NOVEL MAPPING ALGORITHM FOR BILATERAL TELEOPERATION OF BIPEDAL ROBOT USING A MANIPULATOR

by

Koceila Cherfouh

Submitted in partial fulfillment of the requirements
for the degree of Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
April 2022

Table of Contents

List of Tables

List of Figures

Abstract

Research on the Bilateral teleoperation of biped robots is limited. So far, researchers have only studied bilateral teleoperation of bipedal robots using exoskeletons due to the kinematic similarities. This is, however, not practical as it requires considerable resources for the purchase, maintenance, and operator safety training.

This thesis presents a novel mapping algorithm for bilateral teleoperation of a bipedal robot using a robotic manipulator. This research aims to control the gait of a biped robot and receive haptic feedback representative of its movements along its center of mass (CM). Two mapping methods are presented and compared: 1- a cartesian-based mapping that relies on forward and inverse kinematics to map desired trajectories; 2- a deep-learning approach that reduces complexity and computation time. Simulation and experimental results on a 10-degrees of freedom (DoF) biped robot and 3-DoF robotic manipulator are carried out to validate the feasibility of this proposal.

List of Abbreviations and Symbols Used

CM Center of mass

CNN Convolutional neural network

DL Deep learning

DoF Degrees of freedom

FK Forward kinematics
FL Feedback linearization

FMA Forward mapping algorithm

GA Genetic algorithm

HRI Human-robot interaction

IK Inverse kinematics
IMA Inverse mapping algorithm

LSTM Long short-term memory

MASD Maximum Allowable stepping distance

MSSD Maximum situational stepping distance

NDO Nonliinear disturbance observer

PID Proportional, integral and derivative

RBFNN Radial basis function neural network

SLIP Spring-loaded inverted pendulum

SMC Sliding-mode controller

SOSMC Second-order sliding-mode controller

SSP Single support phase

${}_{i}^{0}T$ – Transformation matrix from coordinate frame 0 to i

x_i – Coordinate position in the X-axis of the i^th joint

y_i – Coordinate position in the Y-axis of the i^th joint

z_i – Coordinate position in the Z-axis of the i^th joint

a_i – Link length. Measured from frame Z_{i-1} to Z_i along X_{i-1}

α_i – Link twist. Angle from frame Z_{i-1} to Z_i measured about X_{i-1}

d_i – Link offset. Distance from X_{i-1} to X_i measured along Z_{i-1}

θ_i – Joint angle. Angle from X_{i-1} to X_i measured about Z_i

r, θ_1, θ_2 – Updated states of the 3D SLIP model

$\phi_1, \phi_2..., \phi_{10}$ – Biped desired trajectories. Derived from inverse kinematics

$\phi_{1md}, \phi_{2md}, \phi_{3md}$ – Manipulator desired trajectories. Derived from inverse kinematics. These variable generate the haptic feedback

l_{1m}, l_{2m} – link 1 and 2 of the manipulator

x_{cm}, y_{cm}, z_{cm} – Scaled center of mass position of the biped. This value is used for input to the IK to generate haptic feedback

β_1, β_2 – Variables required to compute the IK

L – Lagrange equation

KE – Kinetic energy

PE – Potential energy

M – Inertia matrix

C – Coriolis vector

τ_i – Torque applied at joint i

τ_h – Human operator torque vector. Only applicable for the master robot (manipulator)

δ – Stochastic noise and external disturbances

f – Reformulated dynamics vector. $f = M^{-1}(-C - G)$

g – Reformulated dynamics matrix. $g = M^{-1}$

$r_s, \theta_{1s}, \theta_{2s}$ – Measured states of 3D SLIP Model

M_j, v_j, I_j, ω_j – Dynamic properties of the j^{th} link. Mass, velocity, inertia and angular velocity, respectively.

B – Inertia Matrix of the 3D- SLIP model

F – Reformulated dynamics vector of the 3D- SLIP model

s – Sliding surface of the SOSMC

λ – Scalar used to define the sliding surface

V – Lyapunov function

X_{se} – Error vector of the 3D SLIP model

K_1 – Gain of the 3D SLIP FL controller

$\hat{\tau}_d, \hat{\delta}_d$ – Estimated disturbance by the NDO

z – Auxialary variable vector required to compute disturbance estimate

L – Disturbance observer gain matrix

K_d – Disturbance observer gain

S_1 – Input to the feedforward RBFNN compensator

H_n – Gaussian Kerna for RBFNN compensator

W_{RBF} – Adaptive weights of the RBFNN compensator

β_{RBF} – Positive Constant

θ_e – State error vector

τ_{total} – Total control input vector

$z_{desired}$ – Desired Z-position for the walking gait algorithm of the biped

x_{master}, x_{EE} – End effector position of the master manipulator

α_{max} – Threshold value between switching from forward, stop, backward motion

$x_{hip}, y_{hip}, z_{hip}$ – Coordinate position of the biped's hip joint (also center of mass position)

$AMSD$ – Absolute maximum stepping distance

θ_{max} – Maximum leaning angle of the biped in the Z-axis before losing stability

d_0 – Euclidean distance between the ankle and the hip joint of the supporting leg before taking a step

γ_0 – Angle of the supporting leg before taking a step

$_N$ – This subscript denotes the time step at which a step is completed or when a change in direction occurred

$MSSD_+$ – Maximum situational stepping distance in the positive Z-direction

$MSSD_-$ – Maximum situational stepping distance in the negative Z-direction

z_4 – Cartesian position of the hip joint

DP – Desired hip position

$_{Ns}$ – This subscript denotes the time step at which a step is completed

r_d – Desired link length of the 3D SLIP model

y_{cm-d} – Desired trajectory of the hip joint in the Y-axis. This generates the left-to-right movement when walking

σ_l – Lateral standard deviation. Value used to compute $y_cm - d$

z_{HIP-i} – Coordinate of the hip join along the Z-axis at time step i

X_{HIP-i} – Coordinate of the hip join along the X-axis at time step i

z_{SF-i} – Coordinate of the supporting foot along the Z-axis at time step i

X_{SF-i} – Coordinate of the supporting foot along the X-axis at time step i

Θ_{sd} – SLIP model desired trajectory vector

$direction$ – Direction of the biped's gait along the Z-axis

f_t – LSTM forget gate

σ – Sigmoid function

x_T – LSTM input

h_t – LSTM hidden state

W_0 – Weights

b_0 – Bias

i_t – LSTM input layer at time t

z_t – LSTM tanh layer that decides which information to store

C_t – LSTM Cell state

o_t – LSTM output state

O_{ReLU} – Output of the rectified linear unit layer

i_{ReLU} – Input to the rectified linear unit layer

O_{cnn} – Output of cross-correlation operation in CNN

I_{cnn} – Input to the cross-correlation operation in CNN

F_{cnn} – Filter of CNN kernel of fixed length

O_{FC} – Output of the fully connected layer

I_{FC} – Input to the fully connected layer

m_t, v_t – Exponentially decaying average first and second moments of the Adam optimizer

ζ_1, ζ_2 – Decay rates of the Adam optimizer

$\hat{m}_t, \hat{v}_t$ – Updated moments

$\tilde{M}, \tilde{C}, \tilde{G}$ – Error in dynamics parameters between the estimated and true parameters

$\hat{M}, \hat{C}, \hat{G}$ – Estimated dynamics parameters

U – System total uncertainty and disturbacne

$PRRP$ – Motor shaft's position relative to its resting position

RAP – Relative angular position of the motor

RP – Resting Position

Acknowledgements

This work has resulted from continued support from numerous colleagues, friends, and family members. First, I thank Dr. Jason Gu for the support, advice, and opportunities he offered through my master's. Dr. Jason Gu supported me in many aspects and motivated me to learn and experiment with new cutting-edge technologies through research publications and internship opportunities. I would also like to thank my graduate committee supervisor, Dr. Kamal ElSankary, and Dr. Mae Seto for their valuable suggestions. Next, I would like to thank my lab partners from the Robotics Lab for Biomedical, Rehabilitation, and Assistive Technologies, including Usman Asad and Dr. Umar Farooq. Dr. Umar guided and introduced me to research publications and conferences and helped me publish numerous papers. I would also like to thank my professors from the University of New Brunswick, namely Dr. Maryhelen Stevenson, Dr. Brent Petersen, Dr. Howard Li, and professor Roy Lavigne for supporting me to start my master's. Finally, I want to thank my parents and sister for their support throughout this journey. Thank you.

Chapter 1

Introduction

Chapter 1 outlines an introduction of bilateral teleoperation systems and the research objectives of this thesis. More specifically, it includes the motivation of this study, an in-depth literature review discussing the history, and a discussion of the recent work conducted on bilateral teleoperation of biped robots. Finally, we discuss our contribution and present the outline of the thesis.

1.1 Thesis Motivation

As artificial intelligence is increasingly becoming popular amongst researchers for various applications due to its adaptability and ease of use, the design of autonomous robotic systems remains a challenging problem due to its large nonlinearities when interacting with dynamic environments, thus requiring large amounts of training data to achieve good performance. Bilateral teleoperation is an alternative to autonomous systems. An operator can control a robot remotely to conduct tasks while simultaneously receiving haptic feedback representative of the controlled robot's experienced forces. This is typically achieved using a master-slave mapping algorithm that translates system trajectories between master and slave systems.

The simplest solution would be a 1-to-1 mapping algorithm, where both master and slave systems are symmetric. An example of such a system is a pendulum model for both master and slave systems, where one pendulum's trajectory depends on the position of the other. However, for most real-world applications, systems are asymmetric, which complicates the design procedure. The most common asymmetries found in bilateral teleoperation include kinematic asymmetries between master and slave, environmental asymmetries, or communication time delay asymmetries. The mapping algorithm must then be designed to account for the asymmetries in the system.

The most used mapping methods in bilateral teleoperation literature are joint-space mapping and workspace mapping. Joint space mapping assigns joint-to-joint desired trajectories but is unsuitable when the master and slave systems have significant kinematic asymmetries. On the other hand, the operating space mapping relies on forward and inverse kinematics of the master and slave systems to map a scaled Cartesian coordinate from one site to another. However, this method often restricts the operator from accessing the complete workspace of the slave robot. Other more refined mapping algorithms have been proposed, such as the machine learning approach or a combination of joint space and operating space mapping. As for biped robots, they have received little attention from researchers in the application of bilateral teleoperation due to their kinematic complexity.

So far, to minimize kinematic asymmetries and simplify the teleoperation of biped robots, researchers used exoskeletons as the remote combined with joint-to-joint space mapping. This simplifies the task because it removes the need to design a walking gait algorithm and compute inverse kinematics for master and slave systems, meaning that desired trajectories are directly fed into the joint state controllers. For this application, one can argue that using exoskeletons as the remote is intuitive as it allows direct haptic feedback on each joint of the biped. However, the purchase, maintenance, and operator training costs are far too expensive. In this work, an intuitive design aimed to reduce the purchase and maintenance costs by using a haptic manipulator - instead of an exoskeleton - as the remote for the biped is proposed.

This new and modified operating space-based method allows the user to control a biped robot using a manipulator while simultaneously feeling haptic feedback representing the left-to-right and up-and-down movements of the biped robot's center of mass (CM). For the forward mapping algorithm (FMA), this approach computes the forward kinematics of the manipulator's end effector and maps it to the desired position for the biped's CM. The 3D spring-loaded inverted pendulum (SLIP) gait is is chosen to control the robot to the desired position and drive the error to zero. The 3D SLIP was chosen because of its simplicity and accuracy in the model compared to the linear inverted pendulum, which does not model the spring force at the knee joint. The SLIP walking gait accepts the desired CM position and outputs desired joint trajectories to the joint state controllers using the biped's inverse kinematics. As

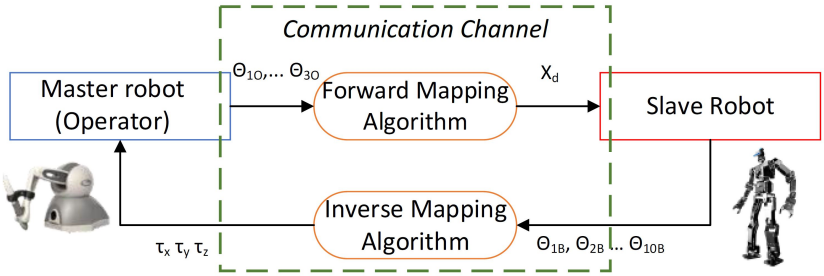


Figure 1.1: Bilateral teleoperation of biped using a manipulator

for the inverse mapping algorithm (IMA), the joint state positions of the biped robot are used to compute the CM point with forward kinematics. Then, a scaled trajectory of the CM is used as input to the manipulator's inverse kinematics. This generates joint trajectories for the manipulator that track the biped's CM point, which is the operator's haptic feedback. However, this method proven to work by numerous researchers is lengthy and prone to significant computational inefficiencies and a lack of robustness to new situations.

Additionally, deep learning approaches have rarely been applied to the bilateral teleoperation mapping field, as most currently existing work uses bilateral teleoperation for robot task learning. This form of Human-robot interaction (HRI) using deep learning from demonstration uses either recorded sensor data from robots, recorded human videos, or move and capture waypoints. Over the years, researchers have applied long-short term memory (LSTM) neural network-based structures for mapping human-to-robot interactions, given their ability to model order dependence in a sequence. Others improved the architecture and applied a combination of convolutional neural networks (CNN) and LSTMs to capture long-term and short-term dependencies. To test the performance of the proposed methods, a comparison between the CNN-LSTM architecture and the modified operating space-based method is carried out on a 3-DoF Omni manipulator and a 10-DoF bipedal robot through simulations and experimentation.

1.2 History of Bilateral Teleoperation and Bipedal Robotic Systems

1.2.1 History of Bilateral Teleoperation

Teleoperation of robotic systems was first introduced in 1949 by early pioneer researcher in robotics Raymon C. Goertz when he developed a mechanical master-slave system to manipulate radioactive materials [1]. A few years later, Goertz developed a bilateral master-slave system in which he laid out many of the fundamental concepts of bilateral teleoperation [2]. In the early to late 1960s, Ferrell and Sheridan studied the stability of asymmetric bilateral teleoperation systems and proposed supervisory control as a method to address communication time delays [3, 4]. The teleoperation field then shifted towards a control system-heavy research area in the mid-1980s to early 1990s. Many of these breakthroughs include the passivity-based control, the stability analysis through Lyapunov [5], and H_∞ control to address time delays. Now, the objective of many researchers is to develop a friendly and immersive virtual environment to make the operator feel as if they are at a remote location [6]. These advances include immersion by audio and visual feedback, thus allowing various asymmetric systems like space and underwater systems to be implemented.

1.2.2 History of Biped Robots

Humanoid robots interest researchers for many reasons, including their design well fit to interact with humans, their versatility in their movement, and the wide range of applications. Originally, biped robots were first conceived by Japanese researcher Ichiro Kato in 1967 and completed his first model in 1972 [7]. Of the many existing bipedal gait algorithms, Miomir Vukobratović, a pioneer in legged robots and Serbian mechanical engineer, developed the zero moment point concept currently widely used in the stability analysis of bipedal locomotion [8]. This concept inspired the development of a model for a human body gait using the inverted pendulum model [9, 10], the spring-loaded inverted pendulum (SLIP) [11, 12], and other complex models with numerous DoF. Over time, more sophisticated gait control algorithms emerged, such as energy-efficient walking gaits using optimization techniques [13, 14], gaits based on the natural human gait [15], and gaits based on artificial intelligence techniques such as deep reinforcement learning [16, 17, 18].

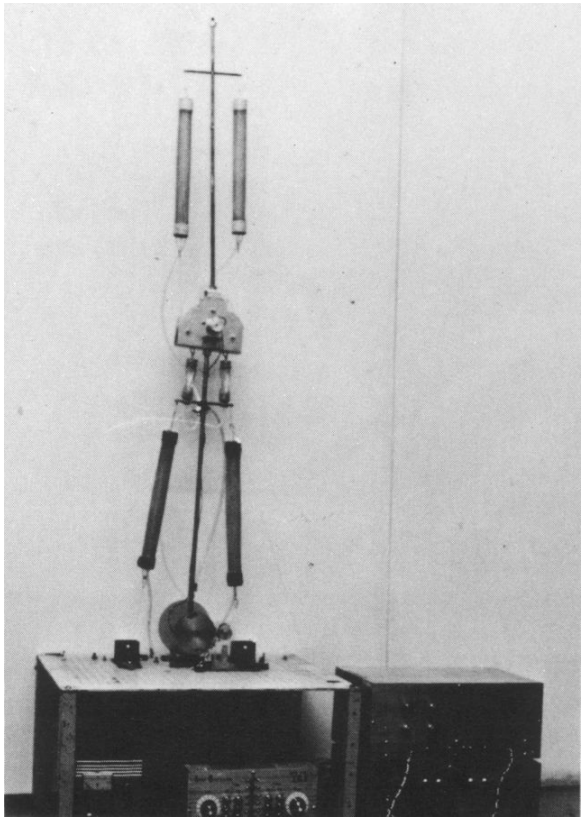


Figure 1.2: Early biped-like double inverted pendulum design to conduct early balance and control testing in 1968

1.3 Literature Review

In recent times, bilateral teleoperation is an area of robotics that has captured many researchers' attention due to its vast applications. Bilateral teleoperation has been implemented in various medical, military, and space exploration fields. For instance, [19, 20, 21] report bilateral teleoperation control algorithms for telesurgery. In [22, 23], algorithms for space robot bilateral teleoperation are presented, while bilateral teleoperation algorithms are designed in [24, 25] for underwater exploration.

Bilateral teleoperation of biped robots has received little attention from researchers, however. As of now, most researchers use exoskeletons as the remote (master robot) to control a biped robot. [26] proposed an adaptive impedance controller to address the challenge of system asymmetries for rehabilitation exoskeletons. [27] developed a new architecture for bilateral teleoperation of biped robots using a seat-like whole-body exoskeleton, while teleoperation of multi-DoF rehabilitation exoskeleton is discussed

in [28]. In general, most teleoperation applications of humanoid robots focus only on robotic arm exoskeleton manipulation [29, 30, 31]. In the mentioned applications of teleoperation above, one of the most significant challenges is to design a stable system that accounts for system asymmetries such as differences in kinematic models, time-varying dynamic parameters, and time-varying communication delays and packet loss.

Numerous control techniques have been implemented to address system asymmetries caused by communication time delays and instability in teleoperation systems. [32] suggests that the most commonly used control techniques can be classified into two categories: passivity and non-passivity-based control methods. Examples of such controllers include wave transformation controllers [33, 34], time-domain passivity control [35], proportional integral derivative (PID) control [36, 37], sliding mode control[38, 39], and adaptive control [40]. Other researchers however focused on communication linear [41, 42] and nonlinear [43] disturbance observers to address communication time delays in teleoperation systems. Unfortunately, time delays are not the only asymmetry to consider when designing a teleoperation system.

The master robot in teleoperation systems is often kinematically different from the slave robot by size or shape. Thus, kinematic asymmetries in master-slave teleoperation systems require specific mapping algorithms to operate as intended. Some of the most popular mapping methods include joint space mapping and operating space mapping. Joint space mapping is often used in kinematically similar systems because it maps joint position and velocity states from one system to another. However, when the master-slave systems are kinematically different by size or shape, it becomes exponentially more challenging to conduct meticulous tasks and control accurately and intuitively. [44] proposed a solution to this problem using workspace mapping. This mapping algorithm uses the scaled position trajectory of the master robot's end-effector obtained using forward kinematics as the desired trajectory for the slave robot. The desired joint trajectories for the slave system are then derived using the slave's inverse kinematics. Haptic feedback for the master robot is derived similarly. The advantage of this approach is that it allows the teleoperation of master-slave systems with high kinematic differences. However, the main drawback of this mapping method is not allowing the operator to use the entire operating space of the

slave/master robot. [45, 46] proposed a hybrid mapping approach for teleoperation systems using both operating and workspace mapping with a transitioning algorithm for smooth switching between the two operation modes. [31] proposed a cartesian-based synergy algorithm using principal component analysis (PCA) to map the most relevant components of a multi-DoF exoskeleton master system to a slave robot, while [47] proposed a supervised neural network approach to map trajectories for bilateral teleoperation systems.

