to happen.

The figure 5.2 shows the trajectories generated under different uncertainties in object position. As the trajectories illustrate, the agent has successfully found several solutions for each trial with different uncertainties. The final stage of each trajectory shows where the end effector has grasped the object. The final values of the trajectories in comparison with the final values of the demonstration show the approach has successfully found the target. For example, the generated trajectory has the largest deviation when the uncertainty is seven centimeters. Since the robot hand is relatively big, the trajectories with uncertainties less than three cm are not corresponding to their plots. For example, the trajectory generated for two

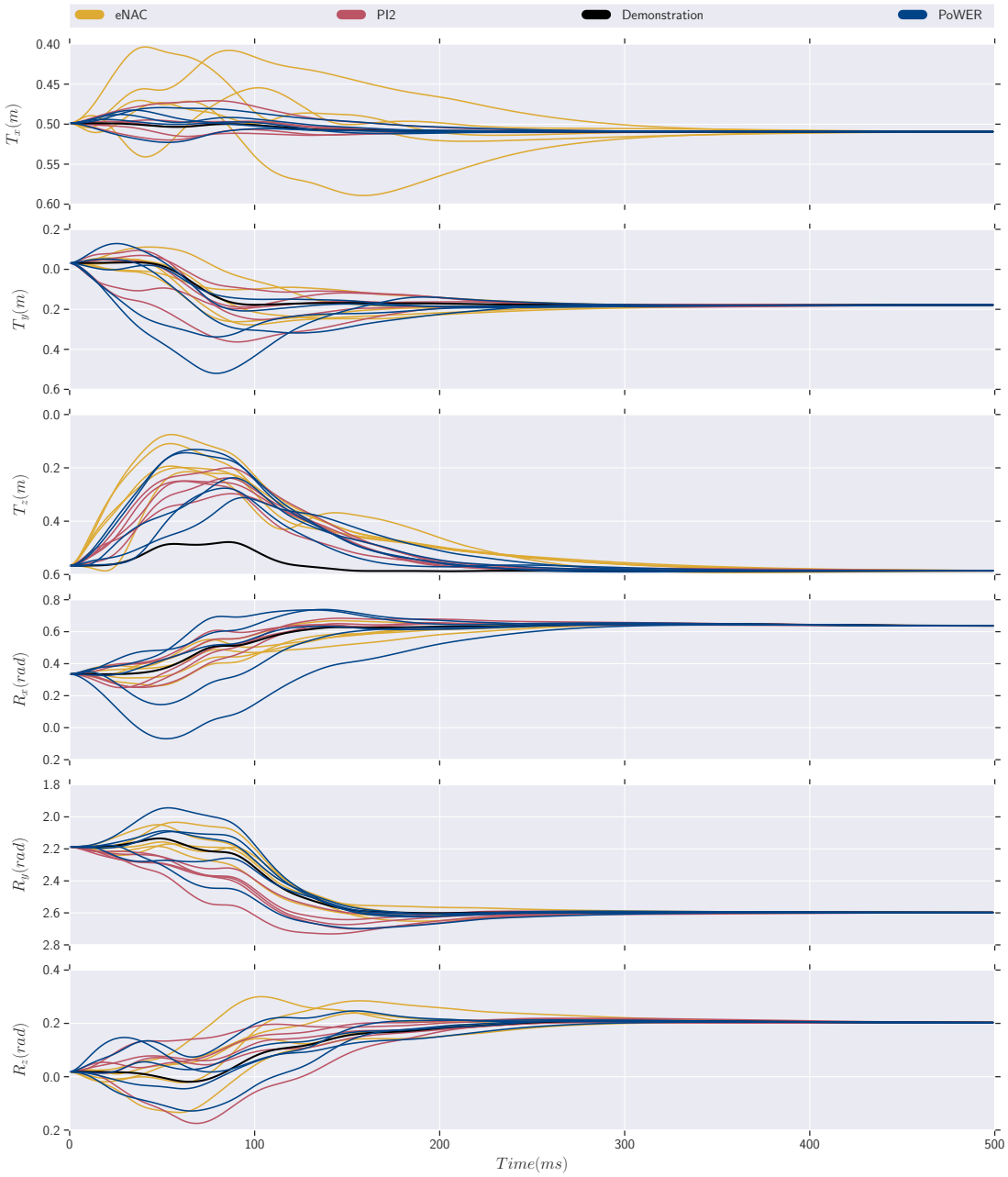


Figure 5.5: The figure shows the trajectories generated by different approaches. In T_x, the eNAC trajectories show a high divergence from the demonstrated trajectory. The trajectories generated using PI^2 have the lowest deviation from the demonstration.

cm uncertainty has less deviation at the end than the trajectory generated for one cm uncertainty.

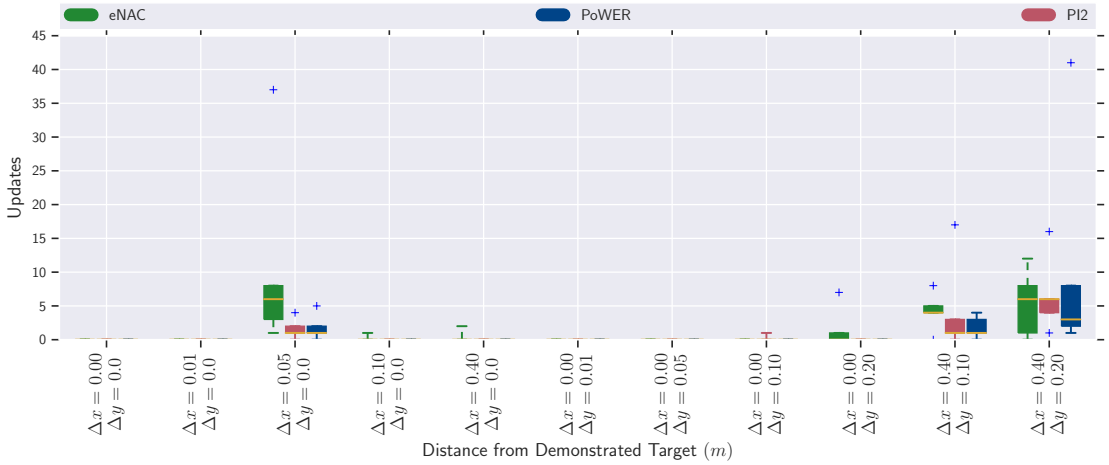


Figure 5.6: The figure shows the number of updates that different RL approaches need to learn to grasp the rock when there is a deviation from the demonstration target in X and Y directions.

