MACHINE LEARNING OF HAPTIC OBJECTS AND REPRODUCTION FOR VIRTUAL REALITY

Praveena Wimarshani Dewapura

(218009L)

Degree of Master of Science

Department of Electrical Engineering

University of Moratuwa

Sri Lanka

September 2022

MACHINE LEARNING OF HAPTIC OBJECTS AND REPRODUCTION FOR VIRTUAL REALITY

Praveena Wimarshani Dewapura

(218009L)

Thesis/Dissertation submitted in partial fulfillment of the requirements for the degree Master of Science (by Research)

Department of Electrical Engineering

University of Moratuwa

Sri Lanka

September 2022

DECLARATION

I declare that this is my own work, and this thesis/dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis/dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: UOM Verified Signature Date: 26.09.2022

The above candidate has carried out research for the Masters/MPhil/PhD thesis/ dissertation under my supervision.

Name of the supervisor: Dr.A.M.H.S.Abeykoon

UOM Verified Signature 31/3/2023

Signature of the supervisor: Date:

ABSTRACT

Humans interact with machines extensively through our auditory and visual senses, but they frequently ignore their most trusted sense: touch. However, the sense of touch has tremendous potential in almost all fields, including medicine, exploration, industrial robots, and gaming. Haptics, or the science of touch, enables humans to not only remove the barriers to achieve realization in virtual world, but also to perform a wide range of real-world manipulation tasks.

Unlike visual and auditory senses, sense of touch is bilateral. Thus, realistic haptic feedback takes utmost importance to achieve realization in the virtual world and to enhance human performance in the real world. Prior studies have used an environment model to reproduce the haptics sensation from the environment as if it's from the real environment. Most studies have employed conventional spring damper model to model the environment model and motion parameters were considered as the factors affecting for force response. However, the traditional spring damper model doesn't reflect the actual object. Furthermore, the influence of learned force on reproduction requires special consideration, but haptics studies mostly consider motion data. However, there can be several factors affecting the recreation of haptic feedback. Thus, it is essential to analyze these factors to precisely reproduce an object in haptic dimension. Most studies have utilized force/torque sensors despite their shortcomings such as narrow bandwidth, signal noise, complicity, non-collocation, and instability. However, robust sensorless force/torque control over a wide bandwidth can be achieved using observer techniques and Disturbance Observer (DOB) and Reaction Force Observer (RFOB) are primarily used to get force measurements. AI enables computers to utilize vast quantities of data and employ their acquired intelligence to arrive at optimal conclusions and uncover insights in mere fractions of the time it would take for humans to do the same. Thus, recent technological studies have focused on using AI techniques to analyze larger and more complex data sets to achieve accurate and faster results. Thus, incorporating AI with haptics allows seamless integration with virtual reality and tune this technology to achieve precise responses.

Thus, this study focused on introducing machine learning and deep learning based vivid force sensation reproduction through a virtual model which replicates the actual environment. The information needed is abstracted through Disturbance Observer (DOB) and Reaction Force Observer (RFOB) based sensorless approach. Furthermore, statistical analysis was conducted on data to identify important features affecting the target value of force response.

Keywords *— Haptic interaction, Force response, Disturbance Observer, Virtual reality, force response, motion parameters, Artificial Intelligence, correlation, Principal Components Analysis (PCA), Random Forest, Haptic object Reproduction, RMSE.*

ACKNOWLEDGEMENT

This research work would not have been possible without the support of many people. I'm delighted to express my thanks and gratitude to my supervisor Dr.A.M.H.S.Abeykoon of the Department of Electrical Engineering, University of Moratuwa for constant guidance, support and encouragement from the very beginning to the end. I would like to acknowledge all the academic staff members of the Department of Electrical Engineering of the University of Moratuwa for their valuable suggestions, comments, and assistance which were beneficial to drive the project towards its objective.

I'm grateful for the financial support provided by the Senate Research Committee (SRC) of University of Moratuwa and administrative support by the Faculty of Graduate Studies to conduct my research.

I would like to thank the technical officers, including Mr. M.W.D. Wasantha and other support staff of Robotics for the assistance they have given to perform laboratory experiments, fabrication, and part designing.

Moreover, I would like to extend my gratitude to my family for their encouragement, understanding, and patience throughout my academic pursuit. Finally, I am grateful to my colleagues and friends for showing interest in my work and giving constructive ideas to lead the success of the research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application program interface
BP	Back Propagation
DNN	Deep Neural Network
DOB	Disturbance Observer
I/O	Input / Output
LSI	Large-Scale Integration
LSTM	Long short-term memory networks
MAE	Mean Absolute Error
MCS	Motion copying system.
ML	Machine Learning
MSE	Mean Squared Error
NN	Neural Network
PC	Principal component
PCA	Principal Component Analysis
PMML	Predictive Model Markup Language
RFOB	Reaction Force Observer
RMSE	Root Mean Squared Error

RNN	Recurrent Neural Network
SDK	Software Development Kits
SGD	Stochastic gradient descent
SVR	Support Vector Regression
XML	Extensible markup language

LIST OF APPENDICES

NOMENCLATURE

M	Motor mass
M_n	The nominal value of motor mass
K_f	Motor force constant
K_{fn}	The nominal value of force constant
I_a^{ref}	Motor current
B	Viscosity coefficient
g_{dis}	Cut off frequency of DOB
g_{rec}	Cut off frequency of RFOB
x	Compression depth
$\dot{x}$	Velocity
F_m	Generated motor force
F_{dis}	Disturbance force
F_{ext}	Reaction force
F_{int}	Interactive force
F_f	Static friction
K_p	Proportional gain of the controller
g_v	Velocity filter constant
g_a	Acceleration filter constant
F_{cmd}	Force command

F_{res}	Force response
$\hat{F}_{dis}$	Estimated disturbance force
$\hat{F}_{ext}$	Estimated reaction force
$\ddot{x}$	Acceleration
c	Cycle No.
d	Permanent Deformation
$A^{F_{res}.x}$	Area F_{res} vs x curve
k	Stiffness coefficient
b	Damping coefficient
F_e	Reaction force from environment
C_p	Proportional gain of the position controller
C_d	Derivative gain of the position controller
C_i	Integral gain of the position controller
x_{ref}	Position reference command by operator
x_{err}	Compression depth error
dx_{err}	Change in compression depth error
$\int x_{err}$	Sum of compression depth errors
x_{err}^{pre}	Previous compression depth error
$\left(\int x_{err}\right)^{pre}$	Previous sum of compression depth errors

1. INTRODUCTION

In the modern era, almost all people interact with machines through auditory and visual senses, yet they often neglect the importance of the sense of touch which they trust most. Besides, only visual and auditory senses are not sufficient for a precise realization in virtual reality. Thus, scientists have taken a keen interest in recreating the sense of touch to achieve realization in the virtual world and to enhance the performance of a vast variety of real-world manipulation tasks [1]. Consequently, the latest studies on haptics consider AI technology to grant the perceptual capabilities of the touch sense to the new generation of robots with intelligence.

Haptics information involves action and reaction. Thus, it is bilateral. Feedback is often perceived as a kinesthetic or tactile response [2]. Kinesthetic feedback is primarily recognized as the force or torque feedback while tactile feedback involves the sensation of pressure, shear, vibration when touching an object [3]. Early studies have described the dimensions of haptics as texture, weight, hardness, volume, temperature, global shape, and these are tightly bound to the nature of the object [3]. Most studies on haptics are based on motor control and reliant on processing and representation of kinesthetic feedback. Furthermore, the design and control of the haptics system could be typically categorized as graspable, wearable, and touchable as per the studies on haptics [1]. Graspable systems are commonly kinesthetic devices, which allow the user to feel the force feedback by pushing them on through the held tool. Wearable devices include exoskeletons that provide kinesthetic feedback and tactile devices which display sensation regarding vibration and deformation. Touchable devices often allow the user to actively explore the tactile properties of the entire surface, but they also can be the hybrid of kinesthetic and tactile versions. Recently, researchers have also taken heed of studies on virtual touchable objects created in midair allowing the user to experience the texture of the surface of the object through the vibration feedback [4].

Various approaches were investigated to interpret the feeling of touch excels at sensing. Many studies of haptics focus on creating kinesthetic sensations and feedback. An object's reaction comprises not only the information regarding force

and motion but also its impedance. Hence, the actual impact of all effecting factors should be assessed and understood from the actual environment to accurately identify and reconstruct objects in virtual reality while providing the user an immersive feel of a virtual world.

With the passage of time, different approaches were investigated to interpret the feeling of touch excels at sensing. Most studies of haptics focus on creating haptic sensations using model-based approaches [5]-[7]. The conventional spring-damper model or spring model were typically used to define the object in the virtual world [5]-[7]. Hence, the force response from the virtual object is displayed with the traditional stiffness rendering algorithm which is based on Hooke's law [8] while incorporating the damping effect and the effect of mass appropriately. However, spring damper behavior doesn't reflect the real sensation of touch as the real sense of touch is nonlinear [9]. Thus, model-based identification failed to interpret the real sense of touch. Furthermore, most haptics studies entirely relied on motion data and the position and velocity information were primarily considered [5]-[7].

In most instances, force/torque sensors have been used to measure the force [7] ignoring their issues like signal noise, instability, narrow bandwidth, non-collocation and complicity [10],[11]. Even the most advanced technological devices use thin flexible force sensors developed by advanced force sensor technologies, yet they can still gauge the force where they are mounted [12]-[14]. Besides, the force sensor adds inertia or mass to the system. However, a robust sensorless force/torque control mechanism over a broad bandwidth can be achieved using the Disturbance Observer (DOB) [15],[16] and later the Reaction Force Observer (RFOB) [10], [17] was introduced with some adjustments to detect the reaction force.

Playback of recorded haptic information is the simplest and direct technique, and motion coping systems commonly use this concept [18],[19]. Nevertheless, the direct replay of haptic information frequently fails to adequately grasp the complexity of the original interaction. As a result, complicated models are needed to generate the haptic information as if it's from the actual environment.

Furthermore, the nonlinear motion made should be exactly matched with the rendered signal for virtual interaction and to realistically interpret the actual interaction. Hence, researchers used the dominance of vision over haptics to capture motion information ignoring the perceptual capabilities that the touch sense alone composed [20]. However, in the real scenario, the haptic information should change with the applied parameters on the virtual object. Thus, haptics needs to empower with intelligence.

In the last decade, researchers have taken many attempts to design and develop haptic devices for many potential applications and recent studies use AI. However, these studies have used AI to have vision information to reproduce sense of touch ignoring the potential of sense of touch [20],[21] and they have relied on force sensors, strain gauges despite their issues [22].

1.1. Problem Statement

Hence, it seems that identifying the actual behavior of the object is critical for object reconstruction in the dimension of haptic. However, the majority of studies relied on traditional spring damper model and spring model to recreate the object ignoring the real nonlinear behavior of the environment object. Furthermore, most studies have used force sensors despite their issues to obtain the force measurements. Recent studies have employed AI with haptics, but they have considered dominance of vision and have relied on force sensors.

However, usage of the AI approaches has proven to produce better results through any studies. Thus, in this work AI approaches were considered to recreate the haptic sensation than relying on the conventional approaches. However, more reliable prediction can be achieved through AI when there is a clear understanding about the dataset. Therefore, it is essential to identify the impact of the features affecting haptic feedback before building the AI model. Furthermore, force measurements that depend on the sensorless force/control mechanism provide more accurate measurements than the force sensors. Thus, sensorless force sensing mechanism was incorporated to obtain haptic information and to achieve better results through proposed approach.

1.2. Project Objectives and Scope

1.2.1. Objectives

The objective of the research is to study haptic objects reproduction in virtual reality using AI.

- Develop a deep learning based generalized virtual object model for an object.
- Utilize the AI for the reproduction of vivid force sensation in virtual reality as if it is from the real environment.

1.2.2. Scope

The scope of this research is to utilize AI based approaches for haptic object reconstruction and reproduction in virtual reality as if it's from the actual environment. A DOB and RFOB based sensorless sensing mechanism was employed to abstract haptic information for the reconstruction and reproduction of haptic sensation. Furthermore, this study follows a statistical analysis of haptic information to understand the features affecting reproduction of haptic sensation in virtual reality.

1.3. Project overview

This research focuses on haptic object reproduction in virtual reality. The procedure of this research follows three stages Abstraction, Reconstruction and Reproduction as shown in Figure 1.1. A sponge was considered as the experimental object throughout the research.

The haptic information is abstracted and mainly force response and compression depth of the object were obtained. However, there can be many other features affecting the feedback from the object. Thus, the features are extracted to create the whole haptic dataset. The haptic dataset was analyzed to understand the behaviour of the features and the impact of them for the response. The features with the highest impact were taken as selected features to be used in building the machine learning model. Since there are numerical input features and numerical output, this prediction is regression. Thus, many regressions algorithms were evaluated to find the best

algorithm match with the dataset. Finally, the selected model was utilized in the reproduction stage to reproduce the haptic object in virtual reality.