Other mapping algorithms for HRI applications have been implemented. A recent survey on mapping solutions for teleoperation [48] indicates that the existing methods can be divided into two main sections: 1. hand pose estimation and 2. gesture recognition. Hand pose estimation includes joint-to-joint space mapping and workspace mapping, which were covered extensively previously. On the other hand, gesture recognition comprises static and dynamic pose mapping using a camera. [49] used video capture as input to an LSTM neural network to predict and map the user input to a slave robot. Intuitive and personalized robot teleoperation was also achieved using video capture input and machine learning models to map the commands to the slave robot [50, 51].

Machine learning models have many advantages in terms of robustness, computation time, and adaptability instead of conventional methods. Sequence-to-sequence regression models are considered to transform signals from one domain (master) to a signal in another domain (slave). These models have received much attention lately due to their vast potential applications, such as speech and video generation or stock market prediction. In the past, sequence-to-sequence regression was solved in many ways, namely using recurrent neural networks such as LSTMs [52, 53], CNN [54], or a combination of both [55]. With numerous researchers proposing various ideas to solve this problem, [56] introduces the idea of Transformer-like architecture for sequence-to-sequence applications such as Natural Language Processing (NLP) using attention mechanisms. This recent proposal was implemented in various sequence-to-sequence regression problems [57, 58]. Inspired by the idea, [59] proposed an attention-based reinforcement learning model for HRI navigation in crowded environments achieving excellent results and outperforming state-of-the-art models in terms of efficiency in completing tasks.

1.4 Applications

1.4.1 Bilateral Teleoperation Applications

Bilateral teleoperation was applied to various industrial, medical, and military fields. Many advantages can be attributed to bilateral teleoperation, including increased manipulation precision, reduced operating risks in hazardous applications, immersive experience of the slave robot using haptic, visual, or audio feedback, and many more.

For most real-world applications, bilateral teleoperation is applied when controlling a robot in hazardous environments or when high precision is required. [60] developed a platform to teleoperate industrial robots using a virtual visual interface. [61] proposed an intuitive bilateral teleoperation architecture of cable-driven robots for use in hazardous areas such as nuclear power plants. Similarly, [62] proposed a signal filtering approach to bilateral teleoperation in dangerous environments such as power plants, underwater and space exploration. Research has also been conducted on bilateral teleoperation of underwater exploration robots [24, 25], and space robots [63].

In the medical field, bilateral teleoperation has received much attention for application in assisted surgery. Many advances were recently conducted in this field focused on choosing the appropriate control strategy for safety-critical high precision applications. [64] proposed a hybrid system for rate and admittance control to reduce human errors and increase precision, while other researchers focused on different control strategies to achieve the same objectives [65, 66]

1.4.2 Bipedal Robots Applications

Bipedal robots interest many researchers due to their human-like kinematics and their versatility in their movements. Over the years, these robots have been implemented in various applications. The most advanced humanoid robots include Honda Motor Corporation's Asimo, Sophia, the social humanoid robot, and Valkyrie, NASA's exploration humanoid robot. One of the leading companies in this field is Boston Dynamics, specializing in robot control systems and movement versatility. Over the years, this company implemented many-legged robots, including Big Dog and Spot robots, two 4-legged dog-like robots used for hazardous terrain exploration. They

Figure 1.3: Atlas running over an obstacle

have also implemented Handle, a legged and wheeled robot used to handle heavy loads. Yet their most popular robot is the Atlas robot, shown in Figure 1.3. This robot was developed to have human-like movement agility and was conceived to conduct various search and rescue tasks. These robots were programmed to execute highly-agile movements such as running, jumping, dancing, and even flipping.

1.5 Contributions

As mentioned previously, very little work has been done in the bilateral teleoperation control of biped robots. Thus, the main contribution in this work is the design of a new Cartesian-based mapping algorithm for bilateral teleoperation of biped robots using a haptic manipulator.

So far, Most of the current work either uses exoskeletons or focuses on manipulation tasks using humanoid robots. This work introduces a way to control the biped robot's trajectory by capturing the position of the master's end effector then using it as input to the walking gait and inverse kinematics algorithms to reach the desired position. The feedback felt by the operator represents the left-to-right and up-and-right movements of the biped's center of mass as it is walking. This haptic feedback model also allows for future expansion of movements such as lateral walking, jumping,

crouching, or running.

The performance of the proposed Cartesian-based mapping algorithm is then compared to two deep-learning models. To the best of our knowledge, deep learning algorithms have been used to predict and stabilize communication delays in teleoperation systems or map desired trajectories in HRI applications. Little to no research has thus been done to develop a deep learning model for asymmetric bilateral teleoperation systems that maps and accounts for communication time delays in the bilateral teleoperation field.

Lastly, this work is validated through simulation and semi-virtual experimentation. In the simulation, a 3-DoF manipulator, acting as the master device, sends its end-effector position to a 10-DoF biped, acting as the slave device, to track its position. Similarly, the biped's CM position is sent back to the master robot for haptic feedback. The results of this simulation are compared between the two mapping methods. In the semi-virtual experimentation, we program a simple gait algorithm on the ToniPy's 10-DoF biped and collect the CM position data. The stored CM position data is transferred into a simulation platform where we track the haptic feedback on a simulated 3-DoF manipulator.

1.6 Thesis Outline

The thesis is presented as follows. Chapter 1 introduces bilateral teleoperation systems and biped robots, discussing research history and recent advances on these two systems. Chapter 2 discusses the kinematic and dynamic models of the 3-DoF and 10-DoF models and the control system architecture to drive joint state errors to zero. Chapter 3 outlines how the mapping algorithms are implemented, discussing forward and inverse kinematics and the walking gait algorithm for the biped. Chapter 4 discusses the simulation results of both methods and compares them based on their trajectory tracking performance and adaptability. In Chapter 5, we cover an application of the proposed mapping algorithm and discuss the obtained semi-virtual experimental results on the ToniPy's 10-DoF biped robot and a simulated 3-DoF manipulator. Chapter 6 contains a discussion and a conclusion with suggestions for future works.

Chapter 2

Control Systems and Kinematic and Dynamic Model of Biped and Manipulator

This chapter discusses the low-level components needed to achieve the overall objective - bilateral teleoperation. First, the kinematics for both mast and slave are presented. These are used to generate an animation of the robots and track the joint's Cartesian positions. Next, the dynamics needed to simulate the two robots are discussed. Lastly, the chosen control system algorithms required to stabilize the system are discussed and compared to other similarly performing methods. The stability of the system is verified using the Lyapunov stability criterion.

2.1 System Kinematics

To achieve bilateral teleoperation and to design the biped walking gait, it is essential to compute the kinematics of the master and slave systems. The kinematics return the Cartesian position of each joint given their respective joint angles and body dimensions. The Denavit-Hartenberg method is used to derive the kinematics. The DH method states that given a set of space transformations for each joint as:

$$ {}^{0}_{i}T = {}^{0}_{1}T\,{}^{1}_{2}T \ldots {}^{i-1}_{i}T \tag{2.1} $$

where ${}^{i-1}_{i}T$ is the transformation matrix:

$$ {}^{i-1}_{i}T = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2} $$

in which α_i is the link twist, a_i the link length, d_i the link twist, and θ_i the join angle, we can derive the joint Cartesian position in 3-D space as:

$$ x_i = {}^{0}_{i}T_{1,4} \tag{2.3} $$

$$y_i =_i^0 T_{2,4} \tag{2.4}$$

$$z_i =_i^0 T_{3,4} \tag{2.5}$$

where (x_i, y_i, z_i) is the Cartesian position of the i^{th} joint end effector. The robots' kinematic properties are modeled after the haptic Omni manipulator for the master robot and the ThormanG3 biped robot, as specified in Tables 2.1 and 2.2. Figure 2.1 shows the coordinate transformation used to obtain the following DH parameters for the master and slave systems.

Table 2.1: DH parameters for master system

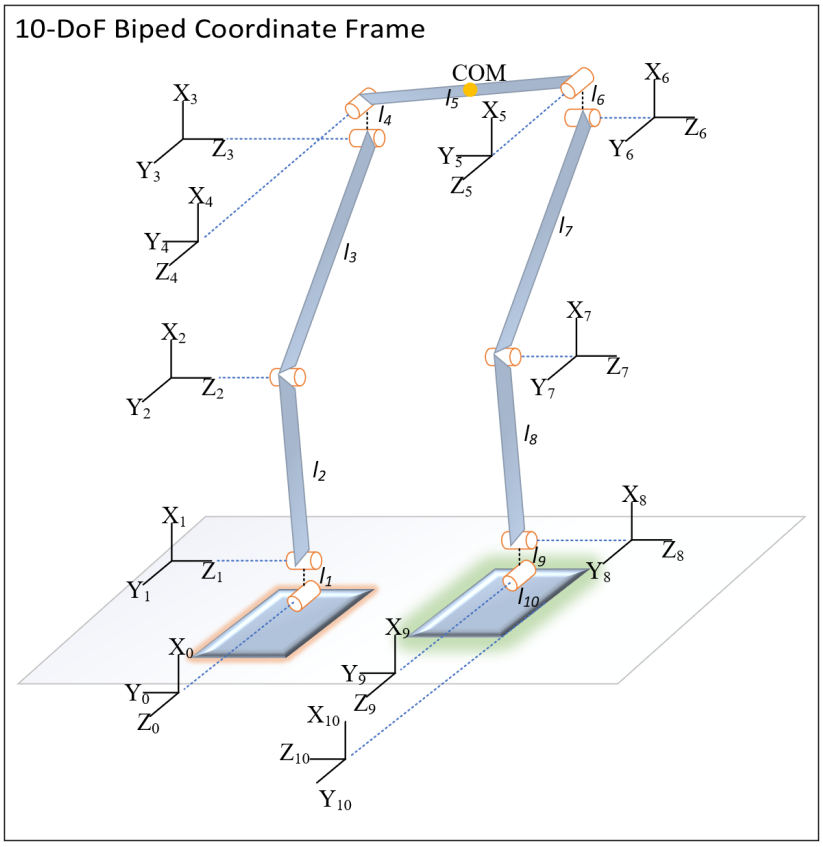
Haptic Manipulator (Master)				
	a_i (cm)	$\alpha_i(rad)$	$d_i(cm)$	$\theta_i(rad)$
1	0	$-\frac{\pi}{2}$	0	θ_{1m}
2	13.2	0	0	θ_{2m}
3	13.2	0	0	θ_{3m}

Table 2.2: DH parameters for slave system

Biped Robot (Slave)				
	a_i (cm)	$\alpha_i(rad)$	$d_i(cm)$	$\theta_i(rad)$
1	7	$\frac{\pi}{2}$	0	θ_1
2	30	0	0	θ_2
3	30	0	0	θ_3
4	7	$-\frac{\pi}{2}$	0	θ_4
5	18.6	0	0	$\theta_5 - \frac{\pi}{2}$
6	7	$-\frac{\pi}{2}$	0	$\theta_6 - \frac{\pi}{2}$
7	30	0	0	θ_7
8	30	0	0	θ_8
9	30	$\frac{\pi}{2}$	0	θ_9
10	4.35	0	0	$\theta_10 + \frac{\pi}{2}$

2.2 Inverse Kinematics

The inverse kinematics computes the corresponding angular position of a joint given its Cartesian position and body dimensions as specified in Tables 2.1 and 2.2. Inverse kinematics is often used to compute joint desired trajectories for a system given some



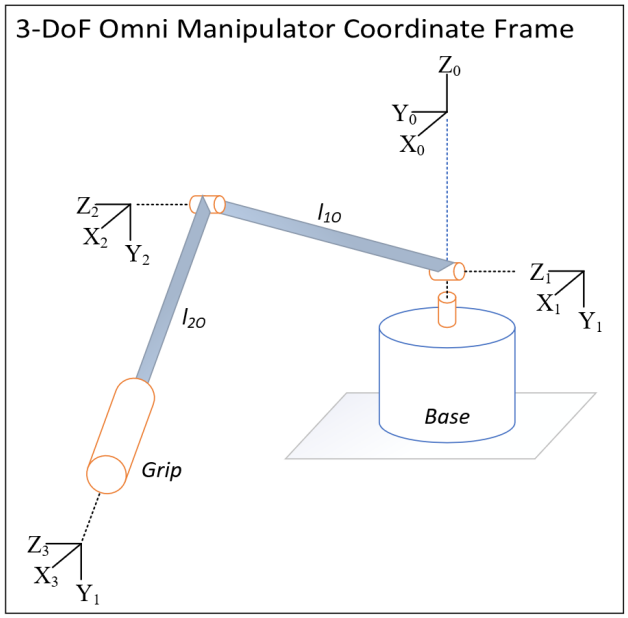


Figure 2.1: Coordinate frames for master and slave systems

trajectory in 3-D space. For the bilateral teleoperation of a biped robot using a haptic manipulator, inverse kinematics is used to determine the desired trajectories for the master device's haptic feedback and determine the desired trajectories of the bipedal robot joints to achieve a stable walking gait.

The biped inverse kinematics are used to determine the specific trajectory for each of the joints [12]. Using the SLIP's states, we compute the trajectory of each joint using inverse kinematics. The desired joint angles for the ankle, knee, and hip joints of the supporting leg rotating about the Y-axis are thus computed as:

$$r\cos\theta_1 = l_2\cos\phi_2 + l_3\cos\phi_3 \tag{2.6}$$

$$r\sin\theta_1 = -l_2\sin\phi_2 + l_3\sin\phi_3 \tag{2.7}$$

in which r and θ_1 are the two first states of the SLIP walking gait model computed at each time step, l_2 and l_3 are the link lengths for the biped robot as defined in Table 2.2, and ϕ_2 and ϕ_3, the angular position for joints 2 and 3.

Rearranging Equations 2.6 and 2.7, we can obtain the corresponding joint position ϕ_2 and ϕ_3:

$$\cos\phi_2 = \frac{1}{2l_2}\left[r\cos\theta_1 + \frac{1}{r}\left(\sin\theta_1\sqrt{-l_2^4 + 2l_2^2l_3^2 - l_3^4 + 2\left(l_2^2 + l_3^2\right)r^2 - r^4} + \left(l_2^2 - l_3^2\right)\cos\theta_1\right)\right] \tag{2.8}$$

$$\cos\phi_3 = \frac{1}{2l_3}\left[r\cos\theta_1 - \frac{1}{r}\left(\sin\theta_1\sqrt{-l_2^4 + 2l_2^2l_3^2 - l_3^4 + 2\left(l_2^2 + l_3^2\right)r^2 - r^4} + \left(l_2^2 - l_3^2\right)\cos\theta_1\right)\right] \tag{2.9}$$

To maintain a vertical upper body position and stabilize the biped robot, the hip joint of the supporting leg is defined as:

$$\phi_4 = -\phi_2 - \phi_3 \tag{2.10}$$

The swing leg also has to mimic the supporting leg to achieve a stable walking gait. The kinematics for the remaining joint rotating about the Y-axis are defined similarly as in Equations 2.8, 2.9, and 2.10:

$$\phi_7 = \phi_2 \tag{2.11}$$

$$\phi_8 = \phi_3 \tag{2.12}$$

To maintain the position of the swing foot flat about the Y-axis upon landing, joint 9 is defined as:

$$\phi_9 = -\phi_7 \tag{2.13}$$

Lastly, we must consider the inverse kinematics of the joints rotating about the Z-axis, joints 1, 5, 6, and 10. These are straightforward and must maintain the upper body position upwards as in Equation 2.10 and must also maintain the foot of the swing leg flat to prepare for landing as in Equation 2.13.

$$\phi_1 = \theta_2 \tag{2.14}$$

$$\phi_5 = 0 \tag{2.15}$$

$$\phi_6 = -\phi_1 \tag{2.16}$$

$$\phi_{10} = 0 \tag{2.17}$$

where θ_2 is the SLIP pendulum angle rotating about the Z-axis. In-depth details of the derivation of the SLIP model are discussed in Section 2.3.

Below, we present the inverse kinematics of the 3-DoF haptic manipulator used to control the biped robot. The manipulator's inverse kinematics are used in the IMA. The inverse kinematics are also used to provide haptic feedback to the operator. The input to the inverse kinematics is the CM of the biped robot (x_{cm}, y_{cm}, z_{cm}), and the output is the desired joint trajectories used to generate the Omni haptic feedback. The resulting joint trajectories of the master system ϕ_{1md}, ϕ_{2md} and ϕ_{3md} are defined as:

$$\phi_{1md} = \text{atan2}\left(y_{cm}, l_{1m} + z_{cm}\right) \tag{2.18}$$

$$\phi_{2md} = \text{atan2}\left(x_{cm}, \sqrt{(l_{1m} + z_{cm})^2 + (y_{cm})^2}\right) \tag{2.19}$$

$$\phi_{3md} = \text{atan2}\left(\beta_1, \beta_2\right) \tag{2.20}$$

where l_{1m} and l_{2m} represent the link lengths of the haptic manipulator, and β_1 and β_2 are defined as:

$$\beta_1 = \sqrt{1 - \beta_2} \tag{2.21}$$

$$\beta_2 = \frac{(\delta_1 x_{cm})^2 + (\delta_2 y_{cm})^2 - l_{2m}^2}{2 l_{1m} l_{2m}} \tag{2.22}$$

2.3 System Dynamics

The system dynamics represent the forces and torques applied at each joint in a body. The dynamic equations for both the master and slave devices in the bilateral teleoperation system are essential to accurately simulate the biped and the haptic manipulator. Additionally, we need the dynamics of the 3-D SLIP model used to generate a stable walking gait for the biped robot. These are computed using the Euler Lagrange dynamic equations described below.

The Euler-Lagrange method states that the equations of motion of any dynamical system can be represented as follows:

$$\tau_i = \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_i} - \frac{\partial L}{\partial \theta_i} \tag{2.23}$$

where τ_i is the input torque or force applied at each joint i, and where L is the Lagrange equation defined as:

$$L(\theta, \dot{\theta}) = KE(\theta, \dot{\theta}) - PE(\theta) \tag{2.24}$$

where KE is the total kinetic energy of the system and PE is the total potential energy of the system. Deriving Equation 2.23 results in the dynamic equations of the system:

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta}) + G(\theta) = \tau + \tau_h + \delta \tag{2.25}$$

where M is the inertia matrix C the Coriolis vector G the gravitational vector τ the control input vector, τ_h the human operator input torque, and δ the noise vector added to simulate real-world uncertainties and disturbances. Note that there is no operator user input for the biped robot and 3D SLIP model $\tau_h = 0$.

Rearranging Equation 2.25, we obtain the system dynamics of the form

$$\ddot{\theta} = f(\theta, \dot{\theta}) + g(\theta)(\tau + \tau_h + \delta) \tag{2.26}$$

given some initial conditions θ_i and control input τ, the system's future states can be simulated by integration.

The dynamics equations for the master and slave robots and the SLIP model for the walking gait were derived using the method mentioned above using a MATLAB Euler Lagrange Tool Package [67]. Below, we discuss the derivation of the 3D SLIP dynamic model in greater detail as an example.

Let us define $c_1 = \cos\theta_1 s$, $c_2 = \cos\theta_2 s$, $s_1 = \sin\theta_1$ and $s_2 = \sin\theta_2$. Given that this system is a 1-joint system revolving about the X and Y axes, the forward kinematics of the SLIP model are simply as follows:

$$x = l\sin\theta_{1s} \tag{2.27}$$

$$y = l\sin\theta_{2s} \tag{2.28}$$

$$z = l\cos\theta_{1s}\cos\theta_{2s} \tag{2.29}$$

The kinetic energy of the system can be computed by the formula

$$KE = \sum_{j=1}^{\infty} \frac{M_j v_j^2}{2} + \frac{I_j \omega_j^2}{2} \tag{2.30}$$

in which M_j, v_j, I_j, and ω_j the mass, linear velocity, moment of inertia and angular velocity of joint j, respectively. Equation 2.30 can be further expanded in the form:

$$KE = \frac{M(\dot{x}^2 + \dot{y}^2 + \dot{z}^2 + \dot{r_s}^2) + I(\dot{\theta}_{1s}^2 + \dot{\theta}_{2s}^2)}{2} \tag{2.31}$$

In this case, deriving equations 2.27, 2.28, and 2.29 results in

$$\dot{x} = lc_1\dot{\theta}_{1s} + \dot{l}s_1 \tag{2.32}$$

$$\dot{y} = lc_2\dot{\theta}_{2s} + \dot{l}s_2 \tag{2.33}$$

$$\dot{z} = l_1(s_1c_2\dot{\theta}_{1s} + c_1s_2\dot{\theta}_{2s}) + \dot{l}c_1c_2 \tag{2.34}$$

The potential energy is described as :

$$PE = Mgz + \frac{K(l-R)^2}{2} \tag{2.35}$$

where K is the spring constant and R the linear displacement of the spring relative to the resting position. The Lagrange can now be defined as in Equation 2.23 and the dynamics can be computed as per steps indicated in Equations 2.23 through 2.25 resulting in a form similar to that described by Equation 2.36:

$$\ddot{X}_s = B^{-1}F + B^{-1}\tau + \delta \tag{2.36}$$

$$B = \begin{bmatrix} M & 0 & 0 \\ 0 & I + Mc_2^2r_S^2 & 0 \\ 0 & 0 & I + Mr_s^2 \end{bmatrix} \tag{2.37}$$

$$F = \begin{bmatrix} K - Mr_s\dot{\theta}_{1s} - Mr_s\dot{\theta}_{2s} + Ms_2^2 r_s \dot{\theta}1s^2 \\ -M\sin(2\theta_{2s})r_s^2\dot{\theta}1s\dot{\theta}2s + 2Mc_2^2 r_s\dot{r}_s\dot{\theta}1s \\ 2Mr_s\dot{r}_s\dot{\theta}2s + \frac{M\sin 2\theta_{2s}r^2\dot{\theta}_{1s}^2}{2} + glMc_2 \end{bmatrix} \tag{2.38}$$

in which $r_s = l - R.$ and the X_s the state vector described as:

$$X_s = \begin{bmatrix} r_s \\ \theta_{1s} \\ \theta_{2s} \end{bmatrix} \tag{2.39}$$

2.4 Control System

A well-tuned control system is essential to maintain the stability of any dynamic system. Over the years, many control approaches have been developed for various applications. This section discusses the controller choice and derives the stability analysis of the master and slave systems.