Figure 5.7: The figure compares different approaches in grasping the cylinder. The result shows a small deviation from the original position of the object during demonstration results in a DMP failure. PI^2 shows the best performance while eNAC requires the highest number of updates.

5.3 Userstudy

The figure 5.12 compares different tasks with the results from the NASA-TLX questionnaire. The results show the pick and place of the cylinder require the highest mental demand, temporal demand, effort, and frustration while the box

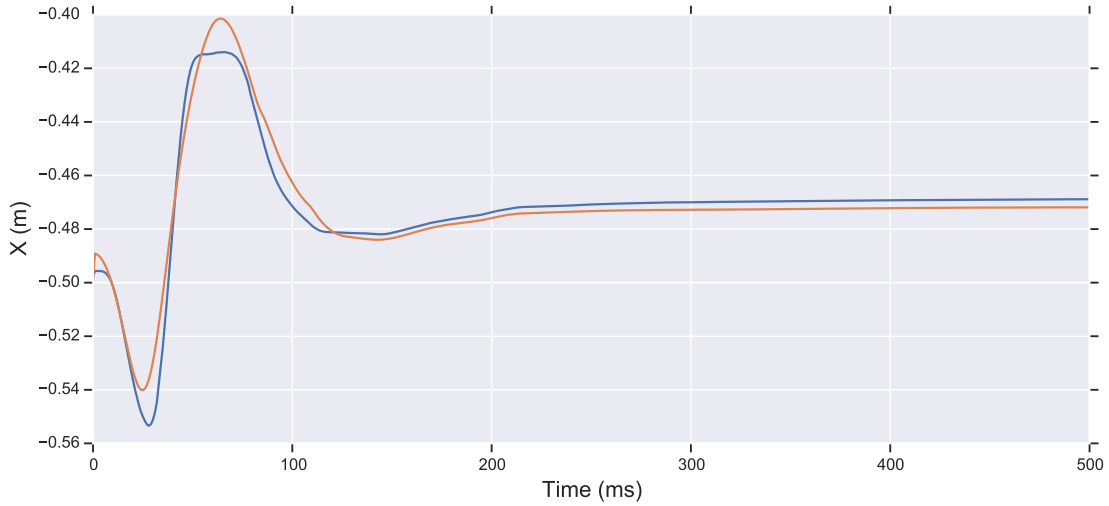


Figure 5.8: The figure compares the trajectories deployed on the simulated and real robots. The vertical axis determines the end-effector position in the X-axis and the horizontal axis shows the time. The blue line illustrates the movement trajectory of the real robot while the orange line shows the simulated trajectory.

requires the highest physical demand. Grasping the rock has shown to have the lowest load index since the shape of the rock was coincidentally easy to grasp. Also building the tower has a lower score which can be due to the order effect: users are already accustomed to use the system.

The order effect has not been compensated for in this study since this preliminary study was part of a bigger study and the order effect was applied to other control variables. The other reason that might have made the tower easier was the different concept of the task which was more interesting for participants. The process of building a tower involved grasping a box and then placing it on another two boxes. The user should have a focus on where they release the object since if they release the object in a wrong way the tower gets destructed. Therefore, as verbal feedback, they enjoyed building the tower more than the other three tasks. For centric grasp, since the object slips inside the hand, the number of meshes in the contact area increases greatly, making the simulation unstable and laggy.

Based on the final unweighted NASA task load index the centric grasp has the highest index and parallel grasp, building the tower and, the arbitrary grasp has lower indices correspondingly.

The results in figure 5.13 shows the NASA-TLX average index over all tasks. It shows the system requires low frustration and physical demand and as expected high mental demand. The final unweighted index is fairly low (27) that shows the system has a low task load. It was verbally confirmed the main reason for high mental demand was due to the small Field of View (FoV) in the Hololens. The Hololens one provides a horizontal FoV of 30° and a vertical of 17.5° using the basic Pythagorean theorem. Therefore, the FoV is quite small and does not show the whole scene in one view, so, the user has to look around to be able to finish a task.

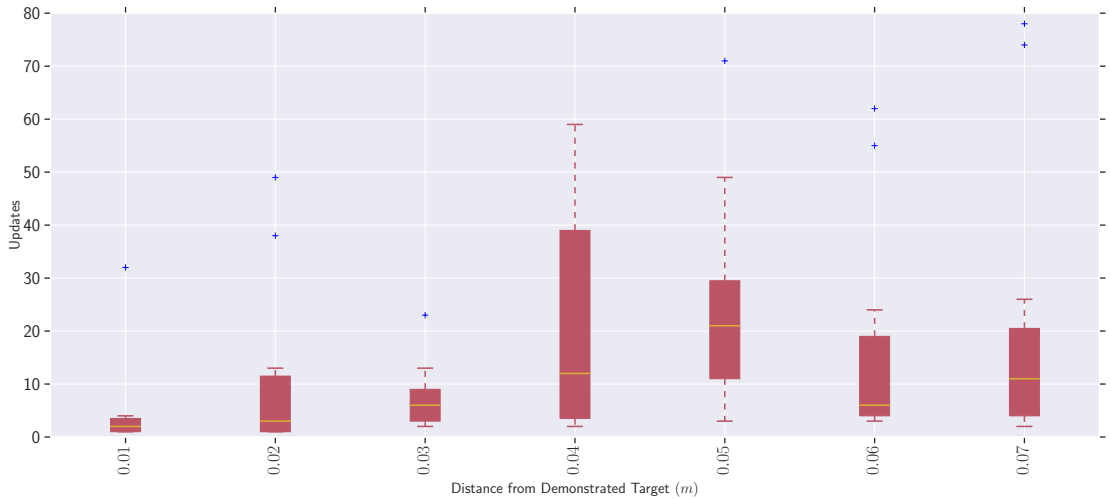


Figure 5.9: The figure shows the number of necessary updates for PI^2 shape and goal learning under uncertainty. The vertical axis determines the number of updates and the horizontal axis determines the magnitude of uncertainty. The magnitude increases to seven centimeters from the initial object position. The result shows an increase in the number of updates due to increasing the uncertainty, but an anomaly has happened after five trials and the number of total updates decreases which can be attributed to the exploration rate.