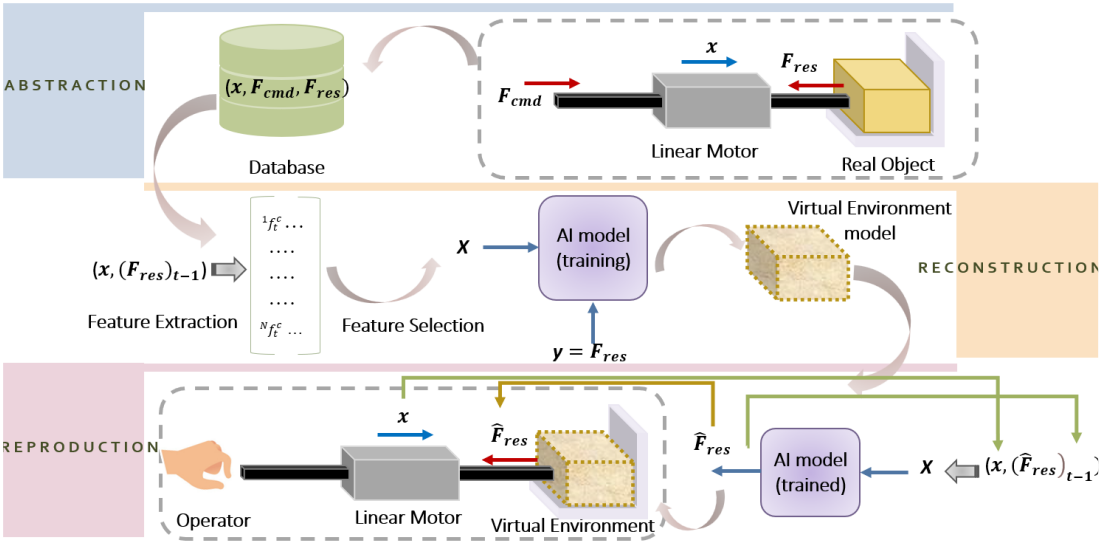


Figure 1.1: Representation of the Abstraction phase, Reconstruction phase and Reproduction phase of the haptic object.

The mathematical notation used in Figure 1.1 is elaborated in section 3.2.1.

1.4. Thesis outline

The thesis structure is organized as follows. Chapter 2 highlight prior research work, chapter 3 explains the proposed AI approach, chapter 4 discusses the system implementation and chapter 5 proves a detailed hardware development for the research. The results of the research are discussed in chapter 6 and the techniques including AI algorithms used in the research are explained in chapter 7.

Chapter 2 provides a glance at the prior research work related to this study area. It's the literature review, and it consists of five main sub-sections. The first subsection explains the traditional conventional approaches which are mostly utilized even in current studies on haptics. The drawbacks of using force sensors are highlighted in the next sub section and it explains the importance of use of sensorless techniques for measuring force. Next two sections explain the issues with playback of haptic information and dominance of vision over haptics. Finally, the chapter highlights the use of AI with haptic technologies.

The proposed approach is discussed in chapter 3, and it provides the detailed procedure of the research explaining the three phases: Abstraction, Reconstruction, and Reproduction. The reconstruction phase has two stages feature extraction and modelling and it is also highlighted in the chapter.

The system implementation of the reproduction phase is explained in chapter 4, and it explains how the AI model is migrated to another intermediate stage and utilized for the haptic object reproduction in virtual reality.

The next chapter shows the hardware used and how they are utilized in the abstraction phase to acquire haptic information and in reproduction phase for haptic sensation reproduction as if from the real environment.

Chapter 6 provides the overall discussion of the results obtained from the research and how the objectives were achieved during the research. This chapter covers the results of the whole procedure of the study including abstraction, reconstruction, and reproduction phases. Finally, discusses the validity of the proposed approach and proves that it is better than existing approaches.

Chapter 7 focuses on highlighting the AI algorithms utilized and analytical techniques considered.

Ultimately, the overall discussion and the conclusion are followed by References and Appendices.

2. LITERATURE REVIEW

Humans interact intensively with machines through auditory and visual senses, yet they frequently disregard the sense of touch, which they rely on the most. Sense of touch is another sensory information which could be controlled as the visual and auditory sense. Therefore, there has been an increasing focus in recent decades on developing technology to replicate the sense of touch and convey important tactile information to improve human performance. As a result, researchers have been exploring ways to give new generations of robots the ability to perceive and interpret touch, incorporating AI to provide intelligence and adaptability to this capability. With the development of immersive technologies, the absence of realistic haptic feedback has been acknowledged as a significant barrier in achieving realization in virtual reality [1] and carrying out a variety of real-life manipulation tasks [23]. However, its potential is enormous in the various fields including medicine [24]-[28], agriculture [29], exploration [30], industrial robots [31],[32], and gaming [33]-[36]. Additionally, the sense of touch is crucial in many teleoperation situations, such as repairs that require avoiding self-presence and procedures that are beyond human scale. One crucial area of research in touch feedback for teleoperation is the provision of real-time touch feedback to da Vinci robotic systems utilized in minimally invasive surgeries, which continues to be of significant interest to researchers [24]-[27], [37]-[39].

The field of haptics, which focuses on the science of touch, not only allows humans to break down barriers to achieving a sense of presence in virtual environments but also enables them to carry out a wide range of real-world manipulation tasks. Haptic information involves a reaction to an action and is bilateral. The haptic feedback can be typically identified as kinesthetic or tactile [1],[2]. Kinesthetic feedback is recognized as force or torque feedback, while tactile feedback involves the sensation of pressure, shear, or vibration when touching an object [2],[3]. Early research identified haptic dimensions such as texture, weight, hardness, volume, temperature, and global shape, which are closely linked to the properties of the object [2],[3]. Most haptic research focuses on motor control and

relies on the processing and representation of kinesthetic feedback. In addition, haptic systems can be categorized as graspable, wearable, and touchable [1]. Graspable systems are often kinesthetic devices that provide force feedback when the user pushes or moves the tool. Wearable devices include exoskeletons that provide kinesthetic and tactile feedback, with tactile devices capable of displaying sensations such as vibration and deformation. Touchable devices allow the user to actively explore the tactile properties of an object's surface and can be a combination of kinesthetic and tactile versions. Recently, researchers have explored the creation of virtual touchable objects in midair, allowing users to experience the texture of an object's surface through vibration feedback [4] and it paves the way for more immersive virtual reality experiences and new opportunities for tactile interaction.

Unlike acoustic and vision information, haptic information depends on the response from the environment as shown in Figure 2.1. Thus, sense of touch is bilateral. Hence, the force response plays a vital role in realization of virtual reconstruction of haptics object. Therefore, to reproduce the sense of touch through an environment model, the response from the real environment should be considered incorporating the input motion parameters.

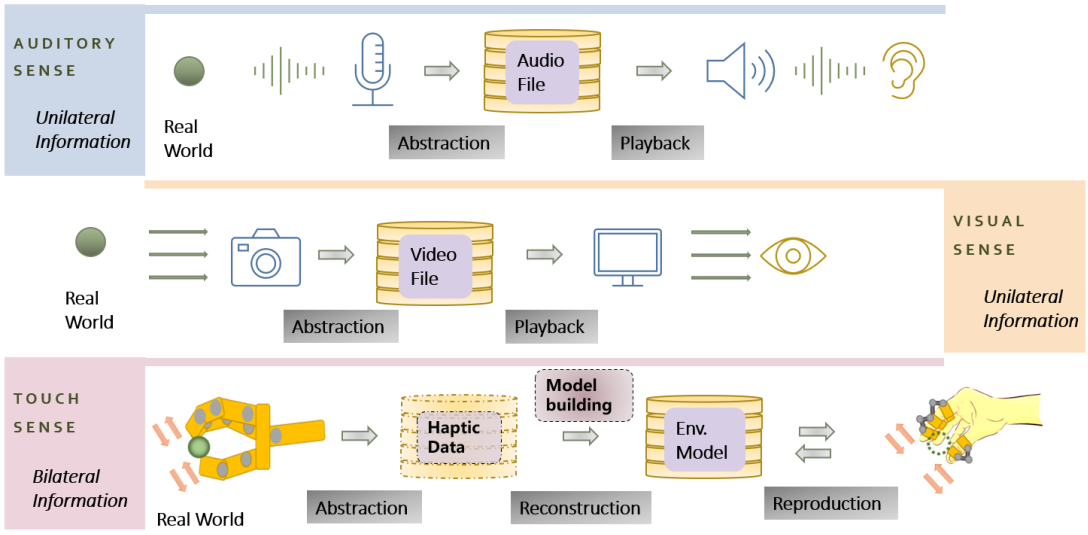


Figure 2.1: Behavior of human sensory information.

2.1. Model based approach.

With the passage of time, various approaches have been explored to better understand how the human sense of touch excels at sensing. The majority of haptic research has focused on creating kinesthetic sensations, and most studies have employed model-based approaches when modelling the behavior of the actual object for reconstruction, and the conventional spring-damper model based on Hooke's law [8] was the most utilized while appropriately incorporating the damping effect and the effect of mass. Hence, they have derived the environmental impedance by predefining the behavior of stiffness and viscosity as exponential or polynomial or constant [5]- [7]. The spring-damper behavior has been used to interpret interaction with virtual objects and to deduce the interaction forces that occur when touching the virtual object in haptic rendering algorithms [7]. However, considering spring damper behavior to represent an object contradicts the real behavior of environmental impedance of that object as the real sense of touch is nonlinear [9]. Therefore, model-based identification has often failed to accurately interpret the real sense of touch.

2.2. Sensorless Force/Torque control

The influence of the learned force requires special attention and yet, haptics studies simply concern considering only motion data. Information with both motion and force were considered to enhance the quality of reproduction of grasp performed using a strain gauge based cyberglove, and force sensors were used to obtain force magnitude in this study [7]. However, force sensors have issues such as,

- Narrow bandwidth
- Signal noise
- Complicity
- Non-collocation
- Instability [10] [11].

Even the latest haptic devices such as haptic gloves use thin flexible force sensors, though force sensors can only measure force at the location of the sensors and the

sensor itself adds mass or inertia to the system [40]. Thus, a sensorless force control with a wide bandwidth achieved using the Disturbance Observer (DOB) [15],[16] and the Reaction Force Observer (RFOB) [10],[17] was introduced by Kouhei Ohnishi et al.

2.3. Playback of haptic information

The direct playback of haptic information is a simple and straightforward method used in motion coping systems. A motion-copying system based on DOB and RFOB was introduced to reproduce position and force information saved based on bilateral control [18]. This system was used to reproduce position and force information stored in motion-data memory. A real, simple haptics device constructed using an LSI module and this device composed of the real haptics basic functions such as,

- Motion transmission from the master to slave
- Motion recording
- Motion playback
- Force and position scaling [19].

However, playback of haptic information fails in fully interpretation of the real interaction complexity. Furthermore, the motion made by the person should be exactly matched with the rendered signal for the virtual interaction to realistically mimic the actual interaction. Thus, complex models are required to render realistic haptic information.

2.4. Dominace of vision over haptics

Researchers have used the dominance of vision over haptics to capture motion information ignoring the perceptual capabilities that the touch sense alone comprised. A Convolutional Neural Network (CNN) has been used in the motion reproduction system to estimate path by using grasping motion and depth information [20]. Hence, they have used visual information to reproduce the grasping force which is estimated through equations derived using kinematics. So, their focus of using machine learning is to analyze motion data through vision information by addressing vision as an alternative for sense of touch and neglecting the real potential sense of touch.

Thus, haptics needs to be empowered with intelligence to provide a more realistic and responsive interaction.

2.5. AI with haptics

Haptic devices often require a high degree of accuracy and precision, which can result in high costs. Additionally, they may require a specialized design or setup, which can limit their applicability and reduce their market potential. Despite these challenges, researchers and companies continue to develop haptic devices for a variety of applications, and there is growing interest in the field of haptics. Though haptic devices have seen significant progress in recent years, they still face challenges when it comes to cost, size, and wearability.

AI has the potential to revolutionize the field of haptics by enabling more personalized and intuitive haptic interactions. With AI, haptic devices can be designed and optimized to provide more natural and realistic touch sensations, and to respond to the needs and preferences. Artificial intelligence paves the way for aggregating robust data-driven methodologies and other technologies to enable solutions. Thus, AI can be used to develop more advanced and sophisticated haptic systems that can be integrated with other technologies, such as virtual reality, to create more immersive and engaging experiences. Data-driven model building based on AI has the potential to overcome the limitations of traditional physics-based models by allowing for more flexible and accurate simulations of haptic feedback. AI techniques can learn from large datasets of haptic interactions and generate models that capture the complexity and nonlinearity of the human sense of touch. This approach can enable more natural and intuitive haptic interactions with virtual environments, and potentially lead to the development of more affordable and accessible haptic devices. Thus, the latest studies on haptics consider AI for seamless integration haptic devices with virtual reality and tune the haptic technology to evoke affective responses. A neural network has been utilized to predict external forces from motion parameters in a neural network model-independent force observer [21]. However, force sensors were utilized in the training stage of the neural network to measure contact forces despite their drawbacks [10], [11]. A non-linear regression

model, SVR (Support Vector Regression) was employed to infer the haptic force positions for stimulation locations which are unseen [22]. However, the force measurements were thoroughly dependent on strain gauges despite the drawbacks of the force sensors [10] [11].