2.4.1 Master and Slave Control Systems

First, we discuss the control system of the haptic manipulator and biped robot. Manipulators and biped have been known to have large nonlinearities in their dynamic model, which makes them harder to control accurately. In designing a control system for these systems, we need to take into account some of the stochastic disturbances and uncertainties of the system and potential external factors that could impact the system's overall performance. Additionally, given that the objective of this research is bilateral teleoperation, one must also consider communication time delay variations which could also lead to instability in the system.

For the reasons mentioned above, we chose a model-based nonlinear controller given that we have the dynamic model of the manipulator. The selected controller is a second-order sliding mode controller due to its known robustness to system uncertainties and disturbances and its reduced chattering effect at the control input, resulting in smooth and accurate operation of the manipulator's actuators. As for the communication time delay, we assume that they are negligible and do not affect the system's stability.

The super-twisting second-order sliding mode controller is widely used to suppress the chattering problem of the first-order SMC. The control law for this algorithm is straightforward. First, the sliding surface is designed as:

$$s = \left(\frac{d}{dt} + \lambda\right)\theta_e \tag{2.40}$$

where λ is a scalar and θ_e the error vector defined as:

$$\theta_e = \theta_{measured} - \theta_{desired} \tag{2.41}$$

The control law of the second order sliding mode controller is then defined as:

$$\tau = \lambda\sqrt{|s|}sign(s) + \omega \tag{2.42}$$

where ω is described as:

$$\omega = \beta \int sign(s)dt \tag{2.43}$$

and where β is a constant vector. The resulting control law can then be defined as an addition of the equivalent and sliding continuous control:

$$\tau = g^{-1}\left(-f - \lambda\dot{\theta} - \lambda\sqrt{|s|}sign(s) - B\int sign(s)dt\right) \tag{2.44}$$

with the control law defined, we can now validate the system's stability with the Lyapunov stability Criterion. First, we define a positive definite function and its negative definite derivative

$$V = \frac{1}{2}s^T s \tag{2.45}$$

$$\dot{V} = s\dot{s} \leq 0 \tag{2.46}$$

Substituting the derivative of Equation 2.40 into 2.46, we obtain:

$$\dot{V} = s(\ddot{\theta}_e + \lambda\dot{\theta}_e) \tag{2.47}$$

Assuming that the derivative of $\theta_{desired}$ is negligible and that the disturbance $\delta = 0$, we can substitute the dynamic equations of the system defined by Equation 2.36

$$\dot{V} = s(f(\theta, \dot{\theta}) + g(\theta)\tau + \lambda\dot{\theta}_e) \tag{2.48}$$

where τ is as defined by Equations 2.42, 2.43 and 2.44 resulting in the following:

$$\dot{V} = s\left(-\lambda\sqrt{|s|}sign(s) - B\int sign(s)dt\right) \tag{2.49}$$

$$\dot{V} = -\lambda|s|\sqrt{|s|} - Bs\int sign(s)dt \tag{2.50}$$

Given that λ and β are positive definite constants, given that the term $|s|\sqrt{|s|}$ can also be said to be positive semi-definite, and given that the term $s\int sign(s)dt$ can be said to be positive semi-definite over small integration periods, we can then confirm that $\dot{V} = s\dot{s} \leq 0$.

2.4.2 SLIP Control System

Next, we discuss the control system needed for the SLIP model gait. This control system takes for input a desired CM position for the biped and outputs future states of the SLIP model used as inputs for the biped's inverse kinematics described by Equations 2.8 - 2.17. Section 3 will explain in greater detail how the SLIP model is used to generate trajectories for the biped robot. We chose to use a simple feedback linearization controller to stabilize the SLIP model described by Equation 2.36. Typically, feedback linearization controllers are not ideal in stabilizing highly nonlinear systems due to uncertainties in the model; however, for simplicity, we will assume that the dynamics of the SLIP model are ideal $\delta = 0$. To stabilize the system described by 2.36, the feedback linearization controller is described as:

$$\tau = B(-B^{-1}F - K_1X_{se}) \tag{2.51}$$

where X_{se} is the state error vector as in Equation 2.41, and K_1 is a positive constant vector. The stability proof for this controller is simple and is as follows. First, we define a positive definite Lyapunov function

$$V = \frac{1}{2}X_{se}^2 \tag{2.52}$$

$$\dot{V} = X_{se}\dot{X}_{se} \tag{2.53}$$

substituting Equation 2.36 into Equation 2.53

$$\dot{V} = X_{se}(B^{-1}F + B^{-1}\tau) \tag{2.54}$$

then, substituting Equation 2.51 into Equation 2.54

$$\dot{V} = X_{se}(B^{-1}F + B^{-1}(B(-B^{-1}F - K_1X_{se}))) \tag{2.55}$$

resulting in

$$\dot{V} = -K_1 X_{se}^2 \tag{2.56}$$

As long as the first derivative of the Lyapunov function described by Equation 2.56 is negative definite, the system is stable. To maintain this condition, K_1 must be a positive scalar.

2.4.3 Disturbance Observer

As mentioned previously, The precise control of highly nonlinear systems such as biped or manipulator robots is challenging. There are often significant model uncertainties and disturbances that may cause poor and inadequate control system performance. Here, we implement a simple nonlinear disturbance observer (NDO), which has to estimate the model uncertainties and external disturbances. The estimated uncertainties are added to the control system as the NDO's input.

The NDO's input can be expressed as:

$$\hat{\tau}_d = z - f(\dot{\theta}) \tag{2.57}$$

where z is an auxiliary variable vector, and $f(\dot{\theta})$ is a function defined as:

$$f(\dot{\theta}) = L\dot{\theta} \tag{2.58}$$

where L is a scalar, and $\dot{\theta}$ is defined the joint angular velocity states. The auxiliary variable vector is defined as:

$$z = \int L\left(f(\theta, \dot{\theta}) + g(\theta)(\tau + \hat{\tau}_d)\right)dt \tag{2.59}$$

The total control input of the master and slave systems is then represented by combining Equations 2.44 and 2.58:

$$\tau_{total} = g^{-1}\left(-f - \lambda\dot{\theta} - \lambda\sqrt{|s|}sign(s) - B\int sign(s)dt + \hat{\tau}_d\right) \tag{2.60}$$

The stability proof of the NDO is simple. First, we compute the first derivative of Equations 2.57 and 2.59:

$$\dot{\hat{\tau}}_d = \dot{z} - L\ddot{\theta} \tag{2.61}$$

$$\dot{z} = L\left(f(\theta, \dot{\theta}) + g(\theta)(\tau + \hat{\tau}_d)\right) \tag{2.62}$$

using the dynamics model previously derived in Equation 2.36, and substituting Equations 2.62 into 2.61, we get:

$$\dot{\hat{\tau}}_d = L\left(f(\theta,\dot{\theta}) + g(\theta)(\tau + \hat{\tau}_d)\right) - L\left(f(\theta,\dot{\theta}) + g(\theta)(\tau + \delta)\right) \tag{2.63}$$

For simplicity, we will redefine the NDO's control input $\hat{\tau}$ as the estimated disturbance $\hat{\delta}$. We then get:

$$\dot{\hat{\delta}}_d = Lg(\theta)(\hat{\delta}_d - \delta_d) \tag{2.64}$$

or simply:

$$\dot{\hat{\delta}}_d = Lg(\theta)(e_d) \tag{2.65}$$

where e_d is the error between the estimated and actual system disturbances and uncertainties. Computing the first derivative of e_d, we get:

$$\dot{e}_d = \dot{\hat{\delta}}_d - \dot{\delta}_d \tag{2.66}$$

which, assuming the disturbance and uncertainty to be constant, the term $\dot{\delta}$ is equal to 0:

$$\dot{e}_d = \dot{\hat{\delta}}_d \tag{2.67}$$

Substituting Equation 2.67 into 2.67, we get the following differential equation:

$$\dot{e}_d = Lg(\theta)e_d \tag{2.68}$$

We can ensure that the differential Equation 2.68 will converge as long as the condition $Lg(\theta) < 0$ is met. This is done by cancelling the inverse inertia matrix $g(\theta)$ and defining the gain L as:

$$L = -K_d g^{-1}(\theta) \tag{2.69}$$

where K_d is a positive constant.

2.4.4 Adaptive Feedforward Neural Network Compensator

The adaptive neural network compensator is an additional tool along with the nonlinear disturbance observer to compensate for any uncertainty in the system and reduce trajectory tracking error. This feedforward neural network uses the desired trajectory of the system as input and feeds it through a radial basis function neural network to

compute the torque compensation. The weights of this neural network are adaptively calculated based on the previous output of the Gaussian operation and the current error of the system.

For the haptic manipulator, the input to the feedforward compensator is defined as:

$$S_1 = \begin{bmatrix} \phi_{1md} \\ \phi_{2md} \\ \phi_{3md} \end{bmatrix} \tag{2.70}$$

The input defined in Equation 2.70 is fed through the n^{th} radial basis function neural network with smooth Gaussian kernel defined as:

$$H_n = e^{-\frac{S_1-(\mu_{RBF}+\mu_O)}{\sigma^2_{RBF}}} \tag{2.71}$$

where μ_{RBF} and σ_{RBF} are the mean and the standard deviation of the Gaussian Function. K_{RBF} is the number of Gaussian functions in the RBF neural network, each separated by an interval μ_O.

The output of the operation defined by Equation 2.71 is then multiplied with the adaptive weights using the dot product to generate the compensated torque Y_{RBF}. These weights are defined as:

$$W_{RBF} = \beta_{RBF}\theta_e H_n \tag{2.72}$$

where β_{RBF} is a constant positive vector.

Equation 2.60 can once again be updated to include the feedforward adaptive neural network compensator:

$$\tau_{total} = g^{-1}\left(-f - \lambda\dot{\theta} - \lambda\sqrt{|s|}sign(s) - B\int sign(s)dt + \hat{\tau}_d\right) - Y_{RBF} \tag{2.73}$$

2.5 Heuristic Optimization of Control System Parameters

The genetic optimization algorithm was first introduced by [68] and has ever since been utilized in various optimization applications. The genetic algorithm is based on the evolutionary and natural selection process and has been extensively used in the parameters tuning of control systems. The GA, which runs on a continuous loop of generations to find the optimal solution, consists of three main steps: reproduction,

mutation, and crossover. The GA first begins by computing the performance of the fitness function with an initial population of chromosomes. In the case of an unoptimized solution, the population goes through selection, crossover, and mutation, which generates a second candidate population for the fitness function.

In this research, the GA from MATLAB is used to obtain optimal gain values for the presented controllers for the master and slave systems. The genetic algorithm with a predefined population size of 50 iteratively computes the fitness function to find the best solution. The GA completes its optimization under two conditions: 1- if the number of consecutive stall generations is equal to the number of max stall generations, set at 50 generations. A stall generation is defined as a generation where the computed fitness function is not better than the previous generation. And 2- if the weighted average relative change is smaller or equal to the function tolerance parameter. Table 2.3 shows the GA parameters used to optimize the presented systems.

Table 2.3: Genetic algorithm optimizer parameters

Parameter	Value
Exploration Range	[-1000 1000]
Population Size	50
Function Tolerance	$10^{-}6$
Max Number of Stall Generation	50

Chapter 3

Proposed Mapping Mapping Algorithm for Bilateral Teleoperation

So far in Chapter 2, we discussed the essential low-level components required for bilateral teleoperation of a biped robot using a manipulator. The forward and inverse kinematic models used in the master system for the FMA and the slave system for IMA have been presented. The dynamic equations used to simulate both master and robots have been derived, and the control systems used to stabilize the master and slave systems were presented. This chapter presents the high-level components needed to operate this system. More specifically, we discuss the forward and inverse mapping algorithms used to map master and slave system trajectories to achieve bilateral teleoperation. The architecture presented in Figure 3.1 illustrates how the system's components interact with each other.

First, we will discuss the FMA and how it operates. In this subsection, we discuss how the forward kinematic of the haptic manipulator generates a command to the slave and how the step planning and walking gait algorithms compute joint trajectories for the bipedal Robot. More specifically, we will discuss how inputs from the master device are interpreted, how and when a step has been completed, which leg to use for the supporting phase of the next step, the parameter adaptation when switching from one leg to another, how the previously discussed SLIP model can track a certain trajectory, and the inverse kinematics used to derive the joint trajectories.

Next, we discuss the IMA and how it can provide relevant haptic feedback of the biped's movements to the operator. This subsection will investigate which features of the biped's dynamics are chosen for haptic feedback and how the inverse kinematics can provide an accurate representation of that.

Lastly, we compare the proposed Cartesian-based method to a deep learning-based approach. Two deep learning models are used and tested to verify their performance at predicting trajectory mapping under time delay circumstances. The deep learning

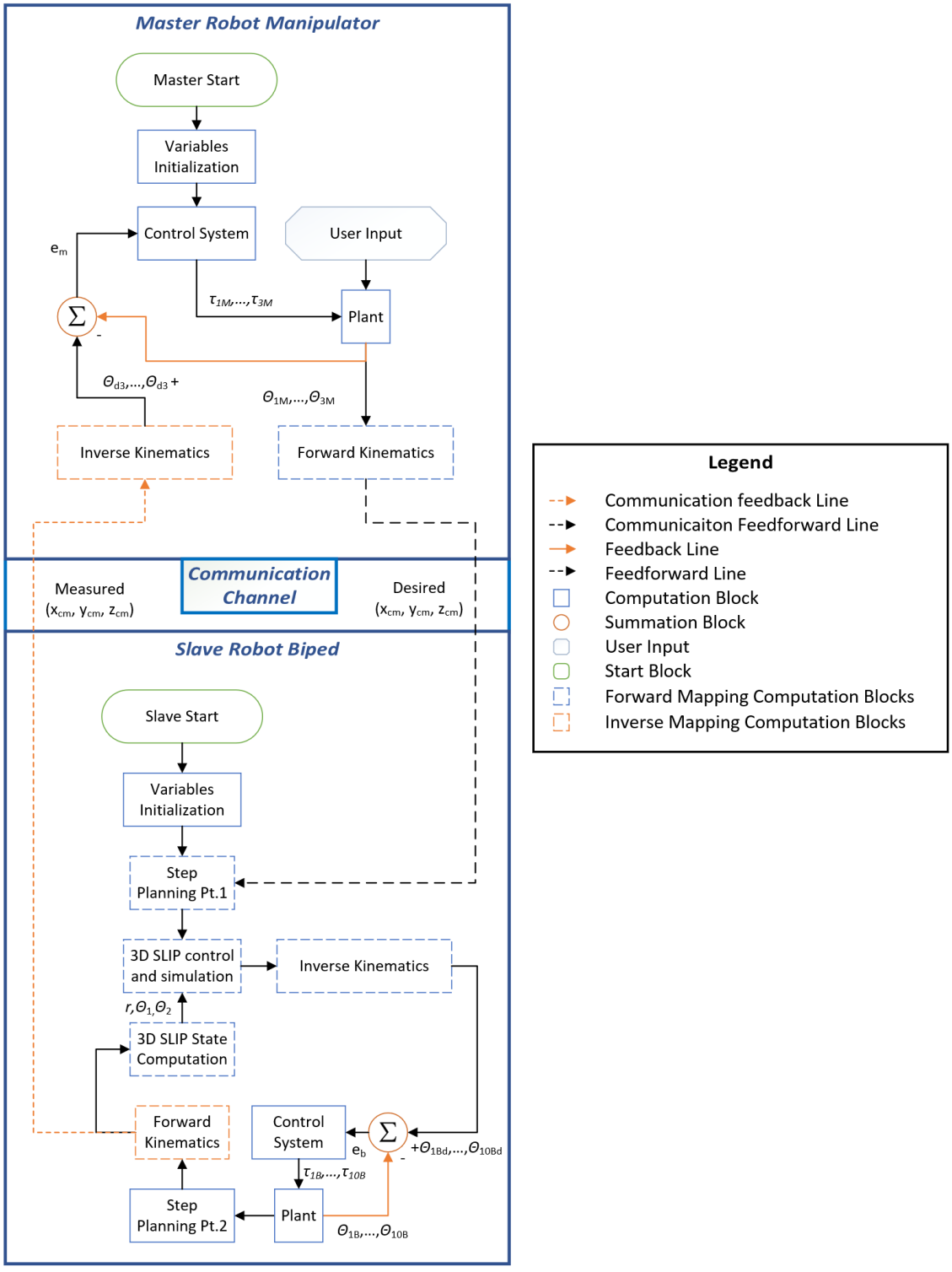


Figure 3.1: System's software flow diagram

architecture includes an LSTM-based architecture and an LSTM-CNN architecture. The data collection, preprocessing, and training parameters are discussed in depth.

3.1 Forward Mapping Algorithm

This subsection presents the forward mapping algorithm used to map trajectories from the master system (manipulator's end effector) to the slave system (biped's CM). More specifically, we discuss how each component interacts to form the FMA.

3.1.1 Master Signal Interpretation and Targets

This subsection describes the core of the forward mapping algorithm. It is what translates master commands to slave commands for the biped robot gait. Given the complexity of the two systems (master and slave), we decided that for simplicity, we will only consider three possible actions that the operator can control: Forward walking, backward walking, and stop.

These functions are dictated by the manipulator's end-effector position in its coordinate frame. First, we defined a dead zone at the center position of the manipulator end effector's coordinate frame for the 'stop' action. This dead zone representing the 'stop' action is defined bounded by α_{min} and α_{max}. Whenever $x_{master} \leq \alpha_{min}$ the biped walks backward and whenever $x_{master} \geq \alpha_{max}$:

$$z_{desired} = \begin{cases} -\infty, & x_{master} \leq \alpha_{min} \\ z_{hip}, & \alpha_{min} \leq x_{master} \leq \alpha_{max} \\ \infty, & x_{master} \geq \alpha_{max} \end{cases} \tag{3.1}$$

Note that for the 'stop' action, the $z_{desired}$ variable is set to $z_{hip} = z_4$ to maintain the biped's current location and stop moving. This is explained below in Equation 3.8 where the error of the step planning algorithm is 0.

3.1.2 Step planning

The first step planning function computes the trajectories followed by the SLIP model gait. It calculates the allowable hip joint distance that can be traveled in a single step in each direction given the desired XYZ position by the master system's forward

kinematics and the measured hip position, as shown in Figure 3.1. This function computes the maximum situational step distance (MSSD). The error determines the desired position for the CM of the biped based on the magnitude of the error. When the error is less than the MSSD, or the target CM position can be reached with a single step, the desired position is set to the error. Conversely, when the error is greater than the MSSD, or the target requires multiple steps to be reached, the desired position is set to the MSSD. In that case, to get to the target, a loop monitors the biped's states to determine if a step is completed to activate the next supporting leg to converge to the target desired position. More details on the step monitoring and switching are discussed in Section 3.1.5.