But, the high demand for the effort was mostly caused by the excess inertia in the LWR hardware due to friction, and joint problems. The other reason that was verbally confirmed is the small workspace of the fingers which makes the interface's finger get detached from the operator's finger during the task demonstration. There is another reason that was observed during the execution, it was a small delay in force feedback when the simulation had processing overhead and could not process in high frame-rate anymore.

The sharp surfaces and object meshes were bouncy because when the LWR and Exodex real-time computers get the forces the meshes are already colliding, therefore, the user receives a higher than expected force feedback. After the experiments, three of the participants actively felt the weight of the object after the grasp which was expected due to high inertia in the system and the low weight of the objects. Since the idea of the diaphragm was a novel idea and we could not find anything similar, there are no results that show the performance of diaphragm in comparison with other similar approaches. The system was also checked without the diaphragm and due to the complexity that the lack of it causes we do not report the results.

After the preliminary user-study, the results show the stability of the system allows for conducting a large scale user-study which covers the HRI aspects and it is beyond the scope of this thesis. The follow-up user-study should have a larger and more balanced pool of participants, e.g. better gender balance.

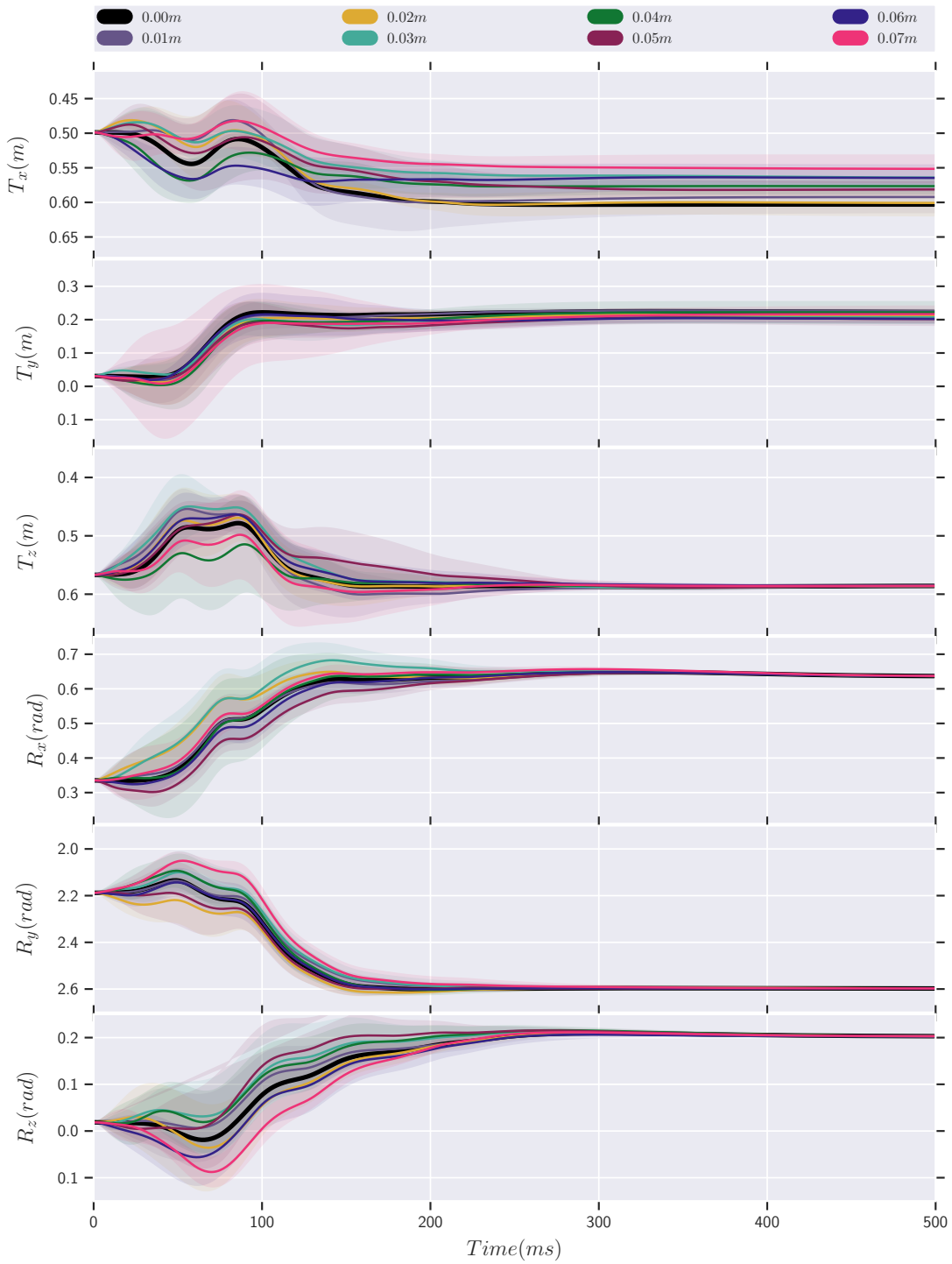


Figure 5.10: The figure shows the trajectories generated under different uncertainty in object position. The vertical axis shows the translation and rotation of the end-effector pose and the horizontal axis determines the time.