The success of object reconstruction in the haptic dimension depends on accurately identifying the object's behavior. Traditional methods using spring damper and spring models to recreate the object's behavior have limitations as they don't account for the non-linear behavior of the environment or object. Additionally, force sensors used in many studies have their own issues. Recent studies have employed AI with haptics and have considered vision as a dominant factor. This study focuses on using an environmental model developed through learning haptic information to recreate haptic sensations in virtual reality. The study identifies significant features for building the virtual model and evaluates various algorithms to find the best fit with the dataset. The study also utilizes a sensorless force control mechanism to obtain force measurements for better "reproducibility" of haptic sensations. Ultimately, the study proposes AI-based approach to vividly reproduce force sensations through a virtual model that replicates the actual environment using information extracted through a sensorless control mechanism.

3. THE PROPORSED AI BASED APPROACH

In the real scenario, the haptic sensation should change with the applied parameters on the virtual object. Thus, complex models are required to render the real haptic information. Therefore, when constructing the virtual environmental model, complex haptic environment modeling techniques should be employed to interpret the nonlinear behavior of the responses as if from the real environment. Thus, it is important to empower haptics with intelligence. Many haptic studies which incorporated AI, have primarily considered motion parameters such as position, velocity, and acceleration as inputs for the AI model. Furthermore, research highlighted the fact that the environmental impedance which considers the effect of stiffness and viscosity coefficient of the object is not a constant. Thus, several factors can affect the haptic force feedback and these factors could complicate the prediction process. Therefore, it is essential to identify the impact of the features before building the AI model. Furthermore, more reliable prediction can be achieved through AI when there is a clear understanding about the dataset. Therefore, a comprehensive study of the features along with a principal component analysis (PCA) was performed to identify the most relevant features that have a higher impact on the force response.

An environmental model developed through learning of haptics information by using the extracted features. Various algorithms were used in the research and the best model fit with the dataset was used for the reproduction of haptic sensation. Furthermore, use of a sensorless sensing system is essential for better “reproducibility”. Therefore, this research introduces deep learning-based approach for vivid force sensation reproduction through a virtual model which replicates the actual environment, and the proposed approach utilizes sensorless force control mechanism to obtain force measurements.

3.1. Abstraction phase

The motion information with the corresponding responses from the object were acquired during this phase. An adequate amount of data is needed to analyze the factors affecting the recreation of haptic sensations using AI. Thus, the relevant

information with the responses from object was abstracted as depicted in Figure 3.1. Despite these drawbacks, force sensors were utilized in traditional force controlling to detect force. Furthermore, the sensor itself adds inertia or mass to the system and it only detects external forces at the location where it is mounted. Thus, a DOB and RFOB based sensorless technique was used in this study to detect vivid force sensation with a wide bandwidth. Robust force control was achieved using DOB, and reaction force from the object was measured using RFOB.

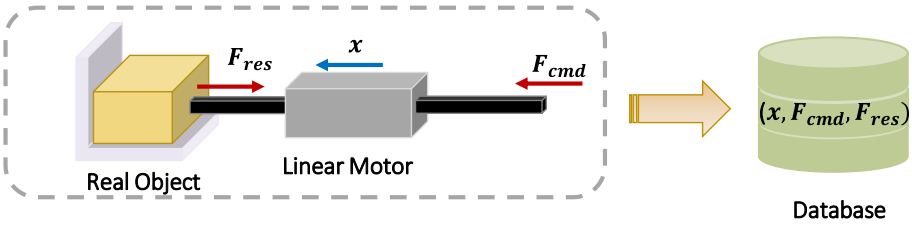


Figure 3.1: Abstraction of haptic information

The disturbance force of the system observed by the disturbance observer without using force sensors is represented as in (3.1).

$$F_{dis} = F_{ext} + F_{int} + \left(F_f + B\dot{x}\right) + \ (M - M_n)\ddot{x} + (K_{fn} - K_f)I_a^{ref} \tag{3.1}$$

$$F_{dis} = K_{fn}I_a^{ref} - M_n\ddot{x} \tag{3.2}$$

The generated motor force, F_m can be represented as shown in (3.3).

$$F_m = K_f I_a^{ref} \tag{3.3}$$

By applying the dynamic equation to the linear motor:

$$F_m - F_l = M\ddot{x} \tag{3.4}$$

Load force, F_l can be represented as in (3.5).

$$F_l = F_{int} + F_{ext} + (F_f + B\dot{x}) \tag{3.5}$$

The parameters, K_f and M can be re-written in terms of nominal values and variations as they are subjected to variations and estimation errors.

$$M = M_n + \Delta M \tag{3.6}$$

$$K_f = K_{fn} + \Delta K_f \tag{3.7}$$

The estimated disturbance force was obtained by passing disturbance through the low pass filter and suppressing the noise due to the differentiator as in (3.8).

$$\hat{F}_{dis} = \frac{g_{dis}}{(s+g_{dis})} F_{dis} \tag{3.8}$$

The DOB was modified and introduced RFOB to estimate the reaction force without using force sensors by recognizing the internal disturbance in the system [6],[11]. Thus, RFOB plays the role as a virtual force sensor in estimating only the reaction force. The estimated value of the reaction force can be expressed as in (3.9) and (3.10).

$$\begin{aligned} \hat{F}_{ext} = \frac{g_{rec}}{(s+g_{rec})} (K_{fn} I_a^{ref} + M_n g_{rec} \dot{x} - \\ (F_{int} + F_f + B\dot{x} + (M - M_n)\ddot{x} + (K_{fn} - K_f) I_a^{ref}) - M_n g_{rec} \dot{x} \end{aligned} \tag{3.9}$$

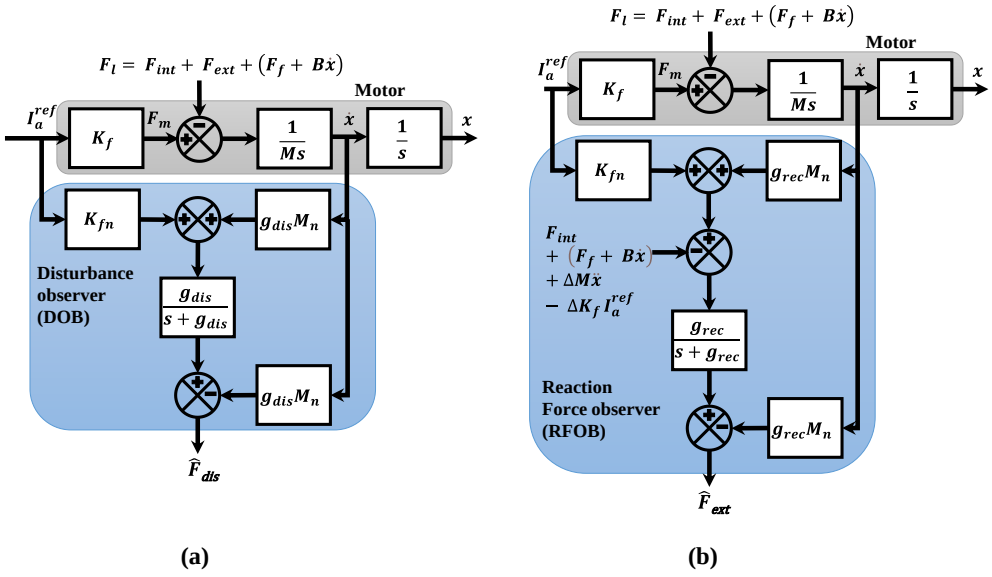


Figure 3.2: Block diagrams (a) Disturbance Observer (b) Reaction Force Observer

$$\hat{F}_{ext} = \frac{g_{rec}}{(s+g_{rec})} F_{ext} \tag{3.10}$$

The block diagrams of the disturbance force observation function by the DOB and the reaction force estimation function by the RFOB are shown in Figure 3.2. Figure 3.3 shows the overall block diagram with the DOB and RFOB based force control mechanism.

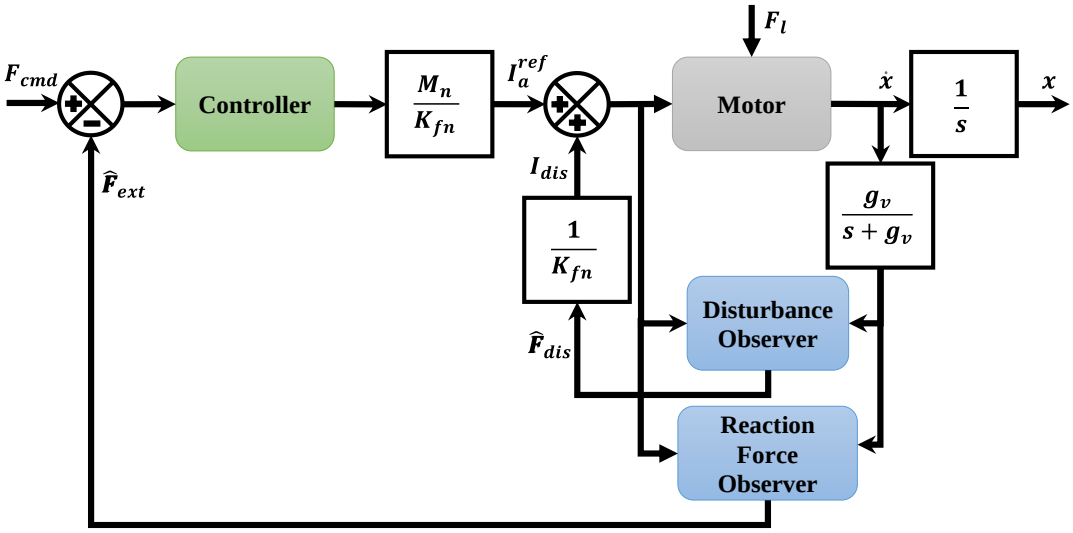


Figure 3.3: DOB and RFOB based Force controller.

3.2. Reconstruction phase

This phase involves reconstructing the environment to recreate the force response from the environment. Most studies have utilized spring damper model by deriving the environmental impedance using stiffness and viscosity. Hence, the environment impedance was predefined by considering the behavior of stiffness and viscosity as exponential or polynomial or constant. Consequently, the force response was defined by assessing the combinational effect of the forces from the spring, and the damping force as expressed in (3.11).

$$F_{res} = F_k + F_b = kx + b\dot{x} \tag{3.11}$$

However, spring damper model doesn't interpret the real nonlinear behavior of the environment/ object. Thus, AI was considered to recreate the environment

model to generate the force response as if from the real environment. The reconstruction phase shown in Figure 3.5 includes two stages, feature extraction and modeling.

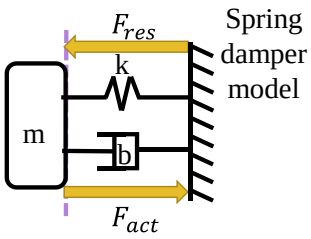


Figure 3.4: Spring damper model

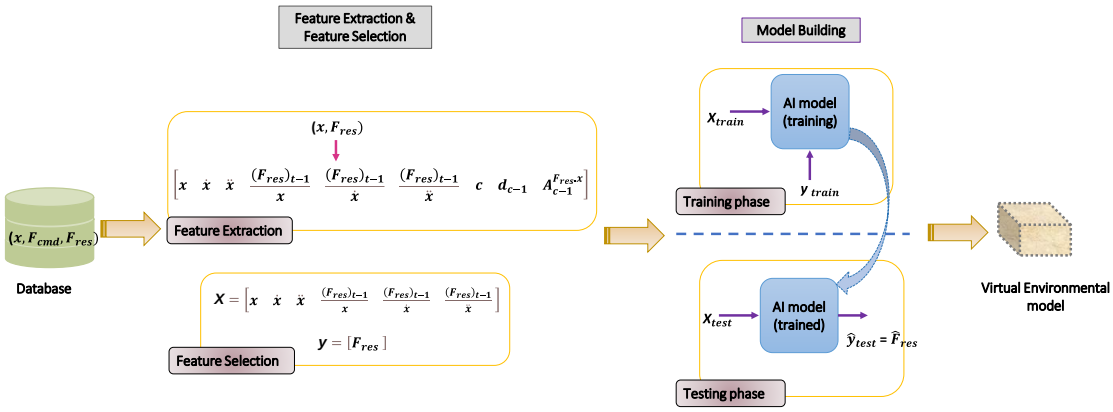


Figure 3.5: Reconstruction phase

3.2.1. Mathematical modelling of features

A feature was defined using mathematical notation. The feature, f can be defined as follows,

$$ {}^{n}f_{t}^{c} \tag{3.12}$$

where n = feature no.

c = cycle no.

t = time elapsed from the cycle start

Thus, the feature matrix can be defined with 3 dimensions. The feature no., time elapsed and the cycle no. change along the dimensions. The feature matrix and the instance of the matrix is shown in Figure 3.6.