Before we compute the MSSD, we must first determine the absolute maximum step distance (AMSD).

$$AMSD = 2l_2l_3 \sin\theta_{max} \tag{3.2}$$

We now introduce a set of parameters that we will use to compute the MSSD. These parameters vary under two conditions:

1. when a step is completed,

2. when the walking direction has been altered (forward-to-backwards walking or vice versa)

In a simpler way, these parameters are the initial conditions of the biped prior to taking a new step:

$$d_0 = \sqrt{(x_{3N} - x_{1N})^2 + (z_{3N} - z_{1N})^2} \tag{3.3}$$

$$\gamma_0 = \arctan\left(\frac{z_{3N} - z_{1N}}{x_{3N} - x_{1N}}\right) \tag{3.4}$$

where d_0 is the euclidean distance between the ankle and the hip joints of the supporting leg before taking a step, γ_0 the initial angle of the supporting leg, and N the time step at which one of the two conditions above is met.

The MSSD is defined as the allowable distance that can be traveled by the hip joint (COM) given the initial position of the supporting leg. The MSSD is computed as:

$$MSSD_+ = \sqrt{(l_2l_3)^2 + d_0^2 - 2(l_2l_3)^2 \cos(\theta_{max} + \gamma_0)} \tag{3.5}$$

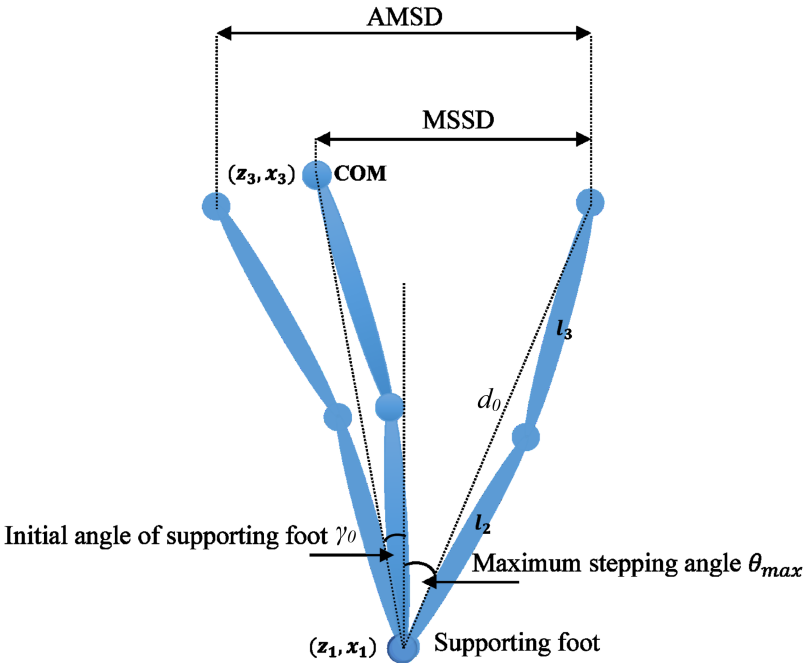


Figure 3.2: Important definitions of the supporting foot during the stepping motion

The MSSD is a direction-dependent parameter, hence the subscript $MSSD_+$ indicating the MSSD value when the desired direction is in the positive direction (forward walking). Alternatively, the MSSD for the negative desired direction is given as:

$$MSSD_- = AMSD - \sqrt{(l_2 l_3)^2 + d_0^2 - 2(l_2 l_3)^2 \cos(\theta_{max} + \gamma_0)} \tag{3.6}$$

Note that the MSSD is bounded by the AMSD

$$MSSD \leq AMSD \tag{3.7}$$

Figure 3.2 shows the definitions needed to complete the step planning algorithm.

Once these parameters are computed, we determine the error in position of the hip joint

$$e = z_{desired} - z_4 \tag{3.8}$$

where $z_{desired}$ and z_4 is the desired hip position and the current position of the hip joint, respectively.

We can determine the desired position for the CM with the computed error and the maximum situational stepping distance. If the error is greater than the MSSD, the desired hip position (DP) is set to the MSSD. By doing this, we ensure that the

movement of the SLIP is bounded to θ_{max}. Conversely, when the error is smaller than the MSSD, the desired CM position, DP, is set to the error itself.

$$DP = \begin{cases} e, & e \leq MSSD \\ MSSD, & e > MSSD \end{cases} \tag{3.9}$$

the desired angular position for the SLIP can then be computed as:

$$\theta_{1d} = \arctan \frac{DP - (z_{1Ns} - z_{3Ns})}{x_{3Ns} - x_{1Ns}} \tag{3.10}$$

where the subscript $_{Ns}$ is an index to the last time a step was detected while the direction is constant.

3.1.3 Trajectory Generation for SLIP model

The trajectory generation for a biped robot gait is crucial to the system's stability. In Equation 3.10, we defined the desired position of the SLIP model for the second state θ_1 of the 3D SLIP model defined in Equation 2.36. Additionally, the desired trajectory for the first state of the 3D SLIP model is set to:

$$r_d = l_2 + l_3 \tag{3.11}$$

for simplicity. This is what generates the up-and-down movement of the biped's CM as it is walking. More specifically, when $\theta_1 = \theta_{max}$, the CM is at its lowest point, and when the CM is aligned with the position of the supporting foot $\theta_1 = 0$, the CM is at its highest position. In this subsection, we discuss how lateral movement is generated.

In natural biped robot gaits, there is a subtle left-to-right movement of the biped's CM as it is walking. This movement is often compared to Kepler's laws of planetary motion. This motion is typically defined by a hyperbolic function dependent on parameters such as CM height and step length. In this study, we generate the left-to-right movement using a simple Gaussian function as the desired trajectory to simplify the task.

The desired trajectory in the Y-axis, which dictates the left-to-right movement, is given by the Gaussian function as follows:

$$y_{cm-d} = Ae^{-\frac{(z_{HIP-i} - z_{SF-i})^2}{2\sigma_l^2}} \tag{3.12}$$

where z_{HIP-i} and z_{SF-i} is the coordinates of the hip joint and the supporting foot along the Z-axis at time step i, and A and σ are tunable scalars.

The desired Cartesian position y_{cm-d} is converted to an angular desired position for the lateral movement as:

$$\theta_{2d} = \arctan \frac{y_{cm-d}}{x_{HIP-i} - x_{SF-i}} \tag{3.13}$$

where x_{HIP-i} and x_{SF-i} is the coordinates of the hip joint and the supporting foot along the X-axis at time step i.

Note that given that there is no joint rotating about the X-axis in the 10-DoF biped model presented in Figure 2.1, and assuming that the biped robot walks solely in the Z-axis, the coordinate for the hip joint are simply defined as:

$$x_{HIP} = x_4 \tag{3.14}$$

$$z_{HIP} = z_4 \tag{3.15}$$

and the coordinates for the supporting foot are as follows:

$$x_{SF} = x_0 \tag{3.16}$$

$$z_{SF} = z_0 \tag{3.17}$$

which can be obtained using the forward kinematics as discussed in Section 2.1.

The resulting desired trajectory for the 3D SLIP model is as follows:

$$\Theta_{sd} = \begin{bmatrix} l_2 + l_3 \\ \arctan \frac{DP-(z_{1i}-z_{3i})}{x_{3i}-x_{1i}} \\ \arctan \frac{y_{cm-d}}{x_{HIP-i}-x_{SF-i}} \end{bmatrix} \tag{3.18}$$

3.1.4 3D SLIP Model States, Control, Simulation and Inverse Kinematics

In this subsection, we use the previously derived SLIP dynamics, control system, and desired trajectories as defined in Equations 2.36, 2.51, and 3.18, respectively.

First, we compute the SLIP's system states using the output of the inverse kinematics. This can be done as:

$$X_s = \begin{bmatrix} r_s \\ \theta_{1s} \\ \theta_{2s} \end{bmatrix} = \begin{bmatrix} \sqrt{(\frac{x_{4i}+x_{5i}}{2} - x_{0i})^2 + (\frac{y_{4i}+y_{5i}}{2} - y_{0i})^2 + (\frac{z_{4i}+z_{5i}}{2} - z_{0i})^2} \\ \arctan\left(\frac{z_{4i}-z_{1i}}{x_{4i}-x_{1i}}\right) \\ \arctan\left(\frac{\frac{y_{4i}+y5i}{2}-y_{1i}}{x_{4i}-x_{1i}}\right) \end{bmatrix} \tag{3.19}$$

With the computed states and desired trajectories as in Equation 3.18, we define the error:

$$X_{se} = X_s - \Theta_{sd} \tag{3.20}$$

where X_s and X_{se} were previously defined as the SLIP model's state vector and state error vector, respectively. With the state error vector now defined, we use Equation 2.51 to control the SLIP model using virtual inputs. Note that for optimal results, the constant vector K_1 must be defined as:

$$K_1 = \frac{1}{dt^2}\rho, \qquad \rho \in [0.1, 10] \tag{3.21}$$

where dt is the sampling time, and ρ is a constant positive vector. The division by dt^2 is necessary for the simulation of the 3D SLIP model due to the state integration during simulations.

The simulation of the SLIP model is straightforward. Here, we use the previously derived SLIP dynamic model as shown in Equation 3.18 to simulate and predict the future states of the system.

Given an initial position X_{s0} and velocity $\dot{X}_{s0}$ states given a virtual control input defined by Equation 2.51, we simulate the future states of the SLIP dynamic model X_{si+1} by double integrating the dynamic model defined by Equation 2.36:

$$X_{si+1} = X_{s0} + \int \dot{X}_{s0} + \int \left(B^{-1}F + B^{-1}\tau + \delta\right) dtdt \tag{3.22}$$

In the discrete-time domain, the sampling time must be small enough to maintain the system's stability. However, integrating acceleration $\ddot{X}_s$ with a small-time sampling dt results in minor position and velocity changes per single sampling time. This is the main reason why K_1 in Equation 2.51 must be very large (multiplied by a factor of $\frac{1}{dt^2}$ to cancel the double integration's squared time sampling) to have a noticeable effect on the SLIP model.

The result of the 3D SLIP simulation is the future state vector of the SLIP model:

$$X_{si+1} = \begin{bmatrix} r_s \\ \theta_{1s} \\ \theta_{2s} \end{bmatrix} \tag{3.23}$$

which is used as the input to the inverse kinematics discussed in Section 2.2. The future states of the SLIP model are fed into Equations 2.8 to 2.17 to produce a joint

angle desired trajectory which will be tracked by the biped's control system defined in Equation 2.44.

3.1.5 Step Detection and Switching

In this subsection, we discuss the final component necessary for the slave system, as shown in Figure 3.1. This component dictates how each step is detected in the walking motion, and the other leg is activated to complete the next step. Given that the slave system is assumed to be most often in the single support phase (SSP), there are many components of the system that depend on the forward kinematics and the dynamics at the state of the biped at time t. These variables that dictate the operation of the system need to be organized accordingly anytime a new step is completed to prepare for the following step and send appropriate control signals to the actuators.

The step detection algorithm has in fact, two components mentioned previously in Section 3.1.2:

1. when a step is completed, and the direction is constant,

2. when the walking direction has been altered (forward-to-backwards walking or vice versa)

The first condition which detects when a step is detected during constant heading direction (forward or backwards) monitors the error states θ_{1e} and θ_{2e} of the 3D SLIP model state error vector X_{se}. Whenever these states maintain a small enough error $\theta_{1e} \leq 1^\circ$ and $\theta_{2e} \leq 1^\circ$ for a certain period of time $T_{step} = 0.5 seconds$, then the step flag is detected and the next-leg switch algorithm is triggered. When a step is detected, we need to ensure that variables are correctly defined for the next step. This is mainly done because the biped is modeled after its SSP, which affects its kinematics and dynamics.

Table 3.1 shows a description of variables and their initialization before and after taking a step. For instance, if the left leg is in the SSP and completes a step, the forward kinematics saves the last values of x_1, y_1, and z_1 as the new values of x_{10}, y_{10}, and z_{10}. This ensures that the initial conditions of the forward kinematics are correct for the following step action. Similarly, joint position states q_1 through q_{10} from the

Table 3.1: Variable reinitialization following a completed step

	SSP Variables		
	Left Leg in SSP	Right Leg SSP	True Joint Variables
Forward Kinematics	x_1	x_{10}	X_1
	y_1	y_{10}	Y_1
	z_1	z_{10}	Z_1
Control System states	q_1	q_{10}	Q_1
	q_2	q_9	Q_2
	q_3	q_8	Q_3
	q_4	q_7	Q_4
	q_5	q_6	Q_5
	q_6	q_5	Q_6
	q_7	q_4	Q_7
	q_8	q_3	Q_8
	q_9	q_2	Q_9
	q_{10}	q_1	Q_{10}
Actuator Control Inputs	τ_1	τ_{10}	A_1
	τ_2	τ_9	A_2
	τ_3	τ_8	A_3
	τ_4	τ_7	A_4
	τ_5	τ_6	A_5
	τ_6	τ_5	A_6
	τ_7	τ_4	A_7
	τ_8	τ_3	A_8
	τ_9	τ_2	A_9
	τ_{10}	τ_1	A_{10}

dynamics model in Equation 2.36 which are based off the SSP and not the true joint of the robot Q_1 through Q_{10}, are initialized as q_{10} through q_1, respectively, for the following step. Most importantly. however, is the correct assignment of control input signals to the appropriate joint actuators. Below we see that in the case of left leg in SSP, control signals τ_1 through τ_{10} are assigned directly to actuators A_1 through A_{10}, while it is the inverse for the case that the right leg is in SSP. it is thus evident why we need to properly store and initialize variables every time a step is detected.

Comparatively, the second condition detects whenever the walking direction is altered. This is simple to see:

$$direction = sign(z_{desired}) \tag{3.24}$$

where we arbitrarily define that $direction = 1$ indicates a forward walking motion and $direction = 0$ a backwards walking motion. When detecting a direction change, Equation 3.10 is altered to the following until a step is completed in the new direction:

$$\theta_{1d} = \arctan \frac{DP - (z_{9Nd} - z_{6Nd})}{x_{6Nd} - x_{9Nd}} \tag{3.25}$$

where the subscript $_{Nd}$ is an index to the last time a change in direction was detected.

3.2 Inverse Mapping Algorithm

As Opposed to the FMA, the IMA is far easier to design, and it implements the well-known and straightforward Cartesian-based mapping method. This method relies on the computation of forward kinematics of the biped robot to obtain its CM trajectory and the inverse kinematics to transform that same trajectory into readable desired joint positions for the master manipulator to generate haptic feedback. Simple amplitude shifting and scaling are necessary to account for the significant kinematic asymmetries of the two systems.

Section 2.1 discusses the forward kinematics of the biped, and Section 2.2 discusses the inverse kinematics of the haptic manipulator that is used as the master robot. Using the forward kinematics, we obtain the hip position of the biped:

$$x_{hip-i} = (x_{4i} + x_{5i})/2 \tag{3.26}$$

$$y_{hip-i} = (y_{4i} + y_{5i})/2 \tag{3.27}$$

$$z_{hip-i} = (z_{4i} + z_{5i})/2 \tag{3.28}$$

where x_{hip-i}, y_{hip-i}, and z_{hip-i} are the (x,y,z) coordinates of the hip joint at time i.

Given that the manipulator and the biped robots lie on two different coordinate frames (X, Y, Z axes in the master robot are Z, Y, X axes in the slave robot, respectively), the input to the desired trajectory (x_{cm}, y_{cm}, z_{cm}) is defined as:

$$x_{cm} = l_{1m} - \delta_1(z_{hip-i} - z_{hip-Ns}) \tag{3.29}$$

$$y_{cm} = \delta_2(y_{hip-i} - y_{hip-Ns}) \tag{3.30}$$

$$z_{cm} = l_{1m} - \delta_3(x_{hip-i} - x_{hip-Ns}) \tag{3.31}$$

where l_{1m} is link 1 of the haptic manipulator as defined in table 2.1, and where δ_1, δ_2, and δ_3 are scalars.

Note the scaling variables δ_1, δ_2, and δ_3, which are used to scale up the small trajectory of the biped's CM for haptic feedback. Given that the objective is to represent the haptic feedback dictated by the CM's left-to-right and up-and-down movements, haptic feedback along the X-axis of the manipulator is not considered, resulting in $\delta_1 = 0$. Additionally, an offset by x_{hip-Ns}, y_{hip-Ns}, and z_{hip-Ns} representing the (x,y,z) position of the CM at the time of the previous step is subtracted to account for the height of the robot at the hip, the mean y and mean z location of the biped's CM to minimize large and misrepresentative haptic feedback to the operator.

3.3 Deep Learning Mapping Algorithms

This section discusses an alternative mapping algorithm based on deep neural networks. Two architectures are explored: (1) an architecture based on deep LSTM networks and (2) an architecture based on the CNN-LSTM model. This section demonstrates the data collection and preprocessing procedure, and the two architectures are explained. The goal is to observe if the supervised deep learning models can accurately predict and outperform mapping values for FMA and IMA algorithms presented in Sections 3.1 and 3.2 under a unidirectional communication time delay of $dt_c = 0.1s$.

3.3.1 Data Preprocessing

It is essential to layout and understand how the data is collected and preprocessed in a supervised learning task before training the model. The collected data is from simulations of the previously discussed Cartesian-based FMA-IMA algorithms in this task. In this simulation, we randomized parameters of the master and slave robots such as the initial conditions, system dynamics, and user inputs to the master robot to obtain real-world-like training data.

The deep learning model set to act as the new FMA accepts as input the Omni manipulator's joint position states θ_{1m}, θ_{2m}, and θ_{3m} and outputs desired joint trajectories for the biped θ_{1Bd}, θ_{2Bd}, ... θ_{10Bd}. As for the IMA deep learning model, it accepts the biped joint positions θ_{1B}, θ_{2B}, ... θ_{10B} as inputs, and outputs the desired

joint trajectories for the Omni manipulator ϕ_{1md}, ϕ_{2md}, and ϕ_{3md}. The collected data set for the DL-FMA and IMA results from a simulation of 3600 seconds in which the robot is set to conduct different operations based on piece-wise randomized user inputs at randomized time intervals T_{switch}.

Table 3.2 shows the randomized variables used to generate realistic training data for the DL models. Note that variables with Gaussian distributions are centered with mean 0 and have their range represent the standard deviation.

Table 3.2: Random values assignment of critical variables for real-world data generation for DL mapping Algorithms

Distribution Type	Variable	Description	Range of Values
Uniformly Distributed Pseudorandom Integers	Foot	Start Foot (Left/Right)	[0 1] $\in Z$
	T_{switch}	Interval Between New User Inputs $\times 10^{-2}(s)$	[500, 2500] $\in Z$
Gaussian Distribution	x_0	Forward Kinematics Initial Conditions (cm)	0
	y_0		[-50, 50] $\in R$
	z_0		[-50, 50] $\in R$
	τ_h	Human Operator Control Input $\times 10^{-4}(Nm)$	[-5, 5] $\in R$
	δ_O	Omni Manipulator Disturbance $\times 10^{-6}(Nm)$	[-5, 5] $\in R$
	δ_B	Biped Robot Disturbance $\times 10^{-6}(Nm)$	[-5, 5] $\in R$

The generated training data then goes through a series of preprocessing steps. Given that the goal is to test the forecasting accuracy given a fixed unidirectional communication time delay of $dt_c = 0.1s$, preprocessing on the training input and output data must be applied. Given that $dt = 0.01s$, a time-shift is applied to the input I and output O data to ensure that the input at time step 1 corresponds to the output of time step 11. Table 3.3 clarifies the time-shifting operation of the training data. Note M is the size of the data set.

Following the time-shifting operation, the next preprocessing operation's objective is to extract pertinent information from the data to enhance the models' learning capabilities. Standardization is a popular first preprocessing step in machine learning because the data set is centered at 0 with unit standard deviation, which speeds up the training process. However, standardized data maintains its outlier data points.

Table 3.3: Time shifting preprocessing of training data

Discarded			Input Data						Discarded		
			I_1	I_2	...	I_{M-11}	I_{M-10}		I_{M-9}	...	I_M
Output Data											
O_1	...	O_{10}	O_{11}	O_{12}	...	O_{M-1}	O_M				

Data normalization is the answer to noisy data sets as it bounds the data in the range $[0, 1]$.

There are many ways to extract features from signals using various signal processing techniques. For example, the Fourier Transform is one of the most popular features extraction techniques, as it allows us to observe the dominant frequencies of a signal. In audio and other sequence-to-sequence machine learning applications, various feature extraction techniques such as the linear prediction coefficient, the line spectral frequencies, and the discrete wavelet transform. Below, we discuss how the Mel frequency spectrogram is used to extract pertinent information from the original data set.