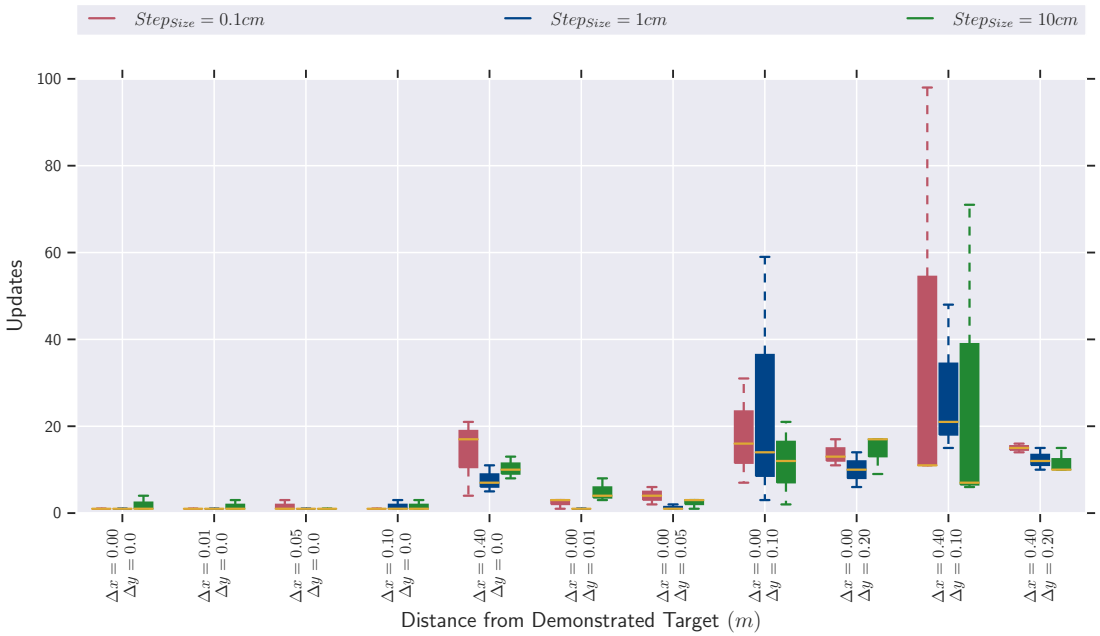


Figure 5.11: The figure compares different step sizes in the eNAC algorithm. The figure shows step size 10 has the best performance and requires fewer updates. The vertical axis determines the number of updates and the horizontal axis determines the magnitude of change in the object position relative to the demonstration.

The figure shows the system usability score, which based on the standard adjective ratings falls into the "Good" category. The separate responses to the questionnaire show the participants tend to use the system frequently (Q1). Moreover, the system has low complexity and easy to use (Q3). But the participants believe they need the support of a technical person to be able to use the system (Q4). The participants rate the integration of different functions high (Q5) with low inconsistency in the system (Q6). furthermore, the learnability of the system is rated high (Q7) with low prior-knowledge (Q10) where people felt confident to use it (Q9). The participants rate the cumbersomeness of the system low (Q8).

5.4 Summary Discussion

DMPs can cope with challenging problems such as the ball-in-cup game, obstacle avoidance, and grasping [41, 52, 42], however, in this thesis the complexity and challenge of the problems stem from the integration of DMPs into a teleoperation framework. Furthermore, Since uncertainty in many cases results from the use of physical hardware/slave devices, it is essential to evaluate the approach under these exact conditions. The results from other researches [76] already show the method performs well in a real uncertain environment. But, this thesis evaluates the result in a new hardware setup, simulation set, and different conditions. The

main reason that the learning on the real robot was not realized was the time limit due to the relatively high number of updates and roll-outs required for a successful grasp. This thesis aims to evaluate the system in a simulation environment and assess the ability to use sim-to-real to speed up the process.
Furthermore, the algorithms and approaches that have been used in this study were all policy search methods that explore the state-space locally. Therefore, it is very likely that the generated solutions are around the local minimum. The deep RL approaches use a global exploration method which can improve the generality of the final solution [6, 26]. However, deep RL uses neural networks as policy and value functions which requires a lot of training data. Therefore, if the slave robot spends a long time gathering data, the teleoperation may fail or the inconsistency between demonstration and the environment may increase.
Augmented Reality (AR) is used instead of Virtual Reality (VR) due to the assumption that the master device is complicated and the operator needs to be aware of the master device visually while performing the tasks. This assumption has to be investigated in practice to ensure that AR is better than VR in a telemanipulation scenario. The preliminary user study includes a second sub-study that investigates the use of a flat-screen instead of the AR goggles. In that study, the AR has got a higher score in SUS and also less Task Load index which proves the AR is a better solution in comparison to a traditional flat screen.

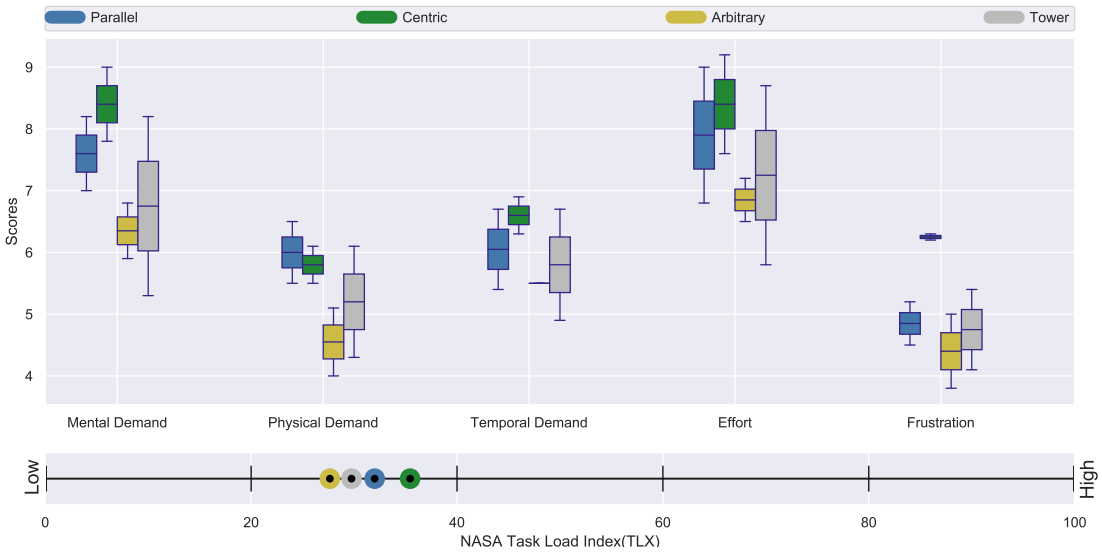


Figure 5.12: The figure shows the result of the NASA-TLX questionnaire comparing the grasp of different objects, The figure on the bottom shows the unweighted task load index. The vertical axis determines the score and the horizontal axis shows the workload measurements. The bottom figure compares different tasks and the axis determines the unweighted NASA-TLX index.