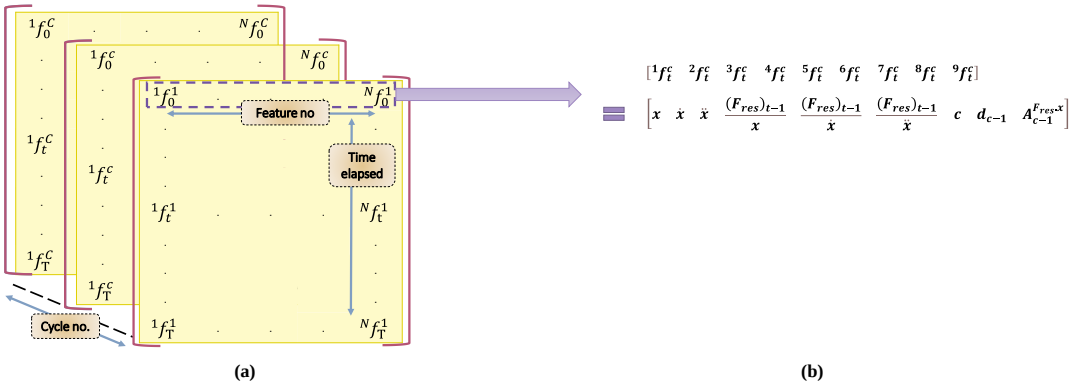


Figure 3.6: (a) Feature Matrix (b) Instance of the Feature matrix

3.2.2. Feature extraction

A labeled structured dataset of 21400000 samples was extracted for a force command applied to the sponge object. However, a reduced version was considered for this analysis. The variation of haptic information relative to time and motion parameters is illustrated in Figure 3.7. Filtered velocity and acceleration were used after passing through a law pass filter.

$$\dot{x} = \frac{g_v}{s+g_v}\dot{x}_{raw} \tag{3.13}$$

$$\ddot{x} = \frac{g_a}{s+g_a}\ddot{x}_{raw} \tag{3.14}$$

where g_v and g_a are velocity and acceleration filter constants and their values used in the research are 30 and 100. $\dot{x}_{raw}$and $\ddot{x}_{raw}$ are the raw value of velocity and acceleration before filtering. The filter constants were selected to obtain a clear variation of the parameters while maintaining a less than 10 % change within the 100ms time frame window.

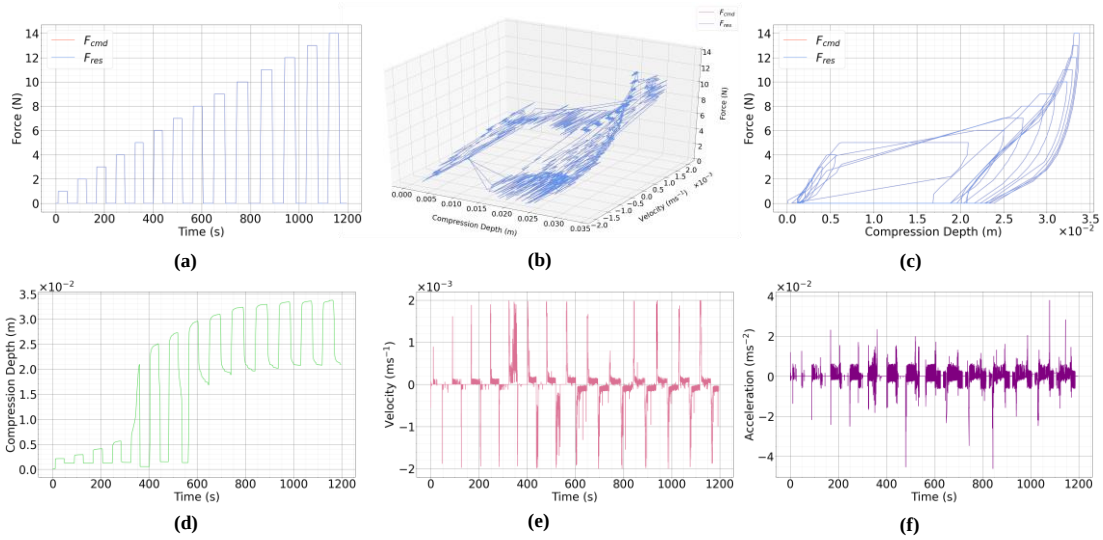


Figure 3.7: a) Force profile on the object over time b) Force profile on the object over motion parameters c) Force profile on the object over compression depth d) Compression depth profile on the object over time e) Velocity profile on the object over time f) Acceleration profile on the object over time.

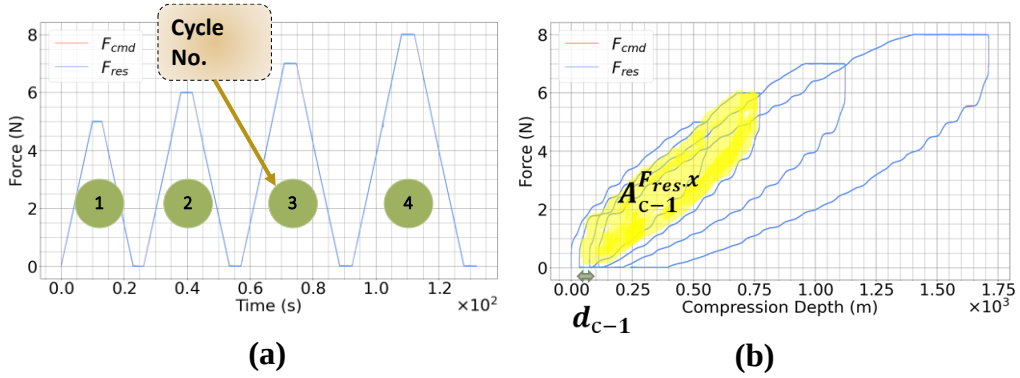


Figure 3.8: Deriving features: cycle no., permanent deformation, and the area from the force response vs compression depth graph.

Most studies highlighted the impact of motion parameters namely compression depth, velocity, and acceleration on the force response. Thus, all motion parameters were used as features. When examining the F_{res}vs x graph it seemed that, the object has an unrecovered deformation as the paths the object takes in F_{res} vs x when increasing the force is not equals to when reducing the force. Thus, it clarifies that the object has not returned to the same state. Furthermore, the area within the path enclosed by the full force apply-release cycle differs from cycle to cycle with the

increase of the maximum force command and Figure 3.8 depicts the area of a cycle. Thus, these observations were considered to derive new features to be used for further analysis. Therefore, cycle no., permanent deformation and the area from the force response vs compression depth graph were chosen as the features after observing the illustrated variations. Since literature highlighted the environmental impedance is not a constant, the effect of instantaneous stiffness coefficient and viscosity coefficient was incorporated by considering the $(F_{res})_{t-1}$ over motion parameters namely compression depth and velocity. Furthermore, $(F_{res})_{t-1}$ over acceleration was also considered. Therefore, previous instantaneous force response over individual motion parameters were considered as the features that impact the force response. Thus, in this study 9 features were identified, and the features considered for this study are listed in Table 3.1.

Table 3.1: Feature description

Feature	Description
x	Compression depth
$\dot{x}$	Velocity
$\ddot{x}$	Acceleration
$\frac{(F_{res})_{t-1}}{x}$	Instantaneous stiffness
$\frac{(F_{res})_{t-1}}{\dot{x}}$	Instantaneous viscosity
$\frac{(F_{res})_{t-1}}{\ddot{x}}$	Previous Force response over acceleration
c	Cycle No.
d_{c-1}	Permanent Deformation
$A_{c-1}^{F_{res}.x}$	Area F_{res} vs x curve

AI approach is data driven. Thus, the prediction will be more accurate when there is more data and the attributes. However, redundancy can cause data inconsistency and will lead to unreliable and meaningless information, and it will cost more

computational power and time. When there are many variables, the relationship between features cannot be identified manually or visually. Thus, it is essential to use descriptive statistical analysis to have insights of the dataset while recognizing significant features and the relationship between features.

Table 3.2: Item Statistics

Feature	mean	Standard deviation	min	25% percentile	50% percentile	75% percentile	max	median	variance
x	0.02	0.01	-6.50E-05	0.02	0.03	0.03	0.03	0.03	1.24E-04
$\dot{x}$	9.68E-06	0.0003	-0.01	-2.70E-05	3.70E-05	6.50E-05	0.01	0.000037	9.38E-08
$\ddot{x}$	-7.69E-05	0.002	-0.10	-0.0002	-3.50E-05	2.60E-05	0.19	-0.000035	3.50E-06
$\frac{(F_{res})_{t-1}}{x}$	305.79	331.35	-19682	9.33	302.17	413.35	41620	302.17	1.10E+05
$\frac{(F_{res})_{t-1}}{\dot{x}}$	146538.6657	341837.09	-14000020	0	128206.23	195654.47	14000020	128206.23	1.17E+11
$\frac{(F_{res})_{t-1}}{\ddot{x}}$	-274613.4836	970149.49	-14000182	-188681.75	-31056.523	0	14000148	-31056.523	9.41E+11
c	5.37	3.54	1	3	4	7	14	4	1.26E+01
d_{c-1}	0.003	0.01	-3.50E-05	0	0.0002	0.002	0.02	0.0002	4.27E-05
$A_{c-1}^{F_{res}.x}$	0.06	0.05	0	0.01	0.03	0.13	0.14	0.03	2.94E-03
F_{res}	5.96	4.87	0	0.15	6.00	10.00	14.00	6.00	2.38E+01

The dataset only consists of numerical variables to detect force response. Most common statistical measures that are used to analyze a dataset can be categorized as the measures for central tendency and variability. Mean, mode and median are the measures of central tendency which reflects center of the distribution. Dispersion, and variability of the distribution can be obtained from common measures such as

range, standard deviation, and variance. The summary of all the statistical measures are given in the Table 3.2.

Many statistical measures were considered to analyze the dataset and find the features with higher impact in recreating the sense of touch. The measures that were used in the research are,

- Correlation
    - Pearson correlation
    - Spearman's correlation
- Mutual information
- Principal Component Analysis (PCA)

3.2.3. Model Building

A continuous ramp force was applied on the object to identify its object's behavior when commanding no. of force apply-release cycles on the object. The training set was created by using different variations of ramp force and a set with different ramp rate was considered as the test set. Figure 3.9 illustrates the variation of recorded force relative to time for both datasets. The training set was a combination of data with different ramp rates: 0.8, 1.8, 3.9 and 6.2 while the ramp rate of the test set was 2.1.

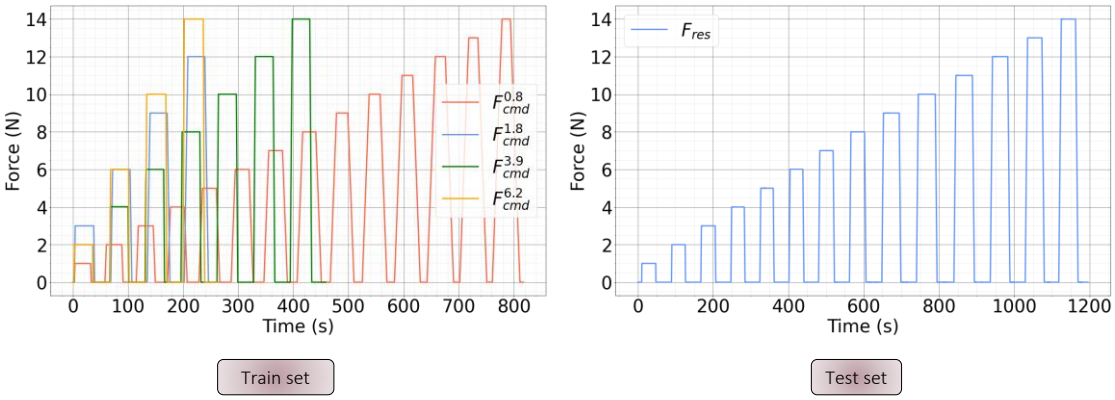


Figure 3.9: Train – Test sets

Table 3.3: Train - Test Datasets

Dataset	No. of samples
Training dataset	362570
Testing dataset	292680

The training dataset was used to build the AI model by recognizing the relationship between force response and extracted features. The testing set was used to test the model built using AI by comparing the actual force response values with the estimated force response values. Moreover, these two datasets are separate sets and there are no data common to both datasets. Hence, the test data were not considered during the training process.