To compute the Mel spectrogram of a signal, apply a window filter to the signal. A fast Fourier transform is used on the windowed signal to obtain the frequency-domain representation of the signal. The frequency-domain representation passes through a logarithmically-spaced bank of Mel filters to output the spectral values. These values are summed then concatenated with the other channels to form the Mel spectrogram.

3.3.2 Deep Learning Architectures

In this subsection, we go over the two proposed deep learning architectures used to predict the Cartesian-based FMA and IMA presented in Sections 3.1 and 3.2. The proposed deep learning architectures are 1- a deep LSTM architecture and 2- a deep CNN and LSTM architecture. Figure 3.3 shows the proposed architectures. The LSTM architecture is simple and is based on two cascaded LSTM networks. These networks, which are a subset of recurrent neural networks, are often used in time series forecasting and classification. The cascaded LSTM networks are separated by rectified linear unit layers which introduce nonlinearity to the model and helps it learn complex nonlinear data. The network then ends with a dropout layer, followed

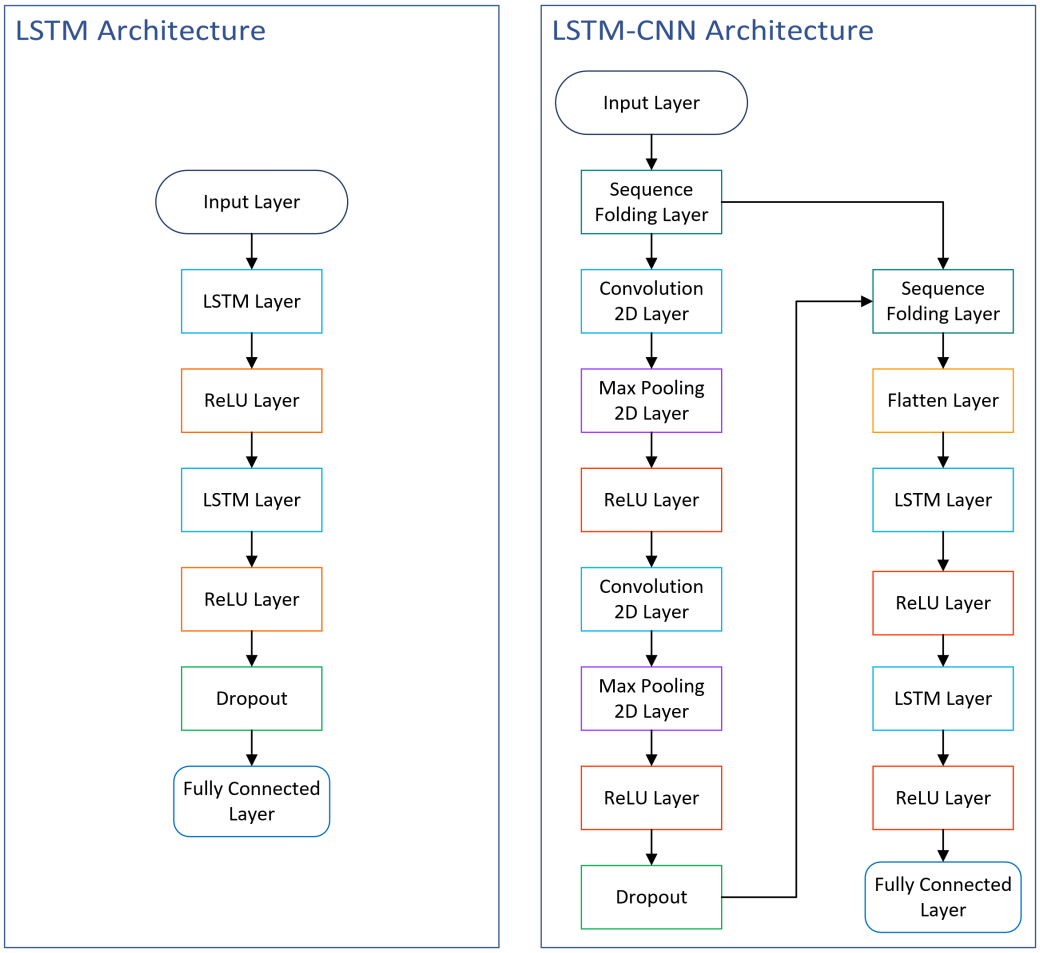


Figure 3.3: Deep learning architectures

by a fully connected layer to generate the output of the model.

In the second architecture, convolutional neural network layers are added to capture the long-term dependencies of the data. This architecture often used in video processing applications begins by folding the input data to input in the convolution layers. Two cascaded convolution layers with ascending number of filters separated by a ReLU activation function and a max-pooling layer are used to capture the sequential data's long-term dependencies. A dropout is applied before sending the CNN's output to the unfolding layer. The data is then flattened and fed to a cascaded LSTM network as in the first presented architecture.

Below, we describe the role of each layer of the DL architectures presented in Figure 3.3.

Input Layer

The input layer accepts the Mel spectogram computed during the preprocessing of the data.

LSTM Layer

A short-long term memory neural network is a type of recurrent neural network often used in sequence-to-sequence applications. Figure 3.4 shows the architecture of the LSTM network.

The LSTM network begins by assessing which information to keep or disregard using the forget gate. A weighted sum of the hidden state at the previous time step h_{t-1} and the input x_t with added bias is used to generate a normalized value between in the range $[0, 1]$ to determine whether the information is to be kept or discarded. A value of 1 maintains the information, and a 0 discards the information from the cell state. The forget gate is described as:

$$f_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_0) \tag{3.32}$$

Next, we defined the input gate and the tanh layer that decides which new information to store in the cell state. These are defined as:

$$i_t = \sigma(W_1 \cdot [h_{t-1}, x_t] + b_1) \tag{3.33}$$

$$z_t = \tanh\left(\sigma(W_2 \cdot [h_{t-1}, x_t] + b_2)\right) \tag{3.34}$$

Using Equations 3.32, 3.33, and 3.34, and given the previous cell state at time $t-1$, we define the new cell state as:

$$C_t = f_t C_{t-1} + i_t z_t \tag{3.35}$$

Then a filtered output based on the current cell state is computed. This is done by applying a tanh function on the current cell state and filter it using a sigmoid activation function:

$$o_t = \sigma(W_3 \cdot [h_{t-1}, x_t] + b_3) \tag{3.36}$$

$$h_t = o_t \tanh C_t \tag{3.37}$$

where W_0, W_1, W_2, W_3, and b_0, b_1, b_2, b_3 are the weights and bias of the LSTM network to be learned.

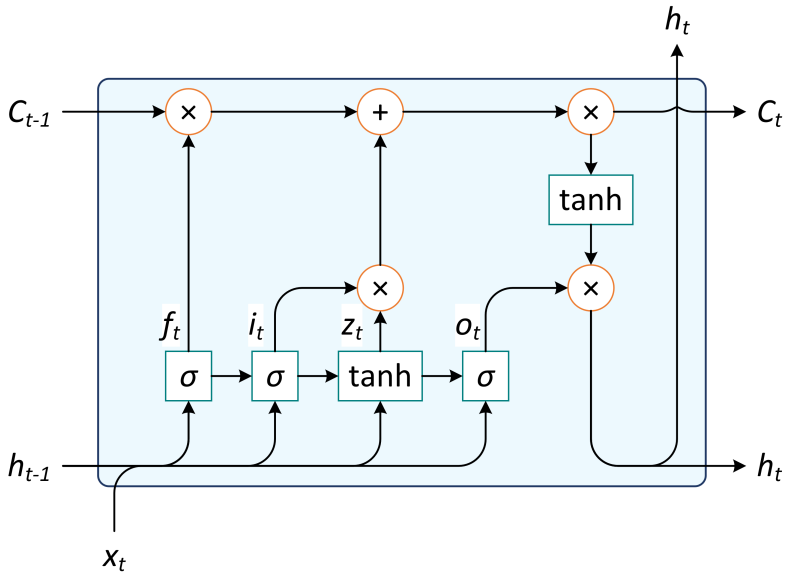


Figure 3.4: Long-short term memory architecture

ReLU Layer

The rectified linear unit is a commonly used activation function in deep learning models. It is a nonlinear piecewise function that allows the model to learn complex structures in the data. This function returns a 0 when the input i_{ReLU} is negative and returns the input when the input is positive. The output of the ReLU layer can be mathematically described as:

$$O_{ReLU} = \begin{cases} 0, & i_{ReLU} \leq 0 \\ i_{ReLU}, & i_{ReLU} > 0 \end{cases} \tag{3.38}$$

Dropout Layer

DL models' dropout layer is used between neural network layers to avoid data overfitting. During the training process, the dropout behaves as a filter given that it randomly creates a binary vector given a dropout probability ϵ_d and multiplies that vector with the output of the previous neural network layer.

2D Convolution Layer

The convolution layers in sequence-to-sequence deep learning models are often used to extract long-term dependencies from the data. These layers use filters to extract

features from the time series data. In DL models, the number of filters in each convolution layer is often in ascending order the deeper the layer is located in the model. This is because the first few layers capture high-level features from typically noisy input data, while deeper layers extract lower-level abstract features.

The convolution layers apply the cross-correlation operator to the input data and compares it to the trained filters to detect features. In signal processing, cross-correlation, which can be described as the inverse operation of convolution, is a measurement used to compare two or more time-series data and determine how and when they best match up with each other. The output of the cross-correlation operation O_{cnn} can be described as:

$$O_{cnn} = I_{cnn} \otimes F_{cnn} \tag{3.39}$$

where I_{cnn} is the input sequence and F_{cnn} the filter or kernel of fixed length. Expanding Equation 3.39 for discrete-time systems results in

$$O_{cnn}[i,j] = \sum_{m=-M}^{M} \sum_{n=-N}^{N} I_{cnn}[m,n] F_{cnn}[m+i, n+j] \tag{3.40}$$

where $2M$ and $2N$ is the filter size in the vertical and horizontal positions.

Given that the convolution layer receives for input the Mel spectogram 1D signal, the cross correlation operation of the CNN layer is simplified to:

$$O_{cnn}[i] = \sum_{m=-M}^{M} I_{cnn}[m] F_{cnn}[m+i] \tag{3.41}$$

2D Max Pooling Layer

The 2D max-pooling layer is typically used after convolutional layers to reduce over-fitting by filtering and generating an abstract form of its input. A filter size N_f with stride S_f is applied to the input and returns the maximum values at each filtered iteration, thus returning a map of the dominant features of the input. In other words, a max-pooling layer performs downsampling of the input data to enhance performance and avoid over-fitting.

Sequence Folding and Unfolding Layer

The sequence-folding layer converts the input sequence data into a batch of images that a convolutional layer can accept. Conversely, a sequence unfolding layer restores

the input data to its original form given its mini-batch size.

Flatten Layer

The flatten-layer collapses the input's spatial dimension into the channel dimension. In our case, as shown in Figure 3.3, the flatten layer is used in the LSTM-CNN architecture to collapse the data into 1-D data to input into the LSTM layer.

Fully Connected Layer

A fully connected layer is a feed-forward neural network. These layers are often at the end of DL models as they gather the previous layers' extracted data and reformulate it to a final output. A simple representation of a fully connected layer is as follows:

$$O_{FC} = W_0 I_{FC} + b_0 \tag{3.42}$$

where I_{FC}, O_{FC}, W_0, and b_0 are the input, output, weight matrix, bias vector of the feedforward neural network, respectively.

3.3.3 Training the Models

The training process optimizes the DL model to minimize a loss function. Given a set of hyper parameters, the optimizer iteratively modifies the weights and biases of the DL model to minimize the loss function. In sequence-to-sequence regression, the loss function is often dependent on some function of the error between the expected and true outputs. In this work, the Adam optimizer is used to minimize the root mean squared error (RMSE) and obtain a model representative of the FMA and IMA algorithms discussed in Sections 3.1 and 3.2.

The adaptive moment estimation algorithm is an optimizer that adapts its learning rate for each parameter. Assuming that the gradient g_t of the loss function is already computed, we can calculate the exponentially decaying average first and second moments m_t and v_t:

$$m_t = \zeta_1 \cdot m_{t-1} + (1 - \zeta_1) \cdot g_t \tag{3.43}$$

$$v_t = \zeta_2 \cdot v_{t-1} + (1 - \zeta_2) \cdot g_t^2 \tag{3.44}$$

where ζ_1 and ζ_2 represent the decay rates of the first and second moments m_t and v_t, respectively. Then, the bias-corrected moments $\hat{m}_t$ and $\hat{v}_t$ to achieve better estimates of the moving averages are computed:

$$\hat{m}_t = \frac{m_t}{1 - \zeta_1^t} \tag{3.45}$$

$$\hat{v}_t = \frac{v_t}{1 - \zeta_2^t} \tag{3.46}$$

Now, the parameters can be updated as follows:

$$\psi_t = \psi_{t-1} - \alpha_a \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon_a} \tag{3.47}$$

Note that the recommanded values for ζ_1, ζ_2 and ϵ are 0.9, 0.999 and 10^{-8}, respectively.

Chapter 4

Simulation Results

This chapter presents the simulation results obtained for the control system design explained in Chapter 2 and the two mapping algorithms presented in Chapter 3. The proposed algorithms, as shown by Figure 3.1 are tested via MATLAB simulation using the previously derived forward kinematic and system dynamics in Sections 2.1 and 2.3. In the following sections, we first explain the methods used to simulate the system, and then, we present the simulation results and a comparative study to alternative methods.

4.1 Control System Simulation Results

4.1.1 Methods

When simulating a control system, one must consider many parameters to ensure the system's stability in a real-world application. These parameters include various model uncertainties, external environmental disturbances, noisy sensor data, varying communication time delay between master and slave systems, and much more. This experimentation aims to test the control system's trajectory tracking error under model uncertainties and external disturbances. The simulation is conducted over 100 seconds with sensor sampling time of 10 ms. The system models are excited by a sinusoidal desired trajectory with frequency $F = 0.1$ Hz and amplitude 0.1π rad. The system dynamics equations defined in Section 2.3 are used to simulate and predict the desired position of future time steps.

To test the adaptability of the designed control system in the simulation, we introduce dynamic model errors $\tilde{M}(\theta), \tilde{C}(\dot{\theta}, \theta)$ and $\tilde{G}(\theta)$:

$$\tilde{M}(\theta) = M(\theta) - \hat{M}(\theta) \tag{4.1}$$

$$\tilde{C}(\dot{\theta}, \theta) = C(\dot{\theta}, \theta) - \hat{C}(\dot{\theta}, \theta) \tag{4.2}$$

$$\tilde{G}(\theta) = G(\theta) - \hat{G}(\theta) \tag{4.3}$$

where $\hat{M}(\theta), \hat{C}(\dot{\theta}, \theta)$, and $\hat{G}(\theta)$ are the estimated intertia, coriolis and gravitational parameters of the system. For simplicity, the estimated paramneters are defined as:

$$\hat{M}(\theta) = 0.8 \cdot M(\theta) \tag{4.4}$$

$$\hat{C}(\dot{\theta}, \theta) = 0.8 \cdot C(\dot{\theta}, \theta) \tag{4.5}$$

$$\hat{G}(\theta) = 0.8 \cdot G(\theta) \tag{4.6}$$

meaning that the error in the parameters is 20% of the true system parameters.

Additionally, an external disturbance applied as an impulse function with amplitude $0.001Nm$ for a 10-second duration was applied at 45 seconds after the start of the simulation. This impulse is defined as:

$$\delta = \begin{cases} 0, & t < 45 \\ 0.001, & 45 \leq t \leq 55 \\ 0, & t > 55 \end{cases} \tag{4.7}$$

The total uncertainty including external disturbances and other noise δ is defined as:

$$U = \tilde{M}(\theta)\ddot{\theta} + \tilde{C}(\theta, \dot{\theta}) + \tilde{G}(\theta) + \delta \tag{4.8}$$

Additionally, this section discusses the effects of the combined second order sliding mode controller with a nonlinear disturbance observer and an adaptive RBFNN feedforward compensator. More specifically, we test three cases:

1. The first is an unoptimized control architecture where the parameters are tuned manually.

2. In the second case, we isolate the SOSMC and tune it using the GA optimizer as described in Section 2.5. Then, the NDO and RBFNN feedforward compensator are added to the architecture after the optimization. Here, The tuning of the NDO and RBFNN feedforward compensator gains are done manually with trial and error.

3. Lastly, and unlike the second case, the SOSMC is optimized using GA while also considering the NDO and RBFNN compensators' effects during the optimization. Here the NDO and RBFNN compensators are also tuned manually.

4.1.2 Results and Analysis

The performance results by adding each component of the control system are described in Table 4.1 by running a 100 seconds simulation with model uncertainties and external disturbances as described in Equation 4.8. Here, the 3-DoF manipulator is used as a plant for its simplicity in demonstrating the efficacy of the proposed control architecture. Similar results were found when applied to the 10 DoF manipulator. The performance is measured using four metrics defined by the average of all joints errors: 1- the integral of the absolute error (IAE), 2- the integral of absolute error multiplied by time (ITAE), 3- the integral of the error squared (ISE), 4- and the root mean square error (RMSE). Figure 4.1 is a bar graph representing the RMSE performance of the control methods shown in Table 4.1 that is used to provide a better visual of the data.

The results obtained in Table 4.1 were obtained with the scalars described in Section 2.4. These scalar are defined in Table 4.2. Figures 4.2 through 4.5 show

Table 4.1: Control architectures performance results under three parameter tuning methods (lower is better)

	Unoptimized Controllers					
	FL	FL +NDO	SOSMC	SOSMC +RBFNN	SOSMC +NDO	SOSMC +RBFNN +NDO
IAE	1.0456e+03	27.7075	259.7389	178.9505	5.4025	5.2805
ITAE	5.3634e+04	1397.8	1.2195e+04	8800.9	261.1194	256.4761
ISE	1.0517e+04	7.2261	2261.1	1110.3	0.1904	0.1721
RMSE	0.1125	3.3e-03	0.0528	0.0353	5.1664e-04	4.8854e-04
GA Optimized SOSMC Without Compensators During Search						
IAE	-	-	253.9123	44.6766	12.2323	9.2932
ITAE	-	-	1.1922e+04	1.9874e+03	634.4539	451.9765
ISE	-	-	2.2276e+03	40.9906	0.9995	0.5622
RMSE	-	-	0.0522	0.0079	0.0011	9.1339e-04
GA Optimized SOSMC With Compensators During Search						
IAE	-	-	Unstable	1.3538e+03	1.5911	1.5581
ITAE	-	-	Unstable	7.1879e+04	70.2249	69.2794
ISE	-	-	Unstable	1.1383e+04	0.0473	0.0453
RMSE	-	-	Unstable	0.1294	2.6886e-04	2.6262e-04

Table 4.2: Control architecture model parameters

		Parameter Values		
	Parameter	Not Optimized	Optimized W/O Compensator	Optimized W/ Compensators
SOSMC	λ	2	$[1.003, 1.006, 1.824]^T$	$[8.621, 7.454, 9.463]^T$
	β	2.3	7.144	1.129
FL	K_1	10	-	-
NDO	K_D	150	150	150
RBFNN	μ_{RBF}	0	0	0
	μ_O	0.1	0.1	0.1
	K_{RBF}	11	11	11
	σ_{RBF}	0.05	0.05	0.05
	β_{RBF}	$[1, 0.1, 0.1]10^{-2}$	$[1, 0.1, 0.1]10^{-2}$	$[1, 0.1, 0.1]10^{-2}$

the simulation results of the 3-DoF manipulator when subjected to various uncertainties and disturbances over a period of 100s. More specifically, figure 4.2 shows the position tracking of the 3-DoF manipulator for each joint. Figure 4.3 shows the control input to the 3-DoF manipulator's actuators during the 100s simulation. The adaptive weights generate the RBFNN compensator's input Y_{RBF} over time shown by Figure 4.4. Figure 4.5 shows the NDO compensator $\hat{\tau}_d$ and the total disturbance and uncertainty computed by Equation 4.8.