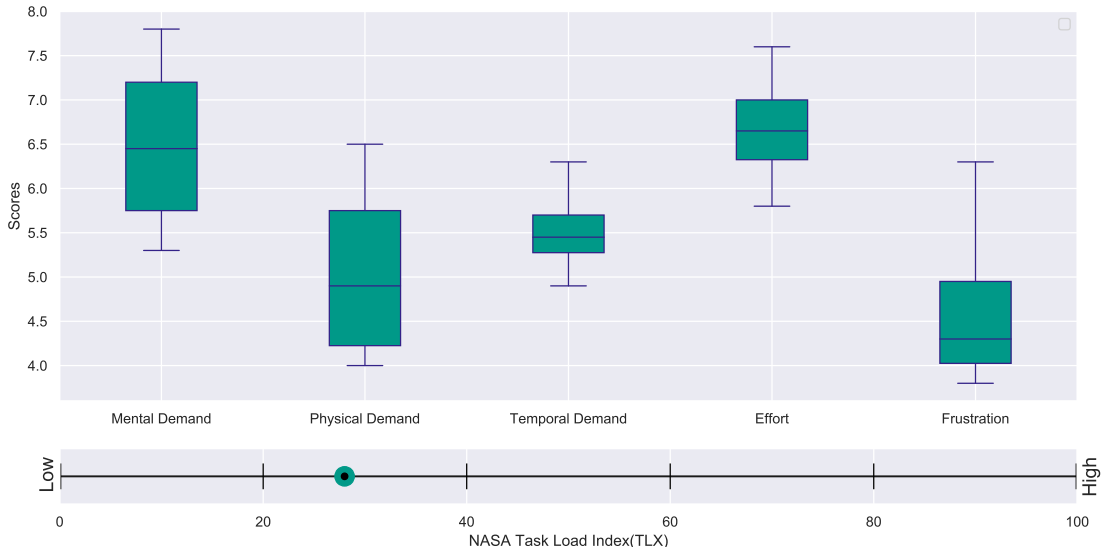


Figure 5.13: The figure shows the result of the NASA-TLX questionnaire Average over all objects. The vertical axis determines the score and the horizontal axis shows the workload measurements. The bottom figure illustrates the average unweighted NASA-TLX index.

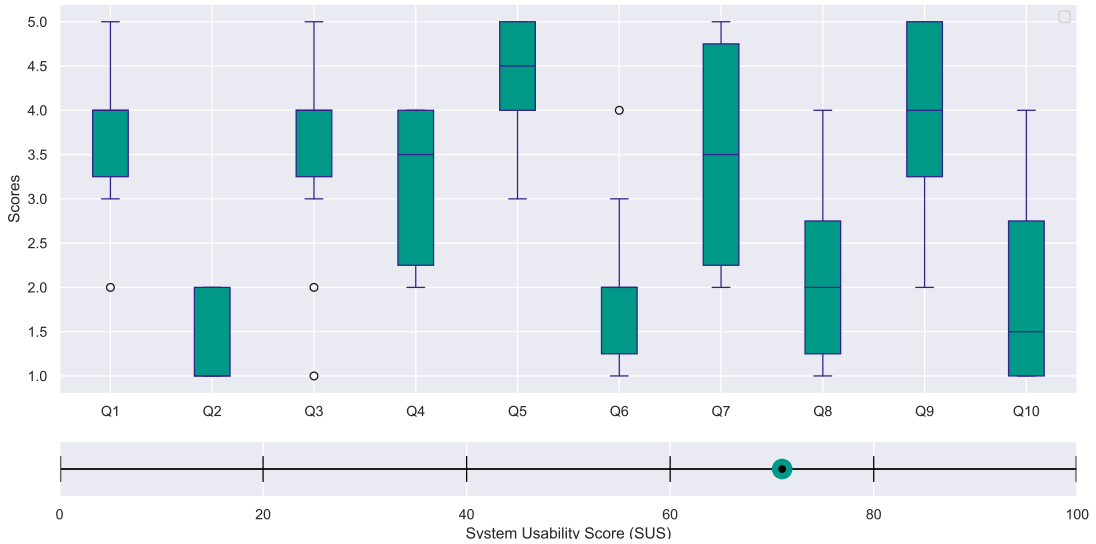


Figure 5.14: The figure shows the result of the system usability score questionnaire averaged over all tasks. The vertical axis determines the score and the horizontal axis shows the questions (Can be found in 4.5.3). The figure on the bottom shows the SUS score which falls into "Good" in adjective ratings.

Chapter 6

Conclusion

The DMPs are powerful tools to adapt an old trajectory to new conditions with low changes in the target position. The used RL methods can compensate for the DMP failure in the case of large changes in the object position. Therefore two layers of DMP and RL can ensure the success of task completion in an MMT scenario. The human error and uncertainty in the object position can be compensated using the RL since DMPs fail to adapt to unknown conditions.
This thesis aims at the design, implementation, and evaluation of a model mediated teleoperation pipeline. The proposed architecture uses DMPs and RL to function in the existence of long time delays. The system assumes the environment is time-variant with high uncertainty. The mediated model in the architecture is a fully simulated environment providing instant haptic feedback for the operator as well as an augmented reality visualization. The simulated environment resembles the real remote environment and its physical properties.
The DMPs have been used for an initial adaptation to new conditions to compensate for the model mismatch. To increase the performance of the system dealing with the uncertainty and inaccuracy of the model a layer of RL methods have been evaluated. The learning process is distributed over seven different simulators which also increase the speed while it assures the system's safety.
The novel haptic interface Exodex has been used for the first time in a teleoperation scenario as the master device to provide haptic feedback with high accuracy. The slave device resembles the right arm of a Justin robot designed by DLR combining a five-finger hand (FFH) attached to a custom configured lightweight robot (LWR 4+). Besides, an augmented reality visualization via Microsoft Hololens which fuses the slave device and virtual environment models was provided to increase the performance of the operator.
The teleoperation scenario was separated into two different parts, the first part is the process of performing tasks by the operator on the master device in the virtual environment. Since this part is closely related to human-robot interaction, the system is evaluated using a small user study. The second part related to the learning and adaptation was evaluated separately for two different challenges. First, the object was misplaced the old trajectory had to be adapted to new conditions and the second target position was given with uncertainty, In both, the DMP transfor-