The dataset consists of numerical input variables and numerical output variables. The proposed AI-based approach uses supervised learning which is based on multi-label regression. Several AI algorithms were considered in model building using python. They are,

- Support Vector Regression (SVR)
- Random Forest Regressor
- Linear Regression
- Neural network

Regression Matrices were used to evaluate the performance of these algorithms to choose the best algorithm fit the haptic dataset. The regression metrics used in this study for model performance evaluation are,

- R^2 score
- Mean absolute error (MAE)
- Mean squared error (MSE)
- Root mean squared error (RMSE)

The model should have a higher R^2 score and lower regression losses denoted by MAE, MSE, and RMSE for the model to have a better performance. The selected AI algorithm was used in model building, and it is converted to the intermediate format

and deployed with the hardware to be utilized for haptic object reproduction in the reproduction phase.

3.3. Reproduction

In the reproduction phase involves utilizing AI model with the hardware to reproduce force sensation as if from the real environment. Thus, the user could be able to feel the response from a sponge object even if there is no object present in the reality. Thus, when user input a force with a position change the force response prediction follows following steps.

- Important features extracted from these features.
- Execute the AI models by using the important features as the inputs to the AI models.
- Find the best AI model with the better performance.
- Predict the force response from the selected AI model.
- Predicted force response is feed into the motor to let the user feel the force feedback as shown in Figure 3.10.

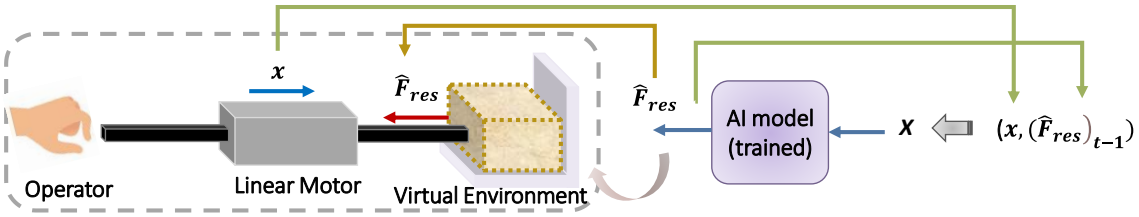


Figure 3.10: Reproduction phase

In this study the AI model was developed in R and converted to PMML format to utilize the AI model to deploy with the hardware. PMML is a file format based on XML and it can be used as an intermediate format between different programming languages.

4. SYSTEM IMPLEMENTATION

Full system implementation is illustrated in Figure 4.1. The system can be separated as the hardware and the software. The hardware system consists of the below components.

- Linear motor
- Linear encoder
- Motor driver
- Analog/digital I/O module

The whole system operates in force control mode. The force is applied as a current command through the linear motor. The compression depth measurement is taken by encoder and the value is sent to the motor driver and Sensoary analogue/ digital I/O module.

The main software program executes in C++. The encoder value and the timestamp values are retrieved to derive forces, current command values and other parameters and features. In the abstraction phase a force command is applied on the sponge object. Thus, the main program executes by commanding force according to the defined ramp force apply and release cycles. However, the predicted value from the ML model is commanded to move the motor in the reverse direction in the reproduction phase. Thus, when the user inputs a motion with a particular force and compression depth / position, the response generation follows the below steps.

- Extracting important features
- Executing the AI model by using important features as the input to the AI model
- Predicted force response from the AI model.
- Predicted value is fed into the hardware as a current command.

The AI model is created using R and it’s migrated to a model interchange format which supports the programming language of the main program.

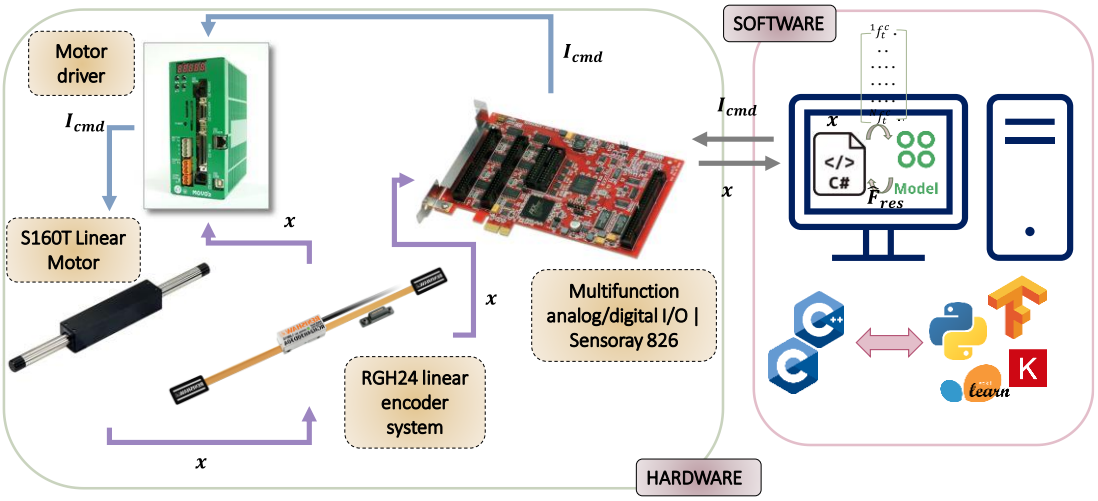


Figure 4.1: System Implementation

4.1. ML model integration with the hardware

Several methods were investigated to seamlessly integrate AI model with the hardware. Hardware is communicated with software using C language while the AI model is created using python. However, C language doesn't support machine learning libraries. But it is essential to have some mechanism to support AI model integration with the hardware.

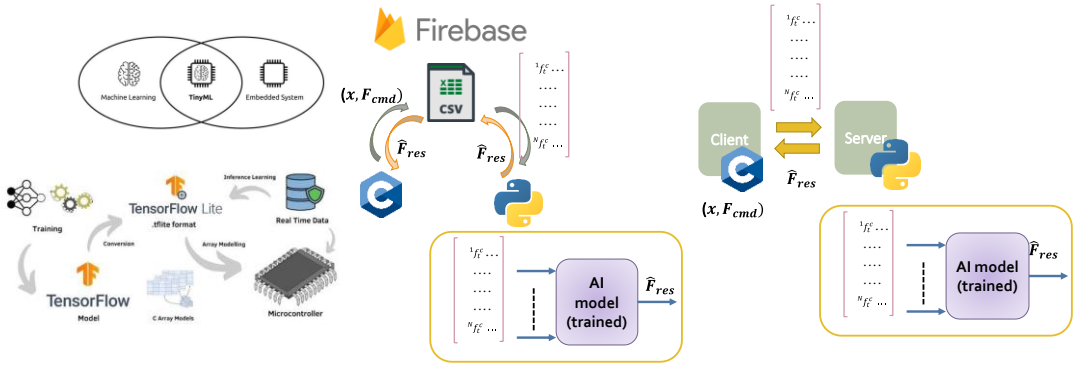


Figure 4.2: (a) Use of TinyML (b) Use of File/ Database (c) Client Server programming
Source (a): https://www.allaboutcircuits.com/technical-articles/what-is-tinyml/

TinyML is the most popular library used in embedded devices to perform AI tasks with real-time responsivity. But it doesn't support for sensoray module. Thus, updating and accessing the same database or the same file was the next attempt to separately execute C and python scripts. However, the whole process of execution takes longer time. In case of updating the file, it takes around 100ms while the case of accessing database will take around 5 ms. Thus, client server programming was considered where main C program was executed as a client while AI python script was executed as the server. However, the total execution process takes a time about 4 ms which is still long enough to have the controller isolated.

4.1.1. Predictive Model Markup Language | PMML

Predictive Model Markup Language, or PMML in short, is a file format which based on XML, and it serves as an intermediate form between different programming languages. A model can be built using R/ Python and saved as an PMML file [41]. Then the PMML file can be utilized in reproduction phase.

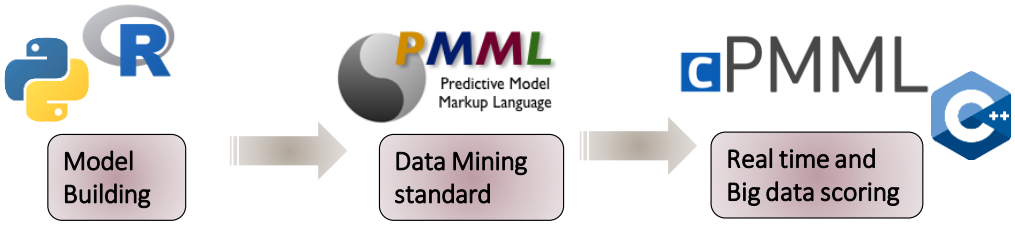


Figure 4.3: Model deployment with PMML

In the research the model was created in R and converted to the PMML file and loaded the AI model form C++ main program. PMML supports,

- Regression models
- Support Vector Machines
- Naive Bayes classifier
- Decision trees (Random Forest)
- Clustering models
- Neural Networks
- Association rules

- Gradient Boosting (LightGBM and XGBoost)
- Text models

4.1.2. cPMML

This library was used to execute AI models with C++ main programmed code. It is a C++ library for scoring PMML-serialized machine learning models. It has a user-friendly and minimalist API. It can achieve high performance in model scoring, and it keeps a predictable and minimal memory footprint. PMML supported elements are,

- PMML General structure (preprocessing, data dictionary, etc.)
- Regression models
- Tree-based models
- Ensembles of the previous

5. HARDWARE DEVELOPMENT FOR THE TEST BENCH

The experimental setup employed to obtain information on haptic sensation is shown in Figure 4.1. This research focused on replicating 1 DOF haptics interaction. A linear motor was used as the actuator and a linear encoder was employed to obtain position/ compression depth. The analog/digital I/O, Sensoray 826 was programmed using C++ language. Table 4.1 lists the values of experimental parameters. The initial position of the actuator was kept on the surface of the object to obtain relevant motion data.

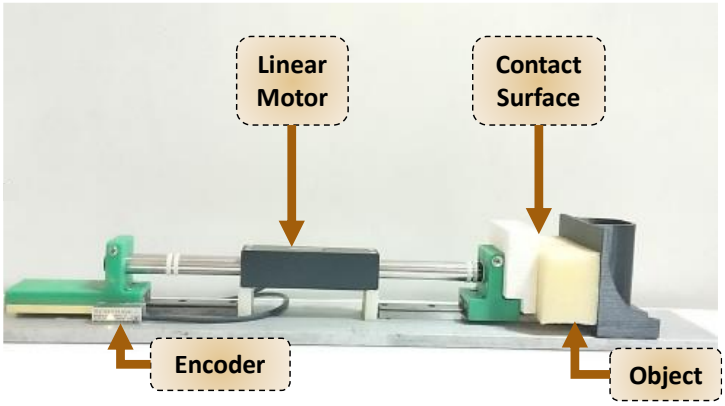


Figure 5.1: Experimental setup

Table 5.1: Experimental Parameters

Symbol	Value
M_n	0.463 kg
K_{fn}	24 N/A
g_{dis}	300 rad/s
g_{rec}	300 rad/s
K_p	2
g_v	30
g_a	100

Some modifications were introduced to the experiment system employed in the acquisition of haptic sensations to utilize in the reproduction phase.

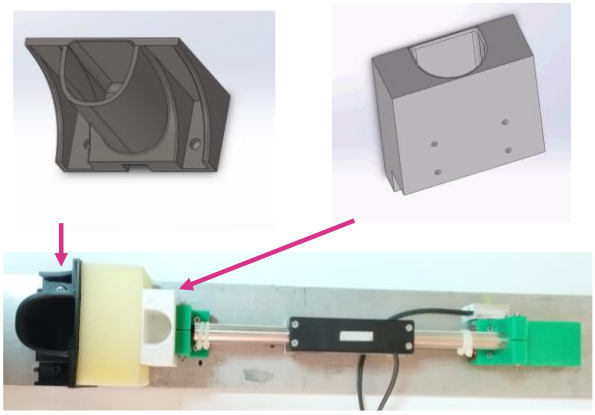


Figure 5.2: Hardware modifications

5.1. Hardware development

5.1.1. Linear motor

Liner shaft motor is employed to achieve linear motion. In the research, the forcer of the linear motor is held stationary while the shaft moves to create a linear motion. There is no restriction on the angle or orientation at which the system can be mounted. The motor is a brushless high precision direct drive linear servo motor with a tubular design. This motor provides user a higher degree of flexibility.

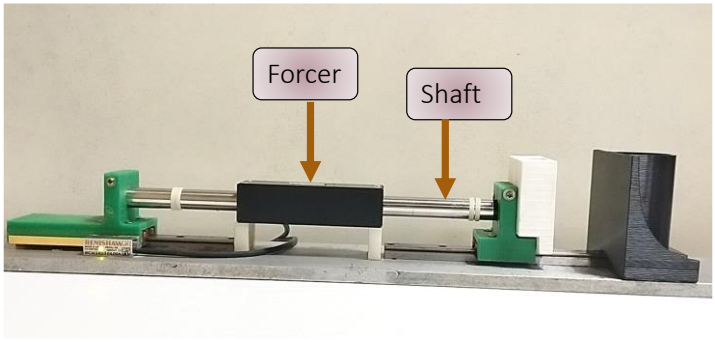


Figure 5.3: Linear shaft motor

The motor has the limit of continuous current of 0.62 A, the current command is generated from the program and the current input taken from the motor driver unit are always kept under safe current range. Furthermore, the continuous force command is always maintained within the acceptable force limits of the motor.