From Table 4.1, we can make some critical observations. A GA-optimized control architecture performs better than an unoptimized control architecture, given that the search environment is identical to the testing environment. This can be seen where we saw a 1.1% improvement in RMSE from the GA-optimized SOSMC architecture without compensator during the search compared to the manually-tuned SOSMC. The same can also be concluded when we compare the manually-tuned SOSMC-RBFNN-NDO structure to the same structure but optimized with compensators, where we saw an improvement of 46%, from an RMSE of 4.8854×10^{-4} to 2.6262×10^{-4}. Alternatively, we can also conclude that modifying the testing environment from the optimization environment will often produce underperforming results. We can also conclude that a manually tuned SOSMC architecture performs better than a manually adjusted FL architecture. Even though this observation might be biased due to the

nature of trial and error in manual tuning, the FL algorithm is known to perform poorly in situations where the system dynamic model used in designing the control input is not identical to the true system dynamic model. This is the main reason why we see over 53% increased performance in an uncertain system simply by using the SOSMC. This control architecture is proven to be more robust to system uncertainties.

Furthermore, adding system compensators will typically always provide better trajectory tracking performance than without the addition of compensators. In this simulation, we specifically note that the RBFNN compensator performs less effectively when compared to the NDO. This is because the NDO uses the estimated dynamic model of the system combined with live state measurements of the system to compute an accurate estimate of uncertainty. As shown in Figure 4.5, the NDO computes a precise estimate of the disturbance and uncertainty experienced by the

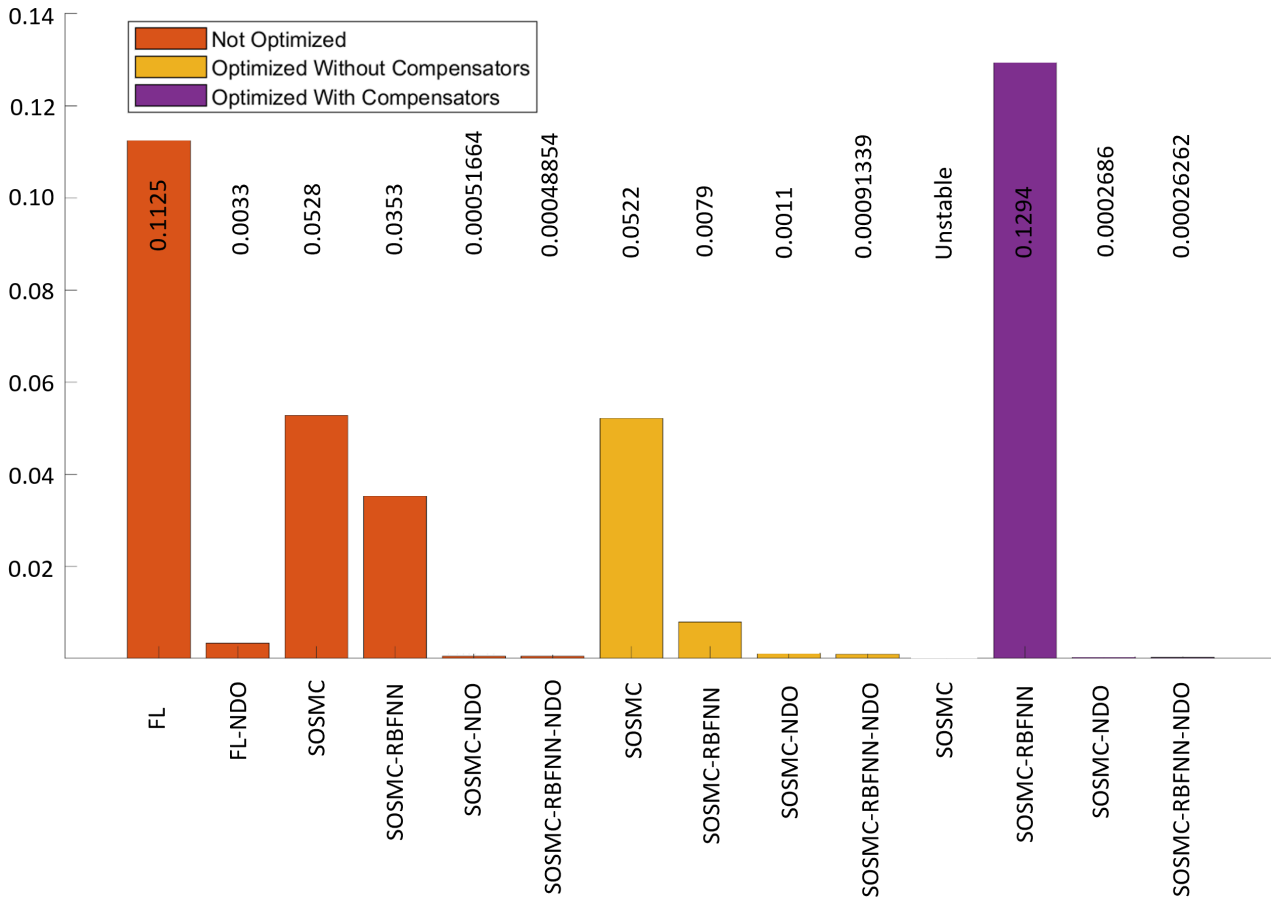


Figure 4.1: Control architectures performance results under three parameter tuning methods (lower is better)

system. This estimate is then applied as a compensated control input to cancel out the uncertainties. The uncertainty error with the NDO average out to 0, as shown in Figure 4.5.

Comparatively, the RBFNN compensator uses the desired trajectories as input and computes the weights adaptively based on the error state. The RBFNN is thus not as effective as the NDO in correcting model uncertainties and disturbances. Looking at Table 4.1, we see that the RBFNN compensator provides at least 33% performance improvement in RMSE. In contrast, the NDO compensator offers at least 97% performance improvement in RMSE, regardless of whether it was optimized or not. Better

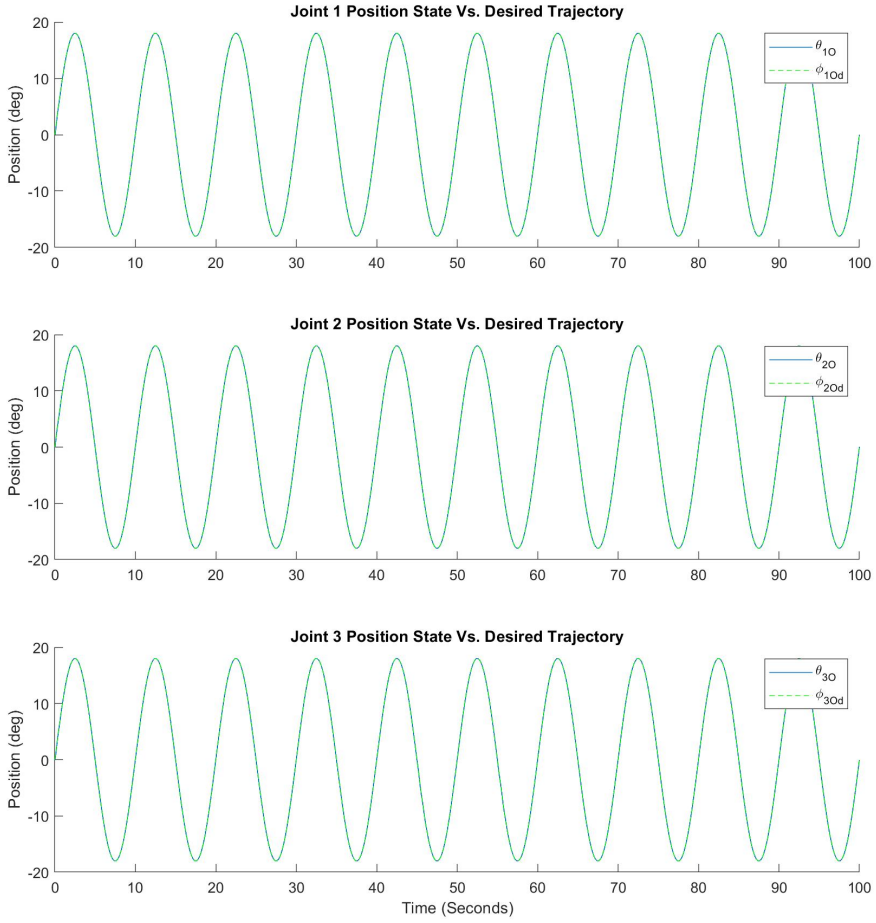


Figure 4.2: 3-DoF manipulator's position tracking of a 0.1 Hz sinusoid desired trajectory with model uncertainties and external disturbances

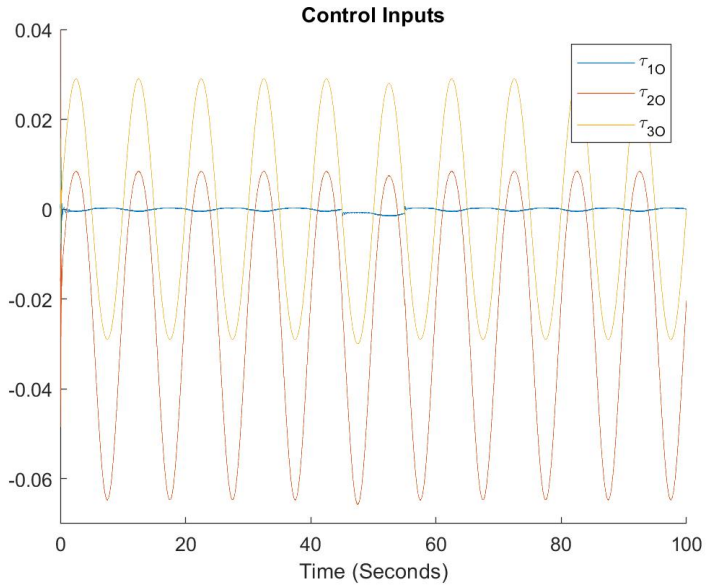


Figure 4.3: Applied control input torque of 3-DoF manipulator during a 0.1 Hz sinusoid desired trajectory with model uncertainties and external disturbances

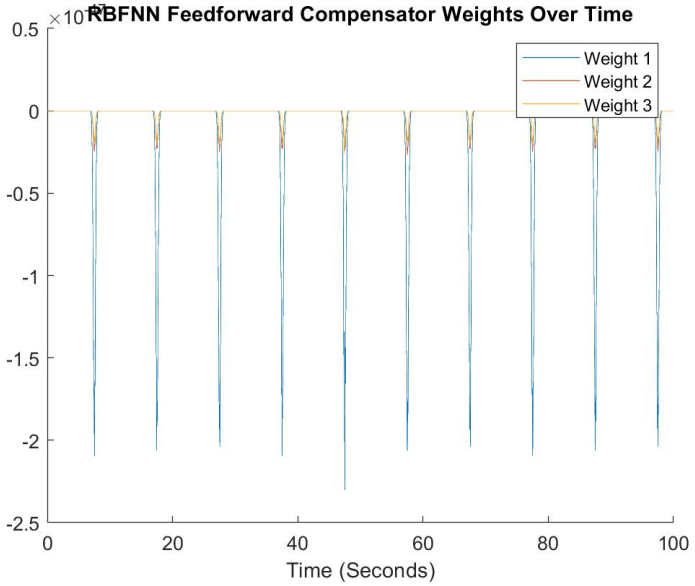


Figure 4.4: Adaptive weights of RBFNN feedforward compensator during a 0.1 Hz sinusoid system excitation with model uncertainties and external disturbances

performance can be extracted from these compensators if we tune the parameters identified in Table 4.2 using an optimizer.

Given this control architecture analysis, it is evident that the GA-optimized

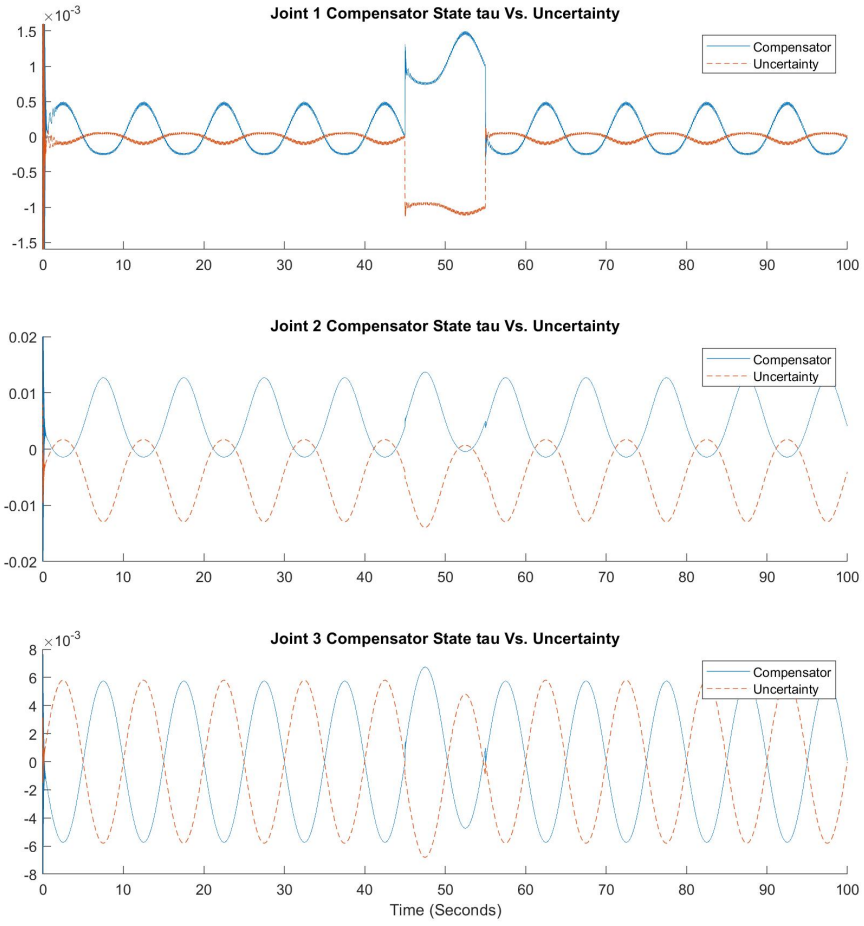


Figure 4.5: System uncertainties and disturbances Vs. NDO response during a 0.1 Hz Sinusoid System Excitation

SOSMC, RBFNN, and NDO architecture is the best for trajectory tracking. In bilateral teleoperation, having a responsive control system architecture is essential in reducing the communication delay effect. This control architecture was thus also applied to the 10-DoF biped robot due to its excellent trajectory tracking and response times.

4.2 Bilateral Teleoperation Mapping Simulation Results

4.2.1 Methods

This section tests the feasibility and practicality of the proposed bilateral teleoperation mapping algorithms between a 3-DoF haptic manipulator and a 10-DoF biped robot. In this simulation, we test the trajectory tracking of the master and slave systems under various operator inputs and test the system response under various communication time delays. The previously derived dynamics equations and control systems are used to simulate the system.

An iterative loop in MATLAB was designed to test the proposed system as per the architecture shown in Figure 3.1. The simulation is conducted over a period of 30 seconds with a sampling rate of 100 Hz and is tested with a unidirectional communication delay of 100ms and 500ms. The chosen communication latencies are typically within the range of delays using WiFi as a communication method between master and slave system. In the simulation, a force representing a user input is applied to the Omni manipulator dynamics, $\tau_h \neq 0$. The applied force changes at each 10 seconds interval to represent the three possible control cases described above.

4.2.2 Results and Analysis

Below, Figure 4.6 shows the results of the simulation. On the left, the states of the biped are shown at t=0s, t=15s, and t=30s, where we see the biped walk backward, stop, then go forward. The applied operator torque vector is:

$$\tau_h = \begin{cases} [0, -0.0005, -0.0005]^T & t \leq 10 \\ [0, 0, 0]^T & 10 \leq t \leq 20 \\ [0, 0.0005, 0.0005]^T & t \geq 20 \end{cases} \tag{4.9}$$

Yellow-to-magenta color-changing lines describe the start (yellow) and end (magenta) of the left and right foot simulation. Similarly, A progressively red-to-blue color-changing line demonstrates the path of the biped's CM from the start (red) to end (blue). The biped is represented in gray at the 15-second mark and a lighter gray at the 30-second mark to show the progression of the biped's position over time. On the right of Figure 4.6, we see the end-effector trajectory of the Omni manipulator.

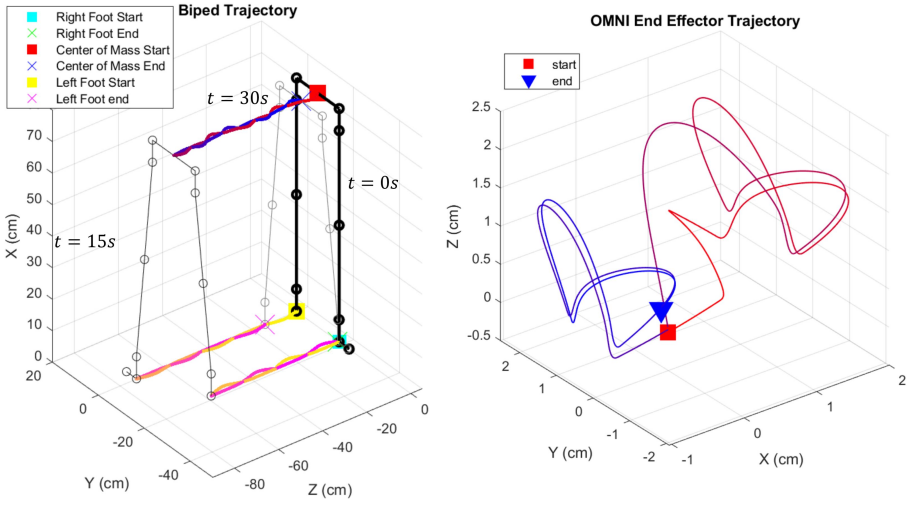


Figure 4.6: Trajectory of master (Omni manipulator) and slave (Biped) over a simulation period of 30 seconds. The user inputs three different forces at fixed 10 seconds intervals to test the forward, backward and stop motions. On the left, the trajectory of the bipedal robot. On the right, the trajectory of the Omni end effector. Color changing trajectory lines are used to emphasize path over time

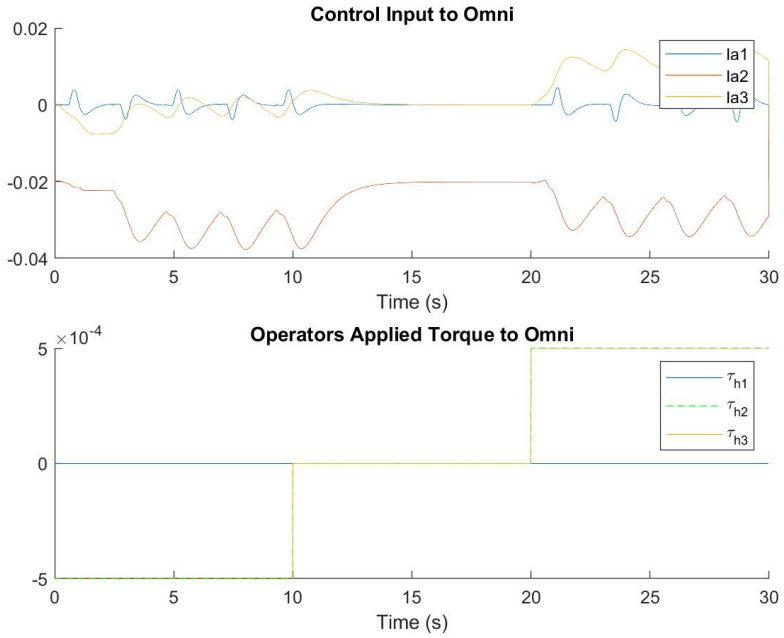


Figure 4.7: Control input and operator input on master robot during 30-seconds simulation period

Like for the biped's CM trajectory, a red-to-blue color-changing line demonstrates

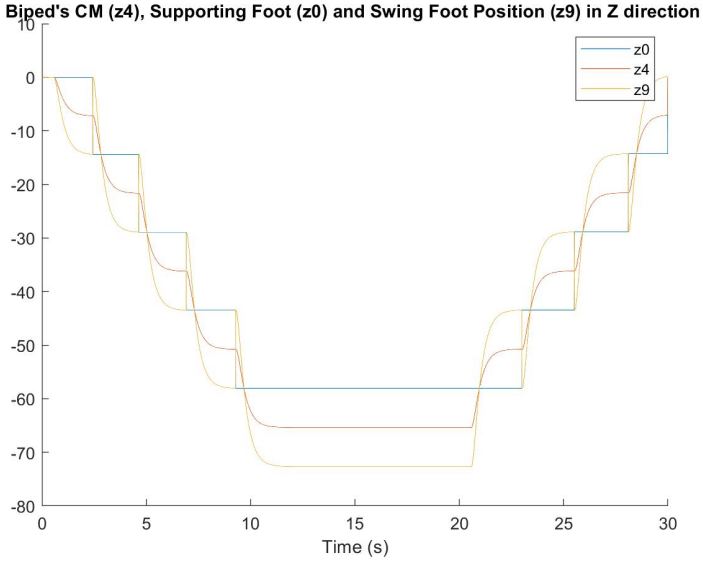


Figure 4.8: Biped position over time of the hip and ankle joints over time in the Z-Axis

the path progression from start to end of the simulation. Fig. 4 shows that it is possible to capture the biped left-to-right and up-and-down movements of the CM using the end effector of a 3-DoF manipulator. Also, note that the forward and backward motion of the end effector is only affected by the user input torque τ_h.