mation and RL agent had to adapt and learn to successfully deploy the grasp in a short time.
Our results show the unweighted NASA task load index of the system is low while the usability score of the system is above average (higher than 60% of all products tested). The learning result shows the deployed RL methods were able to successfully find a solution for all tasks in a limited time. The results also show DMPs were able to adapt to new conditions where no uncertainty or work-space limitation exists.

6.1 Future work

We believe that apart from looking for learning from demonstration, future research should look for different methods such as learning from imitation and from videos to facilitate the learning process. Regardless, future research could continue to explore the online Deep Reinforcement Learning (DLR) methods such as Deep Deterministic Policy Gradient [39] and batch algorithms such as Trust Region Policy Optimization (TRPO) [70]. Investigating the effect of using a virtual reality representation might prove important. Future research should further develop and confirm these initial findings by using a real slave setup. The inverse kinematics for human hand pose estimation can be replaced by a Kinfinity glove [1] or Cyberglove [2]. Future studies should aim to replicate results in a larger user-study. The system can be modified to be used out of the teleoperation setup where a single robot observes the actions and tries to mimic them using DMPs and RL. Looking forward, further attempts could prove quite beneficial to the literature.

6.2 Disclaimer

This study does not introduce new RL and DMP approaches. All the algorithms have been borrowed from different studies. The base of the code is borrowed from the main Matlab repository from Stefan Schaal's lab and it was reimplemented using Python 2.7. The repository will be accessible on GitHub[3]. The Simulink models related to LWR robot control and dynamics were borrowed from the DLR HUG project. The Simulink robot dynamics block of the slave robot arm was borrowed from the Justin robot model. The impedance controller for teleoperation was already implemented in the DLR institute for robotics and mechatronics (RMC). All the Simulink models are for internal use and confidentially stored on internal DLR GitHub servers.

[1] `http://kinfinity-solutions.com/` (Visited on: 31/12/2019)
[2] `http://www.cyberglovesystems.com/` (Visited on: 31/12/2019)
[3] `https://github.com/hadibeikm/PolicySearch.git`(Visited on: 31/12/2019)

Bibliography

[1] Vuforia, https://www.ptc.com/en/products/augmented-reality/vuforia, last check: 2019-12-29.

[2] A. Achhammer, C. Weber, A. Peer, and M. Buss. Improvement of model-mediated teleoperation using a new hybrid environment estimation technique. In *2010 IEEE International Conference on Robotics and Automation*, pages 5358–5363, May 2010.

[3] A. Albu-Schaffer and G. Hirzinger. Cartesian impedance control techniques for torque controlled light-weight robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 1, pages 657–663 vol.1, May 2002.

[4] R. J. Anderson and M. W. Spong. Bilateral control of teleoperators with time delay. *IEEE Transactions on Automatic Control*, 34(5):494–501, May 1989.

[5] Ribin Balachandran, Mikael Jorda, Jordi Artigas Esclusa, J.H. Ryu, and Oussama Khatib. Passivity-based stability in explicit force control of robots. In *IEEE International Conference on Robotics and Automation ICRA*, June 2017.

[6] Hadi Beik Mohammadi, Mohammad Ali Zamani, Matthias Kerzel, and Stefan Wermter. Mixed-Reality Deep Reinforcement Learning for a Reach-to-grasp Task. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11727 LNCS, pages 611–623. Springer Verlag, 2019.

[7] A. K. Bejczy, W. S. Kim, and S. C. Venema. The phantom robot: predictive displays for teleoperation with time delay. In *Proceedings., IEEE International Conference on Robotics and Automation*, pages 546–551 vol.1, May 1990.

[8] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

[9] Mark Brudnak. Predictive displays for high latency teleoperation. 08 2016.

[10] Ricardo Camarero, Thomas Hulin, and Bernhard Vodermayer. The stamas simulator: A kinematics and dynamics simulator for an astronaut's leg and hand exoskeleton. 10 2015.

[11] Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Found. Trends Robot*, 2(1–2):1–142, August 2013.

[12] Emma Delgado, Pablo Falcon, Miguel Diaz-Cacho, and Antonio Barreiro. Four-channel teleoperation with time-varying delays and disturbance observers. *Mathematical Problems in Engineering*, 2015:1–11, 08 2015.

[13] M. Freese E. Rohmer, S. P. N. Singh. Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013. www.coppeliarobotics.com.

[14] Max Fischer, Patrick van der Smagt, and Gerd Hirzinger. Learning techniques in a dataglove based telemanipulation system for the dlr hand. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 2, pages 1603–1608. IEEE, 1998.

[15] A. Forouzantabar, H. Talebi, and Ali Sedigh. Adaptive neural network control of bilateral teleoperation with time delay. *Nonlinear Dynamics*, 67:1123–1134, 05 2012.

[16] Antonio Frisoli, Edoardo Sotgiu, Damaso Checcacci, Francesco Simoncini, Simone Marcheschi, Carlo Alberto Avizzano, and Massimo Bergamasco. Theoretical and experimental evaluation of a 2-channel bilateral force reflection teleoperation system. 2004.

[17] M. B. Hafez, C. Weber, and S. Wermter. Curiosity-driven exploration enhances motor skills of continuous actor-critic learner. In *2017 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pages 39–46, Sep. 2017.

[18] Muhammad Burhan Hafez, Cornelius Weber, Matthias Kerzel, and Stefan Wermter. Deep Intrinsically Motivated Continuous Actor-Critic for Efficient Robotic Visuomotor Skill Learning. oct 2018.