Table 5.2: Specifications of the linear motor

Model	S160T
Continuous Force	15 N
Continuous Current	0.62 A

5.1.2. Linear encoder

The encoder is a non-contact optical encoder system with a set-up led indicator. This is an incremental encoder system which can provide reliable and robust position feedback. This offers proven reliable performance as this is one of the of the most applied encoder systems.

Figure 5.4: Linear encoder

Table 5.3: Specifications of the linear encoder

Model	RGH24D
Resolution	5 μm

Since the encoder can provide the measurements up to an accuracy of 5 μm, the haptic virtual environment has 5 μm sensitivity for the compression depth.

5.1.3. Motor driver

The motor driver operates in the current control mode. A two-phase command through serial communication is received by the motor driver to operate in this mode. This intelligent MOVO servo driver contains the advanced motion control function. The motor driver is working with high flexibility, and it suits widely for a linear

motor, a rotated type of motor such as stepping motor, etc. In this research the motor driver was employed with the linear motor.

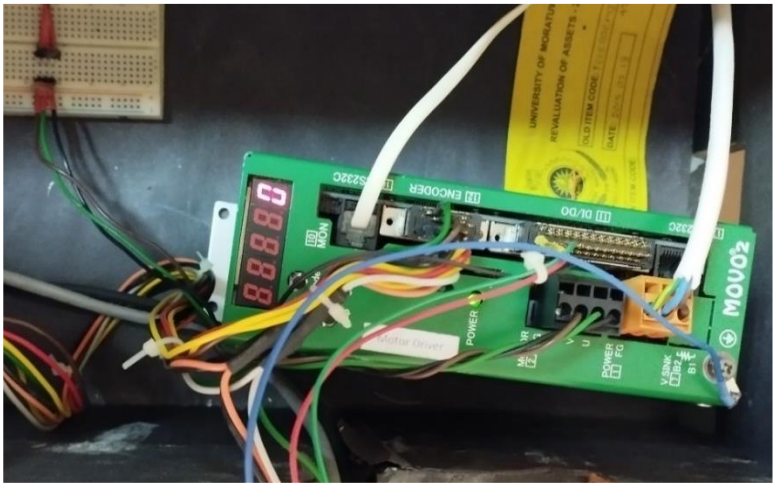


Figure 5.5: Motor Driver

Table 5.4: Specifications of the motor driver

Model	**SVFH1-H3-DSP-SRI**
Power class	200 V single phase
Rated Power	160 W
Rated Current	0.8 A

The hardware block diagram and the parameter list with the set values are given in the APPENDIX-A and APPENDIX-B respectively. The parameter values were set according to the application of use. In this research, a force command is applied from the motor to the object during the abstraction phase. Thus, the motor driver is set to operate in the force/torque control mode which controls the force using the current input. Therefor the mode of the motor driver is set to operate for analog input only. Furthermore, motor length and rated current values are set according to the research requirements and the other used hardware devices.

In the study force controlling is considered. Thus, the motor driver operates in the torque control mode and the operation commands sends through its interface are as follows,

- MASK: Enter the MASK to prioritize MOV/2 to the I/O port.
- SVON: Input the command either from the I/O port or via serial communication.
- G: To initiate the motor operation.
- S: To stop the motor operation.
- SVOF: Servo OFF

Table 5.5: Parameter values configured in the motor driver

Parameter	Value
Encoder resolution	6000 pulses
Linear motor reference length	120 mm
Motor rated current	0.6 A
Maximum output speed	4000 mm/s
Initial mode	ANQ - Analog input only

5.1.4. Analogue/ Digital I/O module | Sensoray 826

Sensoray's model 826 is a versatile analog and digital I/O system on a PCI Express board. It has

- Six 32-bit counters with quadrature decoders and support incremental encoders, frequency/period/pulse measurement and PWM/pulse generation.
- Sixteen 16-bit (300 ks/s) analog inputs,
- Eight 16-bit (900 ks/s) analog outputs,
- 48 digital I/Os with edge detection,
- Multistage watchdog timer with final-stage relay
- Fail-safe output controller,
- Board ID switches for easy device identification and
- A flexible signal router.

The board ID switch allows multiple boards to easily coexist on a backplane. The board is in a compact size, and it has abundant resources and high performance which make perfect for a variety of measurement and control applications such as robotics, motion platforms, scientific instruments, factory automation, and packing and conveyor equipment.

Figure 5.6: Sensoray's Model 826 Multifunction analog/digital I/O

Linux software development kit for Sensoray's model 826 was used to connect the board's hardware resources through its high-level application program interface. Both C/C++ programming languages were used to code the source code. 32-bit counter used to measure the incremental encoder position. It has with high-resolution timestamps (1 μs) which ensures precise measurement. Analog output is configured for 0V – 5V measurement range to output the force as the current command.

Table 5.6: Specifications of the Sensoray's model

Model	Sensoray's Model 826
Counters	
Resolution	32 bits
Analog Output	
Resolution	16 bits
Output voltage range	0V – 5V

6. EXPERIMENTAL RESULTS

6.1. Conventional model-based approach

The spring damper model was the conventional model-based approach considered in this analysis to model the behavior of the object. Thus, the object impedance is defined using stiffness and viscosity parameters and the behavior of these parameters were assumed as constant, exponential, or polynomial.

In this study, different reconstruction methods of object impedance were compared using the approximations for stiffness and viscosity. The stiffness and viscosity were approximated constant or polynomial and the values of coefficients were generated using the Curve fitting tool in Matlab. The approximations considered for stiffness and viscosity in this analysis are,

Case 1: $k = f(x) = \alpha$ (6.1)

$$\boldsymbol{b = f(x) = \beta} \tag{6.2}$$

Case 2: $\boldsymbol{k = f(x) = \alpha_n x^n + \alpha_{n-1} x^{n-1} + \cdots + \alpha_1 x + \alpha_0}$ (6.3)

$$\boldsymbol{b = f(x) = \beta_n x^n + \beta_{n-1} x^{n-1} + \cdots + \beta x + \beta_0} \tag{6.4}$$

where α and β are the real numbers, and $n \epsilon N$,where N is a natural number. Only 1st and 2nd order polynormal approximations were considered with the case of constant stiffness and viscosity. Figure 6.1 shows how the variation of each feature with the other when changing the approximations for stiffness and viscosity.

The performance was evaluated using regression matrices and Table 6.1 outlines the comparison of these regression matrices considered and Figure 6.2 summarizes the same. When comparing R^2 score values, it seems that it increases when considering complex mathematical representation. The regression losses, MAE, MSE, RMSE reduces when increasing the complexity of the mathematical representation as the 2nd order polynomial approximation provide better results than the constant

approximation. Thus, it seems that use of complex mathematical representation for recreating force response shows better results than using simple mathematical representation.

Table 6.1: Mathematical representation for the three cases

Case 1: Constant		$F_e = 280.3x + 2690\dot{x}$
Case 2: Polynomial	**1st Order**	$F_e = (19440x - 281.9)x + (28400x + 840.6)\dot{x}$
	2nd Order	$F_e = (1465000x^2 - 54140x + 584.4)x + (-4663000x^2 + 173700x - 19.08)\dot{x}$

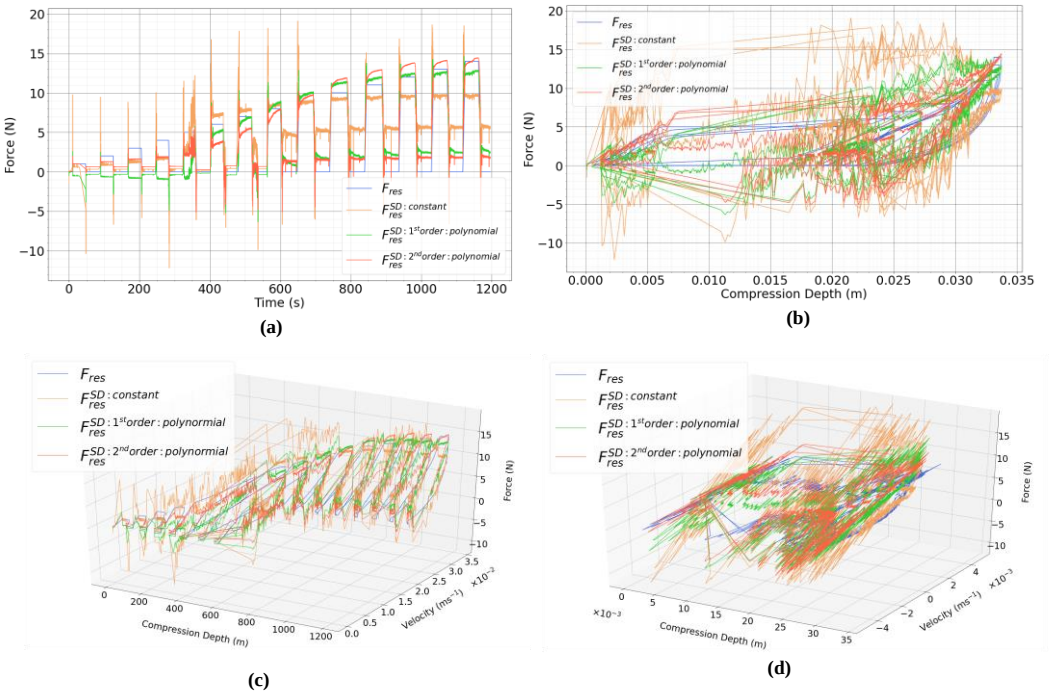


Figure 6.1: a) Force on the object over time b) Force on the object over compression depth c) Force on the object over time and compression depth d) Force on the object over motion parameters.

The profile of force response of 2nd order polynomial approximation for the testing dataset is represented in Figure 6.3. Moreover, the surface of the generated function from the curve fitting tool is shown in the same graph. When observing the graph, it seems that all data points do not lie on the same plane generated by the polynomial relationship and the variation of the distance from the plane the actual

force response was shown using the color range in the same. Thus, the conventional spring damper model fails in representing the behaviour of the object.

Table 6.2: Comparison of Regression Metrics Spring damper models

Regression Metric	**Case 1: Constant**	**Case 2: Polynomial**	
		1st Order	**2nd Order**
R^2 score	0.46	0.82	0.90
MAE	2.96	1.68	1.40
MSE	13.45	4.45	2.44
RMSE	3.67	2.11	1.56

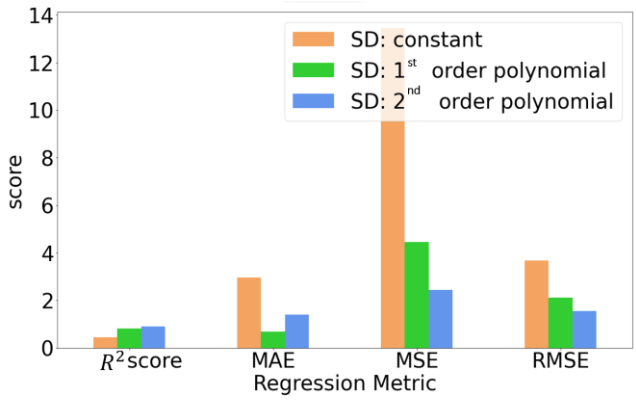


Figure 6.2: Comparison of Regression Metrics of Spring damper models

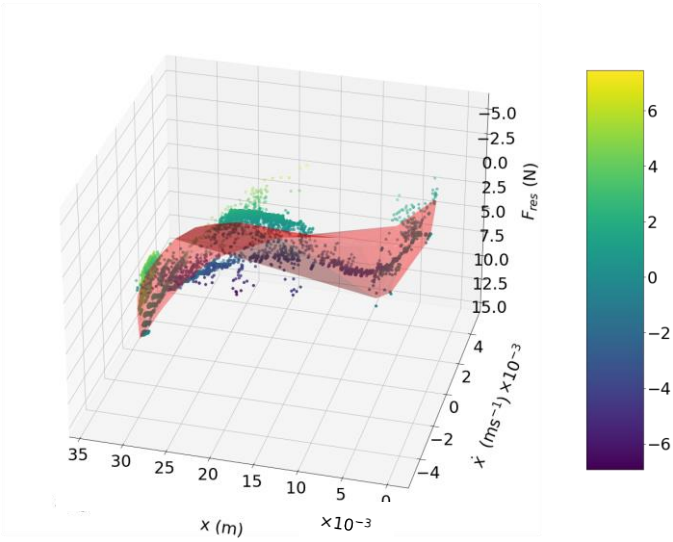


Figure 6.3: Force profile of over motion parameters for the of 2nd order polynomial approximation of stiffness and viscosity

6.2. AI approach is better conventional model-based approach.

A dataset of 5,200,000 samples, which was collected for a ramped force command applied on the sponge was used in this analysis. However, a reduced version of it was used. The train and test data set used for this analysis are summarized in the Table 6.2. The variation of recorded haptic information relative to time and motion parameters are shown in Figure 6.1.