Figure 4.9 shows three plots representing the biped's CM trajectory and the Omni's end-effector trajectory in the X, Y, and Z axes. Figs. 5 and 6 are a continuation of Fig. 4 to demonstrate the matching trajectories of the master and slave robots more clearly. The haptic feedback of the haptic manipulator was scaled to better represent the trajectory tracking. This was scaled as:

$$X_s = (x_{EE} - l_{2m}) \cdot 10 \tag{4.10}$$

where x_{EE} is the end-effector position of the master robot in the X-axis and X_s is the scaled output. As mentioned previously, no haptic feedback is applied in the X-axis to allow the user to control the biped's walking gait forward or backward. As for the Y and Z direction, the Omni's scaled end-effector trajectory closely follows the biped's CM trajectory. Note that the Y-axis haptic feedback is multiplied by a -1 constant. The results shown in Figures 4.6 through 4.9 have constants $\delta_1 = -6$, $\delta_2 = -4$, and $\delta_3 = 0$. Figure 4.9 shows the response with a unidirectional communication delay

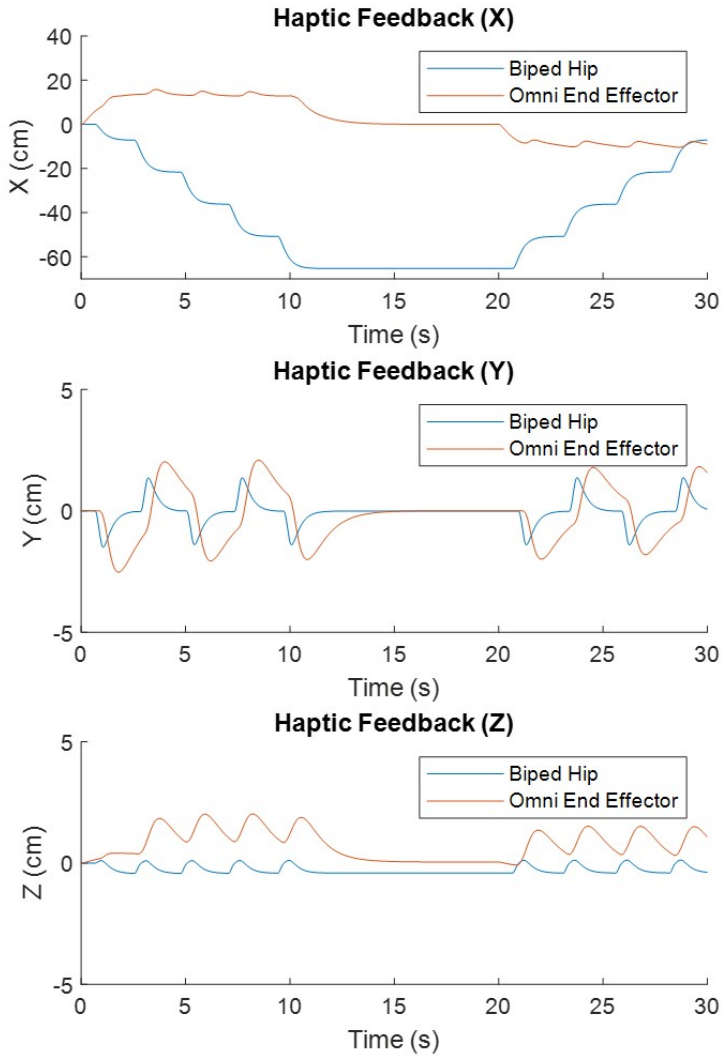


Figure 4.9: Trajectory of master and slave robots over a simulation period of 30 seconds. The user inputs three different forces at fixed 10-Seconds intervals to test the forward, backward and stop motions. The communication time delay is 0.1s.

of 0.1s, whereas Figure 4.10 shows the response of the system with a unidirectional communication delay of 0.5s.

4.3 Deep Learning Mapping Algorithms Simulation Results

The deep learning LSTM and CNN-LSTM models presented in Section 3.3 are trained then simulated. Before training, preprocessing of the training and testing data is

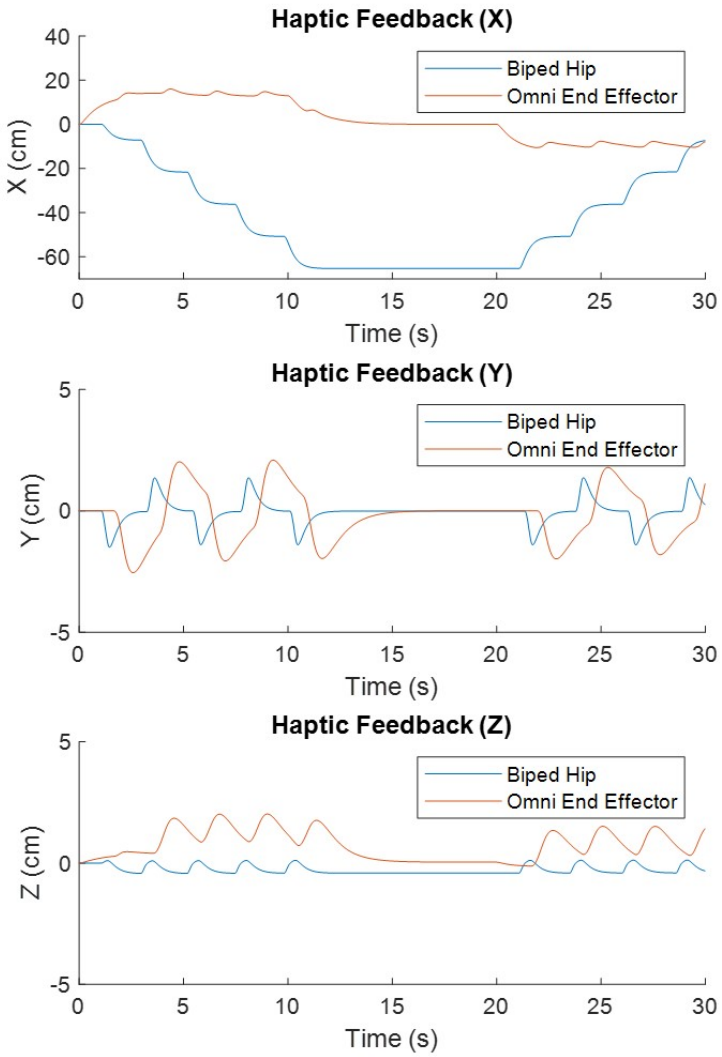


Figure 4.10: Trajectory of master and slave robots over a simulation period of 30 seconds. The user inputs three different forces at fixed 10-Seconds intervals to test the forward, backward and stop motions. The communication time delay is 0.5s.

done. First, we time-shift the data to account for a certain communication delay as shown in Table 3.3. The three joint angles of the haptic manipulator are then used as input to the deep learning model after the Mel spectrogram preprocessing. Figure 4.11 shows the Mel Spectrograms of the three manipulator joint angle signals. The Mel Spectograms highlight the dominant frequencies over time, essential when working with sequence-to-sequence applications. In the mentioned figure, the Y-axis

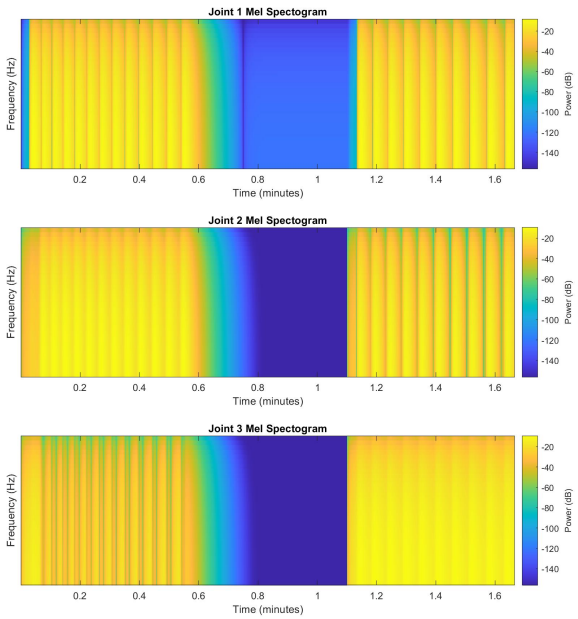


Figure 4.11: Mel spectograms of the three joint angle inputs. First 33 seconds, the biped walkds backwards. The next 33 seconds, the biped stops moving. The last 33 seconds, the biped walks forward

represents a range of frequencies from 0 to 50 Hz, which are separated by 32 horizontal bands. These bands are color-coded using an amplitude of -20 dB (in yellow) for dominant frequencies and -140 dB (in blue) for attenuated frequencies. The models were trained using the simulation data as per the parameters defined in Table 3.2 using an I7 4790k intel 4th generation processor and 16 GB of DDR3 RAM. Training parameters are set as shown below at the top half of Table 4.3. Figure 4.12 shows the results obtained by training the two CNN-LSTM models, and Table 4.3 quantifies tracking accuracy between models. We see that although the LSTM architecture has good tracking of the desired trajectories, the CNN-LSTM architecture has much better tracking of the desired trajectories. This is because the CNN layers capture long-term dependencies far better than simple LSTMs, which lack this property.

From the documented results in Table 4.3, we can make some important observations regarding the performance of the three proposed mapping algorithms. The LSTM architecture designed as per the parameters defined in Section 3.3.2 performs

Table 4.3: Training parameters and results

	LSTM	CNN-LSTM
Convolution 2D Layer 1 filter size	-	[3 1]
Convolution 2D Layer 1 number of filters	-	8
Max Pooling Layer 1 Pool Size	-	[5 1]
Convolution 2D Layer 2 filter size	-	[3 1]
Convolution 2D Layer 2 number of filters	-	16
Max Pooling Layer 2 Pool Size	-	[5 1]
Dropout	0.5	0.5
LSTM Layer 1 Number of Hidden Units	10	10
Fully Connected Layer number of Outputs	10	10
Epochs	1000	1000
Mini-Batch Size (N time steps)	5000	5000
Initial Learning Rate	0.01	0.01
Learning Rate Drop Factor	0.1	0.1
RMSE	0.098130	0.090004
Training Time (Seconds)	4909	25368

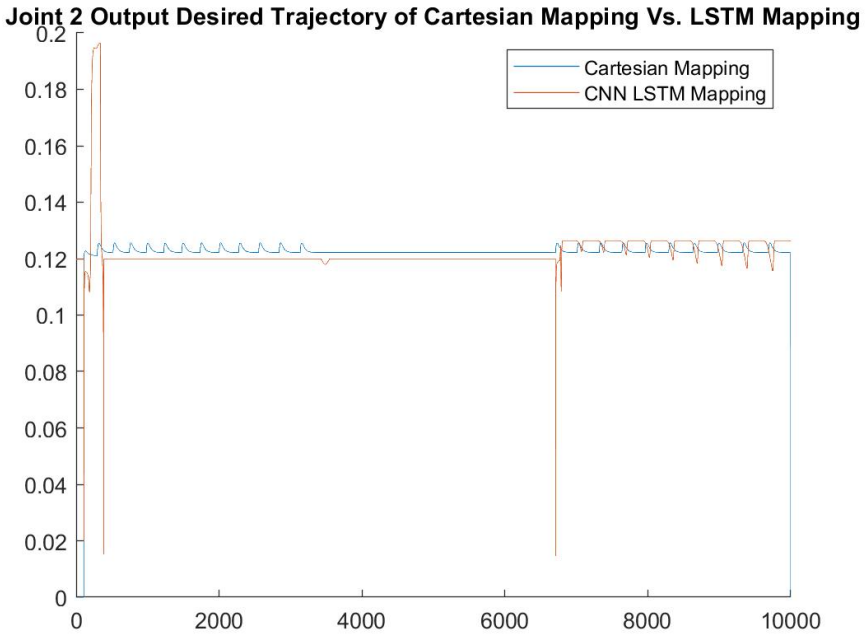


Figure 4.12: CNN-LSTM mapping output Vs. Cartesian mapping output

poorly with over 9% additional error in RMSE when compared to the CNN-LSTM architecture. However, the LSTM requires only 19.4% of the LSTM-CNN architecture's

training time.

In summary, although these architectures perform well on simulated training data, much more data is required to achieve safe and usable models. More specifically, even with the introduced randomness in Table 3.2, there still is not enough randomness in the data to model real-life situations. It is thus recommended to train these models using both simulated and experimental data to achieve the best results. Additionally, introducing deeper models may highly improve the tracking accuracy and may reduce the likelihood of errors in the trained model. The presented physic-approach using forward and Inverse kinematics algorithms, for now, remains the safer operation method. However, we acknowledge that deep-learning models can provide even greater depth in understanding the mapping between master and slave devices.

Chapter 5

Experimental Results

5.1 Experimentation Description

In this work, we use a semi-virtual setup to test the validity of the proposed mapping algorithm. This is done using the 10-DoF tonyPi biped as the slave and a virtual 3-DoF manipulator as the master. The 3-DoF manipulator controls the robot to walk backward. During that time, the Cartesian position of the biped's CM is tracked using joint angles and forward Kinematics (refer to Section 2.1). The tracked CM trajectory is stored, extracted from the biped's embedded system, and applied as the force feedback desired input to the 3-DoF manipulator.

Note that the experimental results do not reflect the simulation results because the slave robot used in the simulation (THORMANG3) is different from that used in experimentation (TonyPi). The objective of this experimentation is to confirm that the mapping algorithms can in fact provide accurate up-down left-right haptic feedback of the slave biped as it is walking.

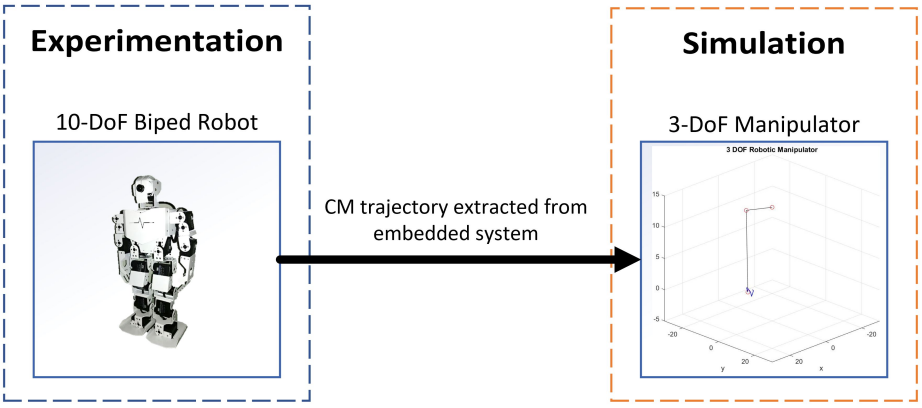


Figure 5.1: Semi-virtual experimentation setup

Table 5.1: Servo motors rest positions

Joint	Resting Position	Resting Position (deg)
1	498	89.64
2	388	69.84
3	498	89.64
4	593	106.74
5	499	89.82
6	498	89.64
7	611	109.98
8	501	90.18
9	405	72.90
10	499	89.82

5.2 Experimentation Setup

The sensory data is accessed at a minimum sampling frequency of 30 Hz in the biped code. This means that the biped robot also responds to changes in the slave system at a frequency of 30 Hz, plus the communication delay, which is set to 0 seconds in this case. The biped robot has a low-level joint motor PD controller that tracks the trajectory input. On a higher level, the biped's walking gait is controlled by a set of predefined trajectories similar yet, simpler to those described in Section 3.1.

5.3 Experimentation Results

In Figures 5.2 through 5.4, we display the noisy sensor data obtained from the TonyPi biped robot. This data originates from the joint angle motor encoders, which are processed through the forward kinematics to obtain the CM of the biped. The sensors return a specific value ranging between 0 to 1000, representing the motor shaft's position relative to its resting position (PRRP). The returned 0 to 1000 value is then processed to determine the relative angular position (RAP) in radians using the following equation:

$$RAP = PRRP \cdot \frac{\pi}{1000} \tag{5.1}$$

However, the relative angular position of the motor shaft does not represent the actual angular positions set by the frame in the Forward Kinematics Section. To

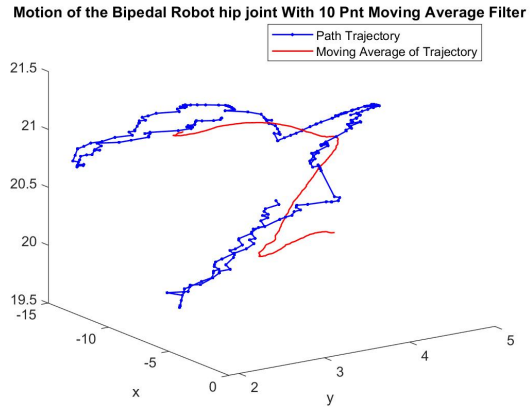


Figure 5.2: 3D trajectory of biped robot's CM when moving backwards

adhere to the mentioned coordinate frame in Section 2.1, before the start of the operation, we identify the biped's resting position (RP) when it is standing up straight and subtract it from the RAP to then compute the forward kinematics. In our case,

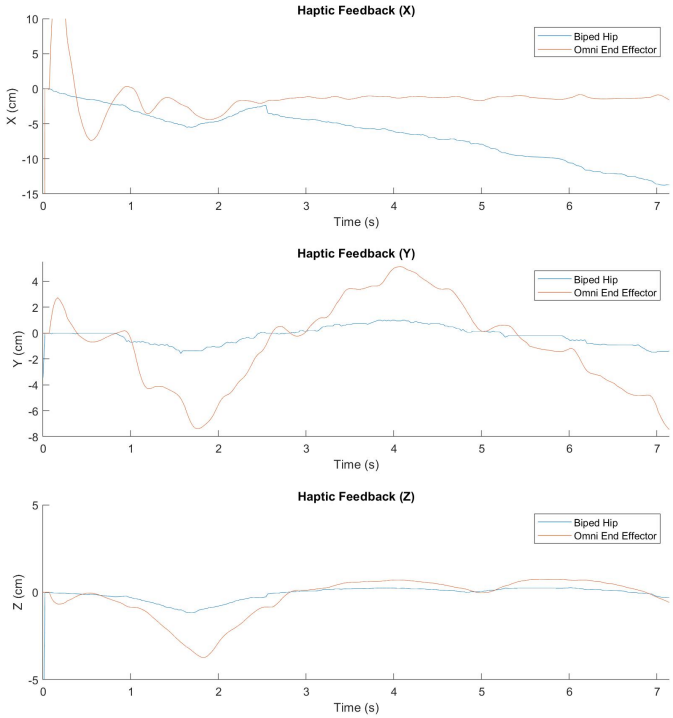


Figure 5.3: Master robot trajectory tracking of the slave's trajectory when walking backwards

Table 5.1 specifies the resting positions of each ten servo motor. The true joint angle as defined in the Forward Kinematics Section can then be derived as:

$$\theta_i = RAP_i - RP_i \cdot \frac{\pi}{1000} - \frac{\pi}{2} \tag{5.2}$$

In Figure 5.2, we display the CM trajectory of the biped robot. Here, we see the biped do the left to right and up and down movement as it completes three steps backward; This is a similar trajectory to what was obtained in the simulation results in Section 4.2.2. To emphasize the oscillating trajectory in the gait, a 10-point moving average was used to display a smoothed-out trajectory of the biped's CM. This 10-point moving-average filter is a basic low pass filter for the input of the master-side control system. This reduces high-frequency noise in data and provides smoother haptic feedback to the operator.