[19] B. Hannaford and Jee-Hwan Ryu. Time-domain passivity control of haptic interfaces. *IEEE Transactions on Robotics and Automation*, 18(1):1–10, Feb 2002.

[20] Sandra G. Hart. Nasa-Task Load Index (NASA-TLX); 20 Years Later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 50(9):904–908, 2006.

[21] Changchun Hua and Yana Yang. Neural network-based adaptive position tracking control for bilateral teleoperation under constant time delay. *Neurocomputing*, 113:204–212, 08 2013.

[22] Thomas Hulin, Katharina Hertkorn, Philipp Kremer, Carsten Preusche, Simon Schätzle, Jordi Artigas, Mikel Sagardia, and Franziska Zacharias. The dlr bimanual haptic device with optimized workspace. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3441–3442, May 2011.

[23] Jin-Quan Huang and F. L. Lewis. Neural-network predictive control for nonlinear dynamic systems with time-delay. *IEEE Transactions on Neural Networks*, 14(2):377–389, March 2003.

[24] Jorge Juan and Emilio J. Sánchez. Control algorithms for haptic interaction and modifying the dynamical behavior of the interface. 2005.

[25] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[26] Matthias Kerzel, Hadi Beik Mohammadi, Mohammad Ali Zamani, and Stefan Wermter. Accelerating Deep Continuous Reinforcement Learning through Task Simplification. Technical report.

[27] Matthias Kerzel and Stefan Wermter. Learning of neurobotic visuomotor abilities based on interactions with the environment, Nov 2017.

[28] Jens Kober and Jan Peters. Policy search for motor primitives in robotics. *Machine Learning*, 84(1):171–203, Jul 2011.

[29] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3, Sep. 2004.

[30] O. Kroemer, C. Daniel, G. Neumann, H. van Hoof, and J. Peters. Towards learning hierarchical skills for multi-phase manipulation tasks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1503–1510, May 2015.

[31] Sergey Levine and Vladlen Koltun. Learning complex neural network policies with trajectory optimization. In *International Conference on Machine Learning*, pages 829–837, 2014.

[32] Neal Y. Lii, Zhaopeng Chen, benedikt Pleintinger, Christoph H. Borst, Gerd hirzinger, and Andre Schiele. Toward understanding the effects of visual- and force- feedback on robotic hand grasping performance for space teleoperation. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3745–3752, October 2010.

[33] Neal Y. Lii, Zhaopeng Chen, Máximo A. Roa, Annika Maier, Benedikt Pleintinger, and Christoph Borst. Toward a task space framework for gesture commanded telemanipulation. *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, pages 925–932, 2012.

[34] Neal Y. Lii, Daniel Leidner, André Schiele, Peter Birkenkampf, Ralph Bayer, Benedikt Pleintinger, Andreas Meissner, and Andreas Balzer. Simulating an extraterrestrial environment for robotic space exploration: The meteron supvis-justin telerobotic experiment and the solex proving ground. In *13th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*, May 2015.

[35] Neal Y. Lii and Miguel Neves. Eingabesystem, de102017220990, May 2019.

[36] Neal Yi-Sheng Lii, Andreas Balzer, Georg Stillfried, Zhaopeng Chen, Benedikt Pleintinger, and Markus Grebenstein. Handexoskeleton and robotic arm with such a hand exoskeleton, de102017220936a1, November 2017.

[37] Neal Yi-Sheng Lii, Georg Stillfried, Zhaopeng Chen, Maxime Chalon, Benedikt Pleintinger, and Annika Maier. exoskeleton, de102017220996a1, November 2017.

[38] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2015.

[39] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

[40] Chao Liu, Jing Guo, and Philippe Poignet. Nonlinear model-mediated teleoperation for surgical applications under time variant communication delay. *IFAC-PapersOnLine*, 51(22):493 – 499, 2018. 12th IFAC Symposium on Robot Control SYROCO 2018.

[41] Jens Lundell. Dynamic movement primitives and reinforcement learning for adapting a learned skill. Master's thesis, Luleå University of Technology, Department of computer science, electrical and space engineering, 2016.

[42] Takamitsu Matsubara, Sang-Ho Hyon, and Jun Morimoto. Learning parametric dynamic movement primitives from multiple demonstrations. *Neural Networks*, 24(5):493 – 500, 2011.

[43] Probal Mitra. Model-mediated telemanipulation model-mediated telemanipulation. 2007.

[44] F. Mobasser and K. Hashtrudi-Zaad. Stable impedance reflecting teleoperation with online collision prediction. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 476–482, Oct 2007.

[45] Hadi Beik Mohammadi, Michael Görner, Stefan Wermter, Matthias Kerzel, Hadi Beik-Mohammadi, Mohammad Ali Zamani, and Manfred Eppe. Neural End-to-End Learning of Reach for Grasp Ability with a 6-DoF Robot Arm. Technical report, 2018.

[46] Hadi Beik Mohammadi, Nikoletta Xirakia, Fares Abawi, Irina Barykina, Krishnan Chandran, Gitanjali Nair, Cuong Nguyen, Daniel Speck, Tayfun Alpay, Sascha Griffiths, Stefan Heinrich, Erik Strahl, Cornelius Weber, and Stefan Wermter. Designing a Personality-Driven Robot for a Human-Robot Interaction Scenario. Technical report.

[47] Katharina Mülling, Jens Kober, Oliver Kroemer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279, 2013.

[48] G. Niemeyer and J. . E. Slotine. Stable adaptive teleoperation. *IEEE Journal of Oceanic Engineering*, 16(1):152–162, Jan 1991.

[49] Emmanuel Nuño, Luis Basañez, and Romeo Ortega. Passivity-based control for bilateral teleoperation: A tutorial. *Automatica*, 47(3):485 – 495, 2011.

[50] Michael Ortega-Binderberger, Stéphane Redon, and Sabine Coquillart. A six degree-of-freedom god-object method for haptic display of rigid bodies. *IEEE Virtual Reality Conference (VR 2006)*, pages 191–198, 2006.