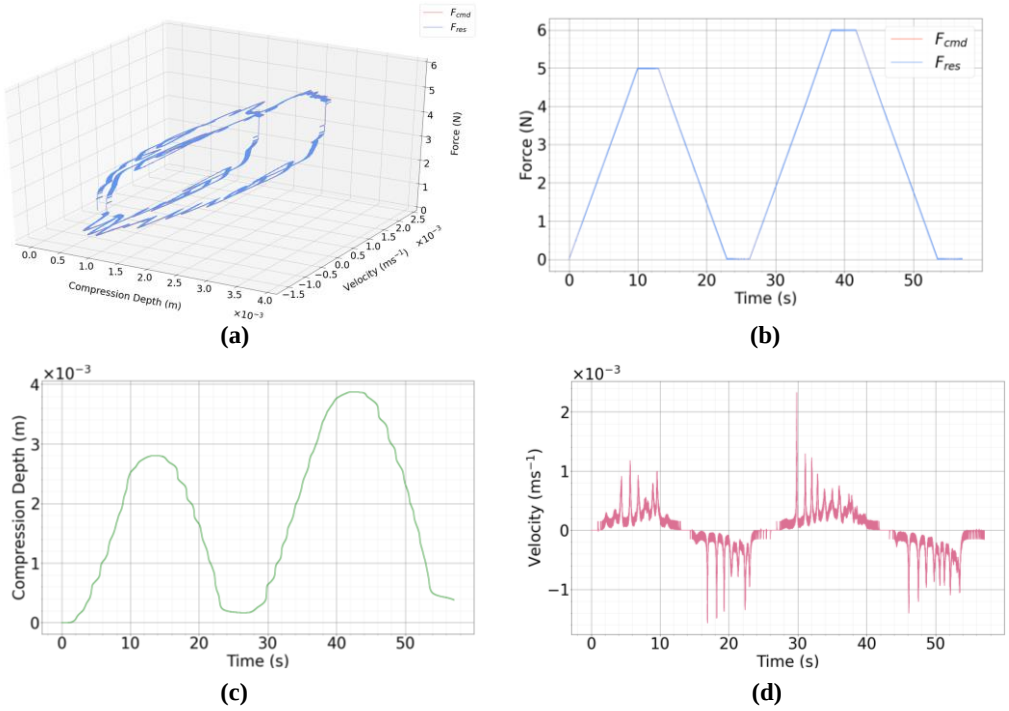


Figure 6.4: Force on the object over motion parameters b) Force on the object over time c) Compression depth on the object over time d) Velocity on the object over time.

Table 6.3: Train - Test Datasets

Dataset	No. of samples
Training dataset	24000
Testing dataset	28000

The spring-damper model was considered as the model-based approach in the analysis. The stiffness and viscosity values were generated using the curve fitting tool

in Matlab by assuming behavior of stiffness and viscosity as constant. The relationship between the force response and motion parameters, compression depth and velocity was derived as represented in (6.5). The force response simulated through the curve fitting tool for the training dataset is shown in Figure 6.5. The surface of the function generated from the tool can be recognized in the same illustration.

$$\boldsymbol{F_{res} = 1624x + 2331\dot{x}} \tag{6.5}$$

The calculated values of force responses of the test set were derived from the relationship shown in (6.5). Thus, all calculated values will lie on the same plane which is created by the function. The graphs in Figure 6.5 shows deviation of the calculated values of force responses, F_{res}^{cal} using the spring-damper model from the actual values of force responses.

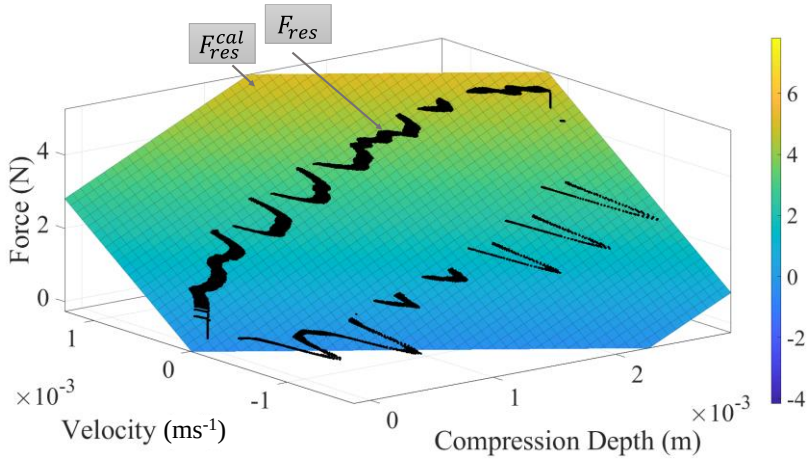


Figure 6.5: Simulated force response from the spring-damper model.

The SVR algorithm was selected to build the virtual environmental model in the AI based approach. However, the relationship between force response and motion parameters cannot be simply defined in the AI approach as spring damper modeling. Hence, the kernel function was utilized to transform the nonlinear feature input space into another space to recognize nonlinear behavior. The hyperparameters of SVR algorithm were tuned to adjustments in its kernel function to have better performance on the training dataset. Table 6.4 shows the specific values of hyperparameters selected to define SVR model to achieve better performance on the training set.

Table 6.4: Hyperparameters of SVR Model

Hyperparameter	**Value**
Kernel Type	'rbf'
Regularization parameter (C)	0.1
Kernel coefficient for 'rbf' (γ)	0.002
Precision (ε)	0.0001

The trained model was utilized to get the predictions for the test samples in the AI approach. The graphs in Figure 6.6 explain how the predicted values of force responses, F_{res}^{pre} using SVR model deviate from the actual values of force responses.

The performance of approaches was evaluated using regression matrices. Table 6.4 shows the comparison of these matrices of both approaches and Figure 6.7 summarizes the comparison. The virtual environmental model should have lower regression losses indicated by MSE, RMSE, MAE, and a higher R^2 score for the model to perform well. When comparing R^2 score values of approaches, it seems that the conventional model-based approach has better performance than the AI-based approach. Since, R^2 score reflects the correlation that indicates linearity, it cannot be considered as a reliable metric in the nonlinear analysis. Hence, RMSE was considered, and it seems that the AI algorithm has a better performance than the conventional spring-damper model to replicate the object's behavior. Furthermore, SVR algorithm is identifying the object up to an accuracy of 82.7% in basis of RMSE as shown in Figure 6.8. Thus, the AI approach is better than the conventional model-based approach when evaluating the nonlinear behavior of responses from the object which cannot be mathematically interpreted with simple relationship. Therefore, it is visible that virtual object modeling using AI is the best approach for object reconstruction.

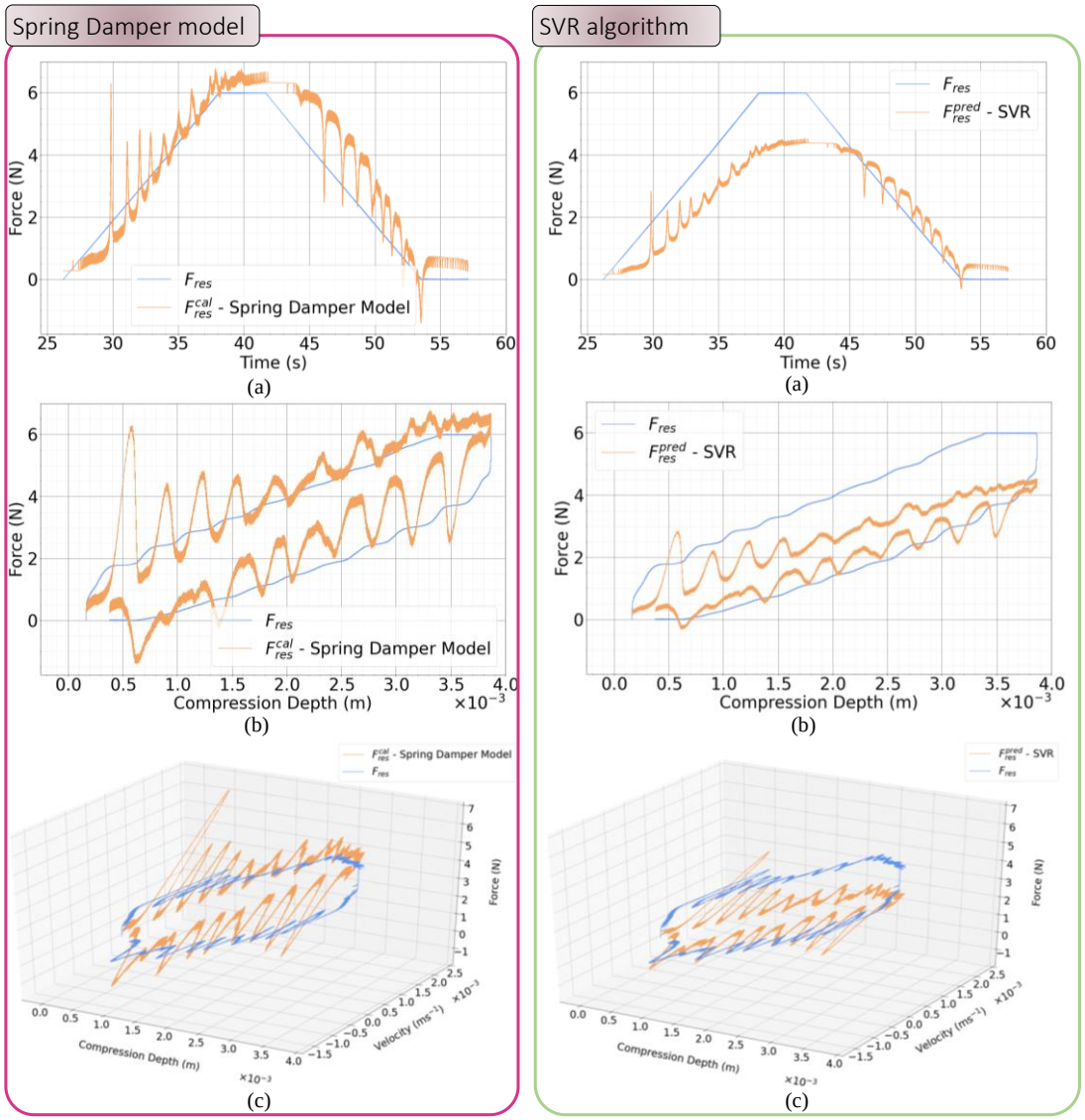


Figure 6.6: Comparison of results for a sponge object a) Force response over time b) Force response over compression depth c) Force response over motion parameters

Table 6.5: Compression of Regression Metrics

Regression Metric	**Spring Damper model**	**SVR algorithm**
R^2 score	0.81	0.75
MAE	0.71	0.15
MSE	0.81	0.03
RMSE	0.90	0.17

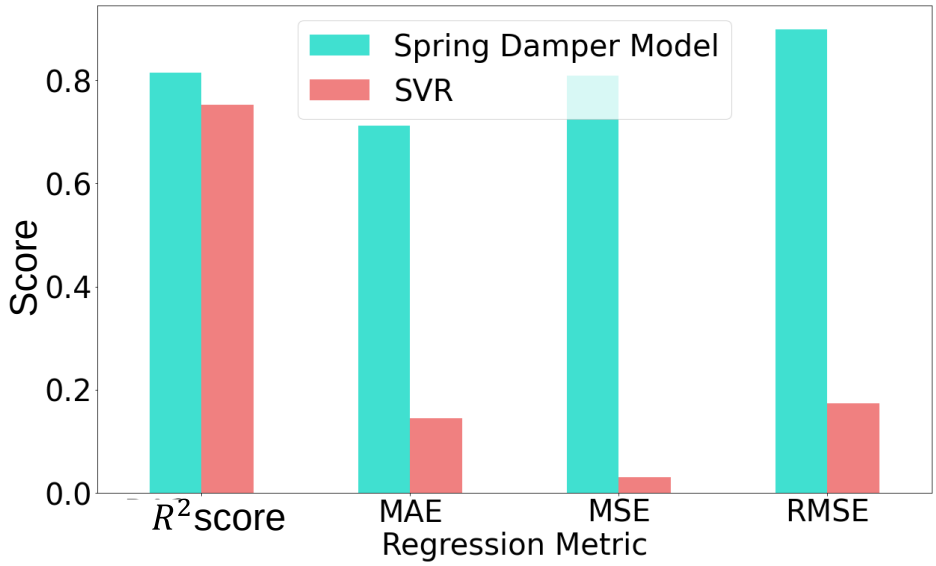


Figure 6.7: Comparison of Regression Metrics for virtual object models

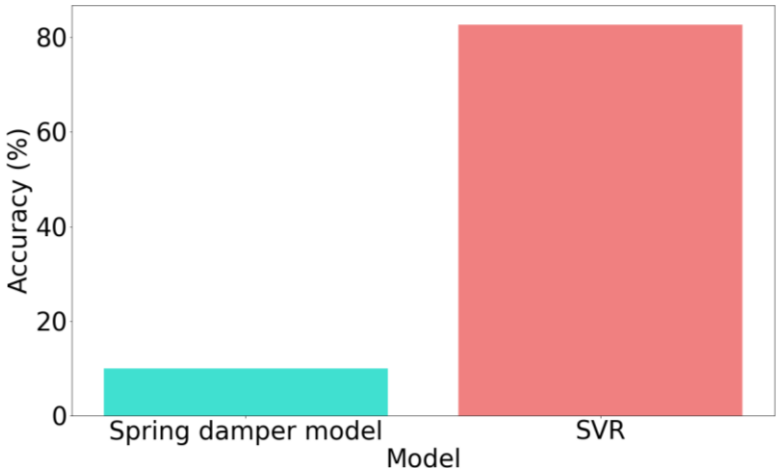


Figure 6.8: Comparison of accuracy for virtual object models.