Figures 5.3 and 5.4 highlight the haptic manipulator's tracking of the biped's CM

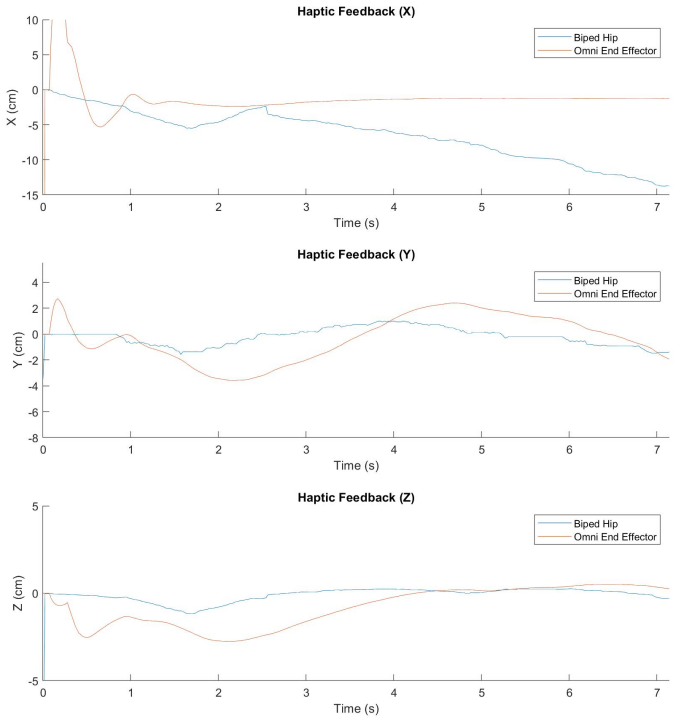


Figure 5.4: Master robot trajectory tracking of the 10-point moving average of the slave's trajectory when walking backwards

trajectory. In Figure 5.3, it can be observed that the manipulator's trajectory is highly affected by all the noise originating from the sensors. This, however, provides a much more accurate state of the biped during its operation. Conversely, in Figure 5.4, the haptic manipulator's trajectory is smoother. However, the 10-point average filtering only provides general information on the biped's state. It may not provide detailed information during extreme operation cases (i.e. when the biped falls and a big change in the amplitude of the CM's trajectory is observed).

Note that a significant overshoot occurred during the biped's first step during the experimentation. This overshoot typically happened at the 1.7 1.8 second mark, where it is suspected that an internal issue with the microcontroller may be causing this overshoot. The following steps during the experimentation occurred normally, and this is reflected when exporting the experimental data to apply it as the force feedback the simulated haptic manipulator.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this work, we proposed a novel mapping algorithm for bilateral teleoperation of bipedal robots using a haptic manipulator. This mapping algorithm was compared to two other deep learning-based algorithms and was tested and validated through simulation and semi-virtual experimentation.

The core of this work was the design of an adaptive and robust control system architecture to reduce external disturbances that might affect the performance of the proposed mapping algorithms. Through simulations, we determined that the ideal control system must compose a GA-optimized feed-forward compensator, a nonlinear disturbance observer, and a second-order sliding mode controller.

As for the Cartesian-based mapping algorithm, we successfully verified and confirmed our hypothesis - to teleoperate and control a biped using a manipulator. Many components were required to map desired trajectories from one domain to another due to the kinematic asymmetries. The forward mapping algorithm components responsible for controlling the bipedal robot included:

1. a computation of the forward kinematics of the master's end-effector
2. a bipedal step planning algorithm that uses the master's end-effector position as its input
3. a walking gait algorithm based on the 3D SLIP model
4. a computation of the inverse kinematics, which outputs the desired trajectories to the biped robot

Similarly, the inverse mapping algorithm components responsible for generating the haptic feedback included:

1. a forward kinematic algorithm to determine the biped robot's CM

2. an inverse kinematics algorithm that takes for input a scaled trajectory of the biped's CM to generate the desired trajectories of the manipulator

This Cartesian-based mapping algorithm was compared to two deep learning mapping algorithms. The chosen sequence-to-sequence models include 1- a long-short term memory deep neural network, 2- a combined convolutional neural network, and a long-short term memory neural network. Through the simulation, we observed that the CNN-LSTM architecture performed better than the LSTM at tracking the randomized simulation-generated training data of the Cartesian mapping algorithm. This is because CNNs better capture long-term dependencies, whereas LSTMs are more suited for short-term dependencies. However, although the deep learning models could track the Cartesian mapping algorithm, they could not maintain stability when applied to the biped-manipulator system. The Deep-learning model offers tremendous potential for adaptability, but even our randomized simulation data was not enough for this supervised-learning application to achieve stability in the system. We suspect that deeper models with more simulated and experimental data will achieve greater results.

Finally, the proposed Cartesian mapping was tested via semi-virtual experimentation using a 10-DoF biped robot and a 3-DoF haptic manipulator. The results shown in Section 5 show that the expected up-down and left-right movements of a biped's CM as it is walking is successfully tracked by the simulated haptic manipulator. This is precisely as shown in Section 4 where we display a similar trajectory tracking of the biped's CM. More experimentation is required to test the complete mapping algorithm. However, these preliminary results show great promise in the bilateral control of bipedal robots using manipulators.

6.2 Future Works

This new concept opens the door for more research in manipulator-biped bilateral teleoperation. In this thesis, we demonstrated the feasibility of the concept by allowing the user to control the biped robot in the forward and backward directions.

The control and haptic feedback can be expanded to support lateral walking, rotation, jumping, crouching, among many other motions. Distance and visual sensors can also generate haptic feedback when close to an obstacle, and velocity-based haptic feedback can also be used when the biped carries a load or encounters external forces such as wind drag. The number of math and physics equations needed for this application is significant, and errors are likely to occur. In the future, we aim to develop the deep learning approach further to reduce the computational requirements, increase robustness, and reduce the effect of the communication delay by predicting the user input.

Bibliography

[1] Peter F Hokayem and Mark W Spong. Bilateral teleoperation: An historical survey. *Automatica*, 42(12):2035–2057, 2006.

[2] Mahdi Tavakoli, Rajni V Patel, Mehrdad Moallen, and Arash Aziminejad. *Haptics for teleoperated surgical robotic systems*, volume 1. World Scientific, 2008.

[3] Thomas B Sheridan and William R Ferrell. Remote manipulative control with transmission delay. *IEEE Transactions on Human Factors in Electronics*, (1):25–29, 1963.

[4] William R Ferrell. Delayed force feedback. *Human factors*, 8(5):449–455, 1966.

[5] Fumio Miyazaki, S Matsubayashi, T Yoshimi, and Suguru Arimoto. A new control methodology toward advanced teleoperation of master-slave robot systems. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 997–1002. IEEE, 1986.

[6] Enrique Cano Marın. Recent advances in research teleoperation, telepresence and virtual reality.

[7] Xiong Yang, Haotian She, Haojian Lu, Toshio Fukuda, and Yajing Shen. State of the art: bipedal robots for lower limb rehabilitation. *Applied Sciences*, 7(11):1182, 2017.

[8] Miomir Vukobratović and Branislav Borovac. Zero-moment point—thirty five years of its life. *International journal of humanoid robotics*, 1(01):157–173, 2004.

[9] Mariano Garcia, Anindya Chatterjee, Andy Ruina, and Michael Coleman. The simplest walking model: stability, complexity, and scaling. 1998.

[10] Arthur D Kuo. A simple model of bipedal walking predicts the preferred speed–step length relationship. *J. Biomech. Eng.*, 123(3):264–269, 2001.

[11] Hartmut Geyer, Andre Seyfarth, and Reinhard Blickhan. Spring-mass running: simple approximate solution and application to gait stability. *Journal of theoretical biology*, 232(3):315–328, 2005.

[12] Elvedin Kljuno and Robert L Williams. Humanoid walking robot: modeling, inverse dynamics, and gain scheduling control. *Journal of Robotics*, 2010, 2010.

[13] Zhi Liu, Liyang Wang, CL Philip Chen, Xiaojie Zeng, Yun Zhang, and Yaonan Wang. Energy-efficiency-based gait control system architecture and algorithm for biped robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):926–933, 2011.

[14] Hongbo Zhu, Minzhou Luo, Tao Mei, Jianghai Zhao, Tao Li, and Fayong Guo. Energy-efficient bio-inspired gait planning and control for biped robot based on human locomotion analysis. *Journal of Bionic Engineering*, 13(2):271–282, 2016.

[15] Seung-Suk Ha, Jae-Hyoung Yu, Young-Joon Han, and Hern-Soo Hahn. Natural gait generation of biped robot based on analysis of human's gait. In *2008 International Conference on Smart Manufacturing Application*, pages 30–34. IEEE, 2008.

[16] Mohammadali Shahriari and Amir A Khayyat. Gait analysis of a six-legged walking robot using fuzzy reward reinforcement learning. In *2013 13th Iranian Conference on Fuzzy Systems (IFSC)*, pages 1–4. IEEE, 2013.

[17] Jae Won Kho, Dong Cheol Lim, and Tae Yong Kuc. Implementation of an intelligent controller for biped walking robot using genetic algorithm. In *2006 IEEE International Symposium on Industrial Electronics*, volume 1, pages 49–54. IEEE, 2006.

[18] Shouyi Wang, Jelmer Braaksma, Robert Babuska, and Daan Hobbelen. Reinforcement learning control for biped robot walking on uneven surfaces. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 4173–4178. IEEE, 2006.

[19] Abdulrahman Albakri, Chao Liu, and Philippe Poignet. Stability and performance analysis of three-channel teleoperation control architectures for medical applications. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 456–462. IEEE, 2013.

[20] Sung Min Yoon, Won Jae Kim, Min Cheol Lee, and Woo Hyeok Choi. Bilateral control for haptic laparoscopic surgery robot. In *IEEE ISR 2013*, pages 1–5. IEEE, 2013.

[21] Hiroyuki Tanaka, Kouhei Ohnishi, Hiroaki Nishi, Toshikazu Kawai, Yasuhide Morikawa, Soji Ozawa, and Toshiharu Furukawa. Implementation of bilateral control system based on acceleration control using fpga for multi-dof haptic endoscopic surgery robot. *IEEE Transactions on Industrial Electronics*, 56(3):618–627, 2008.

[22] Lei Li, Qing Wei, Zhilin Hou, and Lei Zhao. Design and realization of the experimental platform of space robot bilateral teleoperation system. In *Proceedings of the 30th Chinese Control Conference*, pages 3968–3972. IEEE, 2011.

[23] Takashi Imaida, Yasuyoshi Yokokohji, Toshitsugu Doi, Mitsushige Oda, and T Yoshikwa. Ground-space bilateral teleoperation experiment using ets-vii robot arm with direct kinesthetic coupling. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 1, pages 1031–1038. IEEE, 2001.

[24] Mariela Serna, Luis G Garcia-Valdovinos, Tomas Salgado-Jimenez, and Manuel Bandala-Sanchez. Bilateral teleoperation of a commercial small-sized underwater vehicle for academic purposes. In *OCEANS 2015-MTS/IEEE Washington*, pages 1–5. IEEE, 2015.

[25] Weidong Liu, Jianjun Zhang, et al. Fuzzy impedance and sliding mode bilateral control in underwater ratio teleoperation based on observer. In *OCEANS 2016-Shanghai*, pages 1–7. IEEE, 2016.

[26] Georgeta Bauer, Ya-Jun Pan, and Henghua Shen. Adaptive impedance control in bilateral telerehabilitation with robotic exoskeletons. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 719–725. IEEE, 2020.

[27] Yasuhiro Ishiguro, Tasuku Makabe, Yuya Nagamatsu, Yuta Kojio, Kunio Kojima, Fumihito Sugai, Yohei Kakiuchi, Kei Okada, and Masayuki Inaba. Bilateral humanoid teleoperation system using whole-body exoskeleton cockpit tablis. *IEEE Robotics and Automation Letters*, 5(4):6419–6426, 2020.

[28] Jessica Lanini, Toshiaki Tsuji, Peter Wolf, Robert Riener, and Domen Novak. Teleoperation of two six-degree-of-freedom arm rehabilitation exoskeletons. In *2015 IEEE International Conference on Rehabilitation Robotics (ICORR)*, pages 514–519. IEEE, 2015.

[29] Joao Rebelo, Thomas Sednaoui, Emiel Boudewijn Den Exter, Thomas Krueger, and Andre Schiele. Bilateral robot teleoperation: A wearable arm exoskeleton featuring an intuitive user interface. *IEEE Robotics & Automation Magazine*, 21(4):62–69, 2014.

[30] Xi Chen, Satoshi Nishikawa, Kazutoshi Tanaka, Ryuma Niiyama, and Yasuo Kuniyoshi. Bilateral teleoperation system for a musculoskeletal robot arm using a musculoskeletal exoskeleton. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2734–2739. IEEE, 2017.

[31] Anais Brygo, Ioannis Sarakoglou, Arash Ajoudani, NG Hernandez, Giorgio Grioli, M Catalano, Darwin G Caldwell, and N Tsagarakis. Synergy-based interface for bilateral tele-manipulations of a master-slave system with large asymmetries. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4859–4865. IEEE, 2016.

[32] Georgeta Bauer and Ya-Jun Pan. Review of control methods for upper limb telerehabilitation with robotic exoskeletons. *Ieee Access*, 2020.

[33] Fangping Yang, Hongyi Li, and Yuechao Wang. Wave-transformation-based control law for teleoperation with large time-varying delays. In *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1610–1614. IEEE, 2012.

[34] Carlo Benedetti, Matteo Franchini, and Paolo Fiorini. Stable tracking in variable time-delay teleoperation. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, volume 4, pages 2252–2257. IEEE, 2001.

[35] Jee-Hwan Ryu, Carsten Preusche, Blake Hannaford, and Gerd Hirzinger. Time domain passivity control with reference energy following. *IEEE Transactions on Control Systems Technology*, 13(5):737–742, 2005.

[36] Zhong Shi, Xuexiang Huang, Qian Tan, and Tianjian Hu. Fractional-order pid control method for space teleoperation. In *2015 IEEE International Conference on Multimedia Big Data*, pages 216–219. IEEE, 2015.

[37] A Roushandel, A Khosravi, and A Alfi. Bilateral control of teleoperation systems via robust pid controllers based on lmi. In *The 3rd International Conference on Control, Instrumentation, and Automation*, pages 16–21. IEEE, 2013.

[38] Fanghao Huang, Wei Zhang, Zheng Chen, Jianzhong Tang, Wei Song, and Shiqiang Zhu. Rbfnn-based adaptive sliding mode control design for nonlinear bilateral teleoperation system under time-varying delays. *IEEE Access*, 7:11905–11912, 2019.

[39] Jong Hyeon Park and Hyun Chul Cho. Sliding-mode controller for bilateral teleoperation with varying time delay. In *1999 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (Cat. No. 99TH8399)*, pages 311–316. IEEE, 1999.

[40] Ali Shahdi and Shahin Sirouspour. Adaptive/robust control for time-delay teleoperation. *IEEE Transactions on Robotics*, 25(1):196–205, 2009.

[41] Tianlin Zhu and Yijun Zhang. Observer-based control of bilateral teleoperation with time delay. In *2018 5th International Conference on Information Science and Control Engineering (ICISCE)*, pages 859–863. IEEE, 2018.

[42] Kenji Natori, Toshiaki Tsuji, Kouhei Ohnishi, Ales Hace, and Karel Jezernik. Time-delay compensation by communication disturbance observer for bilateral teleoperation under time-varying delay. *IEEE Transactions on Industrial Electronics*, 57(3):1050–1062, 2009.

[43] B Aboutalebian, HA Talebi, and AA Suratgar. Nonlinear disturbance observer based adaptive control for nonlinear teleoperation systems. In *2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM)*, pages 091–095. IEEE, 2015.

[44] Zhangfeng Ju, Chenguang Yang, Zhijun Li, Long Cheng, and Hongbin Ma. Teleoperation of humanoid baxter robot using haptic feedback. In *2014 International*

Conference on Multisensor Fusion and Information Integration for Intelligent Systems (MFI), pages 1–6. IEEE, 2014.

[45] Zheng Chen, Shuifeng Yan, Mingxing Yuan, Bin Yao, and Jinfei Hu. Modular development of master-slave asymmetric teleoperation systems with a novel workspace mapping algorithm. *IEEE Access*, 6:15356–15364, 2018.

[46] Septimiu E Salcudean, NM Wong, and Ralph L Hollis. Design and control of a force-reflecting teleoperation system with magnetically levitated master and wrist. *IEEE Transactions on Robotics and Automation*, 11(6):844–858, 1995.

[47] Xiao Gao, Joao Silvério, Emmanuel Pignat, Sylvain Calinon, Miao Li, and Xiaohui Xiao. Motion mappings for continuous bilateral teleoperation. *IEEE Robotics and Automation Letters*, 6(3):5048–5055, 2021.

[48] Rui Li, Hongyu Wang, and Zhenyu Liu. Survey on mapping human hand motion to robotic hands for teleoperation. *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.

[49] Liang Yan, Xiaoshan Gao, Xiongjie Zhang, and Suokui Chang. Human-robot collaboration by intention recognition using deep lstm neural network. In *2019 IEEE 8th International Conference on Fluid Power and Mechatronics (FPM)*, pages 1390–1396. IEEE, 2019.

[50] Matteo Macchini, Fabrizio Schiano, and Dario Floreano. Personalized telerobotics by fast machine learning of body-machine interfaces. *IEEE Robotics and Automation Letters*, 5(1):179–186, 2019.

[51] Mengxi Li, Dylan P Losey, Jeannette Bohg, and Dorsa Sadigh. Learning user-preferred mappings for intuitive robot control. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10960–10967. IEEE, 2020.

[52] Jerome Connor, Les E Atlas, and Douglas R Martin. Recurrent networks and narma modeling. In *Advances in neural information processing systems*, pages 301–308, 1992.

[53] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[54] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[55] Akara Supratak, Hao Dong, Chao Wu, and Yike Guo. Deepsleepnet: A model for automatic sleep stage scoring based on raw single-channel eeg. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(11):1998–2008, 2017.

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[57] Neo Wu, Bradley Green, Xue Ben, and Shawn O'Banion. Deep transformer models for time series forecasting: The influenza prevalence case. *arXiv preprint arXiv:2001.08317*, 2020.

[58] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. *arXiv preprint arXiv:2010.02803*, 2020.

[59] Changan Chen, Yuejiang Liu, Sven Kreiss, and Alexandre Alahi. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6015–6022. IEEE, 2019.

[60] Tayfun Abut and Servet Soygüder. Haptic industrial robot control and bilateral teleoperation by using a virtual visual interface. In *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4. IEEE, 2018.

[61] Hongseok Choi, Jinlong Piao, Eui-Sun Kim, Jinwoo Jung, Eunpyo Choi, Jong-Oh Park, and Chang-Sei Kim. Intuitive bilateral teleoperation of a cable-driven parallel robot controlled by a cable-driven parallel robot. *International Journal of Control, Automation and Systems*, pages 1–14, 2020.

[62] Jun Ueda and Tsuneo Yoshikawa. Force-reflecting bilateral teleoperation with time delay by signal filtering. *IEEE Transactions on Robotics and Automation*, 20(3):613–619, 2004.

[63] Takashi Imaida, Yasuyoshi Yokokohji, Toshitsugu Doi, Mitsushige Oda, and Tsuneo Yoshikawa. Ground-space bilateral teleoperation of ets-vii robot arm by direct bilateral coupling under 7-s time delay condition. *IEEE Transactions on Robotics and Automation*, 20(3):499–511, 2004.

[64] Takayuki Osa, Satoshi Uchida, Naohiko Sugita, and Mamoru Mitsuishi. Hybrid rate—admittance control with force reflection for safe teleoperated surgery. *IEEE/ASME Transactions on Mechatronics*, 20(5):2379–2390, 2015.

[65] Jing Guo, Chao Liu, and Philippe Poignet. Enhanced position-force tracking of time-delayed teleoperation for robotic-assisted surgery. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 4894–4897. IEEE, 2015.

[66] Francheska B Chioson, Noelle Marie D Espiritu, Francisco Emmanuel T Munsayac, Ryan Christoper R Dajay, Michael Bryan S Santos, Renann G Baldovino, and Nilo T Bugtai. Implementation of a bilateral teleoperation haptic feedback

controls to robotic minimally invasive surgery. In *2020 IEEE 12th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, pages 1–5. IEEE, 2020.

[67] Mischa Kim. Euler-lagrange tool package.

[68] John H Holland. Genetic algorithms and adaptation. In *Adaptive Control of Ill-Defined Systems*, pages 317–333. Springer, 1984.