[51] M. Panzirsch, H. Singh, M. Stelzer, M. J. Schuster, C. Ott, and M. Ferre. Extended predictive model-mediated teleoperation of mobile robots through multilateral control. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1723–1730, June 2018.

[52] D. Park, H. Hoffmann, P. Pastor, and S. Schaal. Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*, pages 91–98, Dec 2008.

[53] C. Passenberg, A. Peer, and M. Buss. Model-mediated teleoperation for multi-operator multi-robot systems. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4263–4268, Oct 2010.

[54] L. S. D. Pecly, M. L. O. Souza, and K. Hashtrudi-Zaad. Model-reference model-mediated control for time-delayed teleoperation systems. In *2018 IEEE Haptics Symposium (HAPTICS)*, pages 72–77, March 2018.

[55] Luis F Penin and Kotaro Matsumoto. Teleoperation with time delay: A survey and its use in space robotics. Technical report, National Aerospace Laboratory (NAL), 2002.

[56] A. Pereira, G. Stillfried, T. Baker, A. Schmidt, A. Maier, B. Pleintinger, Z. Chen, T. Hulin, and N. Y. Lii. Reconstructing human hand pose and configuration using a fixed-base exoskeleton. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3514–3520, May 2019.

[57] J. Peters and S. Schaal. Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2219–2225, Oct 2006.

[58] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, May 2008.

[59] Vesna Rankovic, Jasna Radulovic, Nenad Grujovic, and Dejan Divac. Neural network model predictive control of nonlinear systems using genetic algorithms. *International Journal of Computers Communications and Control*, 7(3):540–549, 2014.

[60] Florian Richter, Yifei Zhang, Yuheng Zhi, Ryan K. Orosco, and Michael C. Yip. Augmented reality predictive displays to help mitigate the effects of delayed telesurgery. In *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*, pages 444–450, 2019.

[61] Thomas Rückstieß, Frank Sehnke, Tom Schaul, Daan Wierstra, Yi Sun, and Jürgen Schmidhuber. Exploring parameter space in reinforcement learning. *Paladyn, Journal of Behavioral Robotics*, 1:14–24, 03 2010.

[62] Andrew Sanders. *An Introduction to Unreal Engine 4*. A. K. Peters, Ltd., Natick, MA, USA, 2016.

[63] S. Schaal and C. G. Atkeson. Robot juggling: implementation of memory-based learning. *IEEE Control Systems Magazine*, 14(1):57–71, Feb 1994.

[64] Stefan Schaal. *Dynamic Movement Primitives -A Framework for Motor Control in Humans and Humanoid Robotics*, pages 261–280. Springer Tokyo, Tokyo, 2006.

[65] Stefan Schaal and Christopher G. Atkeson. Assessing the quality of learned local models. In *NIPS*, 1993.

[66] Peter Schmaus, Daniel Leidner, Ralph Bayer, Benedikt Pleintinger, Thomas Krüger, and Neal Y. Lii. Continued advances in supervised autonomy user interface design for meteron supvis justin. In *2019 IEEE Aerospace Conference*. IEEE Computer Society, March 2019.

[67] Peter Schmaus, Daniel Leidner, Thomas Krüger, Ralph Bayer, Benedikt Pleintinger, André Schiele, and Neal Y. Lii. Knowledge driven orbit-to-ground teleoperation of a robot coworker. *IEEE Robotics and Automation Letters*, 5(1):143–150, October 2019.

[68] Peter Schmaus, Daniel Leidner, André Schiele, Benedikt Pleintinger, Ralph Bayer, and Neal Y. Lii. Preliminary insights from the meteron supvis justin space-robotics experiment. *IEEE Robotics and Automation Letters*, 3(4):3836–3843, July 2018.

[69] Annika Schmidt. Viscosity perception of virtual fluids rendered by a hand exoskeleton. Master's thesis, Delft University of Technology, Faculty of Mechanical, Maritime and Materials Engineering, 2018.

[70] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015. PMLR.

[71] A. C. Smith and K. Hashtrudi-Zaad. Adaptive teleoperation using neural network-based predictive control. In *Proceedings of 2005 IEEE Conference on Control Applications, 2005. CCA 2005.*, pages 1269–1274, Aug 2005.

[72] Ora Smith. Closer control of loops with dead time. 1957.

[73] Georg Stillfried, Ulrich Hillenbrand, Marcus Settles, and Patrick van der Smagt. *MRI-Based Skeletal Hand Movement Model*, pages 49–75. Springer International Publishing, Cham, 2014.

[74] Georg Stillfried and Patrick van der Smagt. Movement model of a human hand based on magnetic resonance imaging (mri). 10 2010.

[75] Freek Stulp and Olivier Sigaud. Robot Skill Learning: From Reinforcement Learning to Evolution Strategies. *Paladyn*, 4(1):49–61, 2013.

[76] Freek Stulp, Evangelos Theodorou, and Stefan Schaal. Reinforcement learning with sequences of motion primitives for robust manipulation. *Robotics, IEEE Transactions on*, 28:1360–1370, 12 2012.

[77] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[78] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *J. Mach. Learn. Res.*, 11:3137–3181, December 2010.

[79] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. Learning policy improvements with path integrals. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 828–835, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[80] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, Oct 2012.

[81] Unity Technologies, https://unity.com, Last check: 2019-06-10. Unity.

[82] K. Warwick, editor. *Industrial Digital Control Systems*. Control, Robotics amp; Sensors. Institution of Engineering and Technology, 1988.

[83] Wikipedia contributors. Impedance control — Wikipedia, the free encyclopedia, 2019. [Online; accessed 8-November-2019].

[84] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992.

[85] Xiao Xu, Burak Cizmeci, Clemens Schuwerk, and Eckehard G. Steinbach. Model-mediated teleoperation: Toward stable and transparent teleoperation systems. *IEEE Access*, 4:425–449, 2016.

Erklärung der Urheberschaft

Ich versichere an Eides statt, dass ich die Master thesis im Studiengang Intelligent Adaptive Systems selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ort, Datum Unterschrift

Erklärung zur Veröffentlichung

Ich erkläre mein Einverständnis mit der Einstellung dieser Master thesis in den Bestand der Bibliothek.

Ort, Datum Unterschrift