A Support Vector Regression (SVR) model to identify object to be reconstructed was introduced to predict force response from position and velocity information and it has shown that AI approach provides higher performance than the traditional model-based approach [42].

6.3. Features extraction and important features.

The dataset comprised of a numerical target variable and numerical input data. The popular feature selection techniques that can be used are,

- Correlation Statistics
- Mutual Information Statistics [43]-[45]

Item statistics only convey a simple understanding about the dataset. However, it is essential to preprocess and reformate data to analysis dataset before initiating AI approach. First, the dataset is needed to normalize, and the Standard scaler was used in this study to analyze data with higher degree of transparency. The features are standardized to scale to unit variance by removing the mean. The normalization formula is shown in (6.6)

$$\boldsymbol{y'} = \frac{\boldsymbol{y}-\bar{\boldsymbol{y}}}{\boldsymbol{\sigma_y}} \tag{6.6}$$

where y' represents the normalized value and $\bar{y}$ is the mean of the feature, y. and σ_y is the standard deviation of the feature.

6.3.1. Correlation Statistics

Correlation measures how two variables change together. Pearson's correlation is the most common measure of correlation which assumes a Gaussian distribution to each variable and reflects on their linear relationship.

6.3.1.1. Pearson's Correlation Coefficient

This is a measure of linear relationship, and the score varies between -1 to 1 and 0 represents no relationship performance of approaches was evaluated using regression matrices. The heatmap shown in Figure 6.9 illustrates how strong the relationship between the features is while the graph in Figure 6.10 summarizes the relationship between features and the target variable. It seems that the compression depth has the highest correlation with the response. Compression depth is the only feature with the correlation > 0.4, which considered as the normal benchmark to define moderate correlation. Thus, compression depth should be included in the feature list to be used as the input to the model.

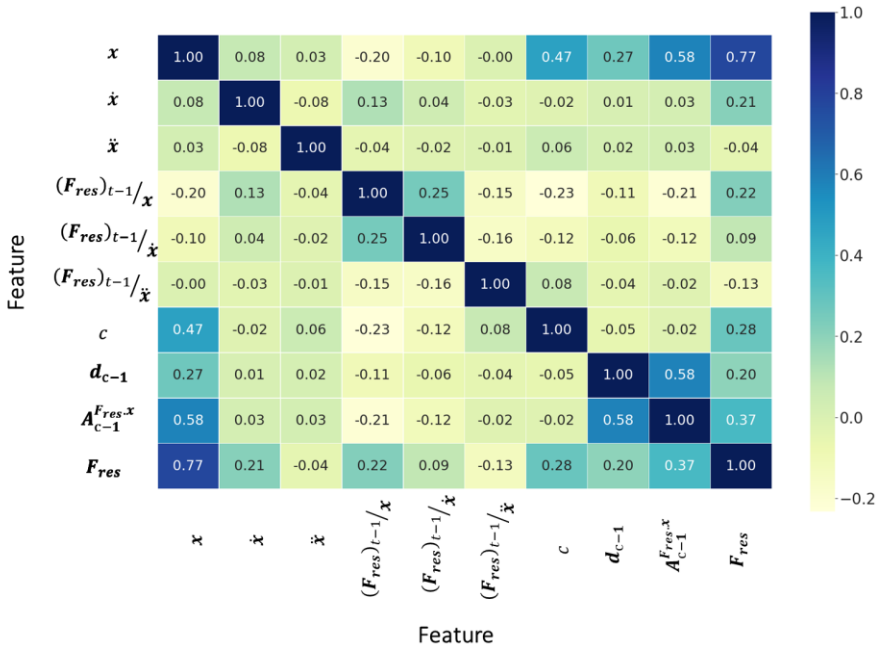


Figure 6.9 : Heat Map of variable correlations

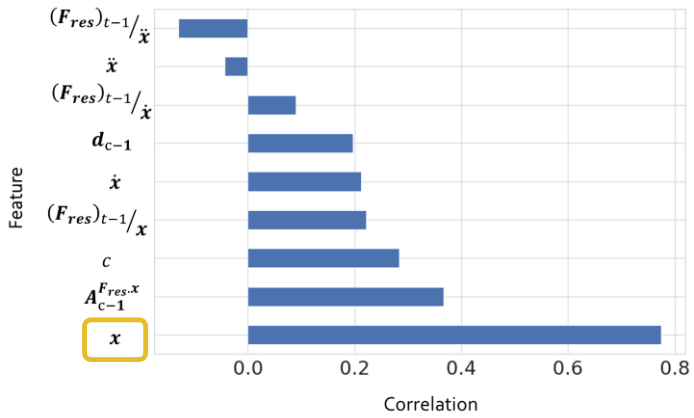


Figure 6.10: Correlation coefficient of features with force response

6.3.1.2. f regression

In most cases the positive score with the bigger the value shows the strength of the relationship between variables and features with larger positive value are more likely selected for modeling.

The linear correlation can be transformed into a correlation statistic that only contains positive values. The f_regression() function provides correlation statistic

implementation. This uses feature selection strategies, such as the SelectKBest class to select the topmost relevant features. First, the r_regression function is used to compute the cross correlation between each regressor and the target. Pearson's r for each feature variable and the target variable from r-regression is also known as the Pearson correlation coefficient. Then the value is converted to an f score and then to a p value. The f value is considered in analysis of variance (ANOVA). It decides the ratio of explained variance to unexplained variance. Explained variance is the variance in the response variable in the model which can be explained by the predictor variables in the model while unexplained variance, error variance refers to the variance of errors. A higher explained variance shows that the model is explained more by the variation in data. The f value determines the p value. The p value decides the significance of the results in relation to the null hypothesis. It is the probability of getting the observed results given that the null hypothesis is true. The features with larger score are considered important and Figure 6.11, the bar graph illustrates the scores for each variable. It seems that the compression depth is the most important feature from the feature variables, and it should not be neglected when building the model.

Table 6.6: Results from f_regreeion()

Feature	**p value**
x	364296.26
$\dot{x}$	11540.47
$\ddot{x}$	447.21
$\frac{(F_{res})_{t-1}}{x}$	12546.32
$\frac{(F_{res})_{t-1}}{\dot{x}}$	1964.22
$\frac{(F_{res})_{t-1}}{\ddot{x}}$	4220.49
c	21398.45
$d_{\mathrm{c}-1}$	9843.58
$A_{\mathrm{c}-1}^{F_{res}.x}$	37866.24

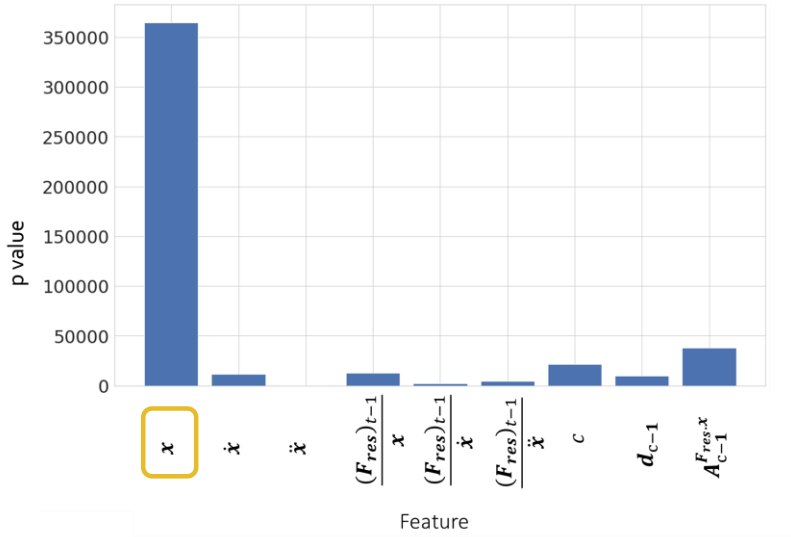


Figure 6.11: Results from f_regreeion()

6.3.1.3. Spearman's Correlation

The Pearson correlation is primarily used to understand the linear relationship between two features. When the variables are related by nonlinear relationship, Spearman's correlation is used, and it consider the dataset as a non-Gaussian distribution. It measures the strength of a monotonic relationship which varying in a way that it either never decreases or never increases. The data is considered monotonic when the one variable increase or decreases the other variable will entirely nonincreasing or nondecreasing.

When examining the spearman's Correlation, through the Figure 6.9, it concludes the $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$ have a highest correlation with the target variable. Furthermore, the parameters, $\ddot{\boldsymbol{x}}$, $\frac{(\boldsymbol{F_{res}})_{t-1}}{\boldsymbol{x}}$, $\frac{(\boldsymbol{F_{res}})_{t-1}}{\dot{\boldsymbol{x}}}$, and $\frac{(\boldsymbol{F_{res}})_{t-1}}{\ddot{\boldsymbol{x}}}$ have a considerable correlation value. Thus, these six features have a considerable impact on target variable than other features. Therefore, the features,

- $\boldsymbol{x}$,
- $\dot{\boldsymbol{x}}$,
- $\ddot{\boldsymbol{x}}$,
- $\frac{(\boldsymbol{F_{res}})_{t-1}}{\boldsymbol{x}}$,

- $\frac{(F_{res})_{t-1}}{\dot{x}}$, and
- $\frac{(F_{res})_{t-1}}{\ddot{x}}$

are the most important features and these features should be chosen as the input features for the AI model.

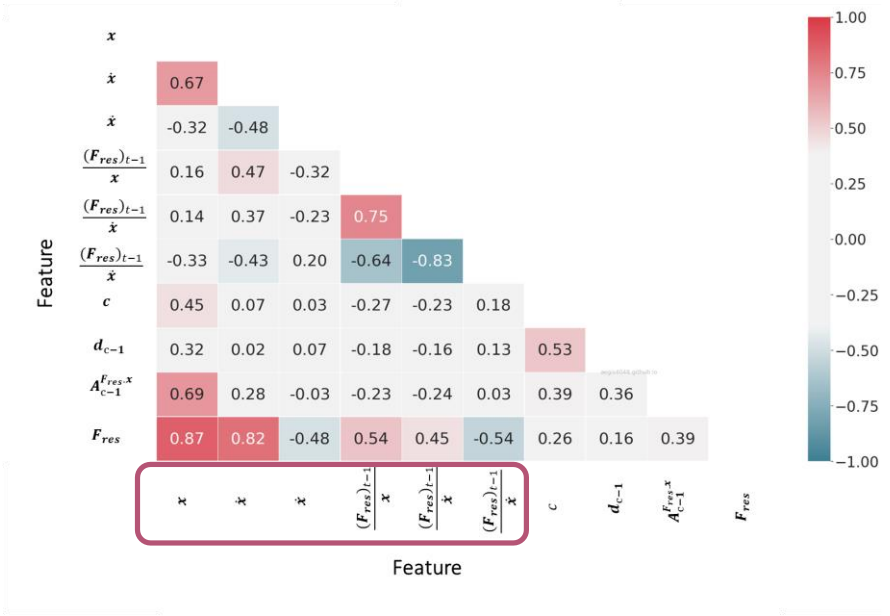


Figure 6.12: Spearman's correlation of features

6.3.2. Mutual Information Statistics

This focuses on estimating mutual information for a continuous target variable. Mutual information is a measure of the dependency between the two variables, and it is a non-negative value. It is the measure of the reduction in uncertainty for one variable given the known value of the other variable. If two random variables are independent, this value becomes zero. The higher value reflects higher dependency. Thus, x the $(F_{res})_{t-1}$over motion parameters namely compression depth, velocity and acceleration show higher mutual information. Therefore, these features cannot be neglected when creating the feature matrix.

Table 6.7: Estimated mutual information.

Feature	Mutual information
x	2.73
$\dot{x}$	1.88
$\ddot{x}$	1.01
$\frac{(F_{res})_{t-1}}{x}$	3.46
$\frac{(F_{res})_{t-1}}{\dot{x}}$	3.43
$\frac{(F_{res})_{t-1}}{\ddot{x}}$	2.66
c	1.46
d_{c-1}	1.63
$A_{c-1}^{F_{res}.x}$	2.01

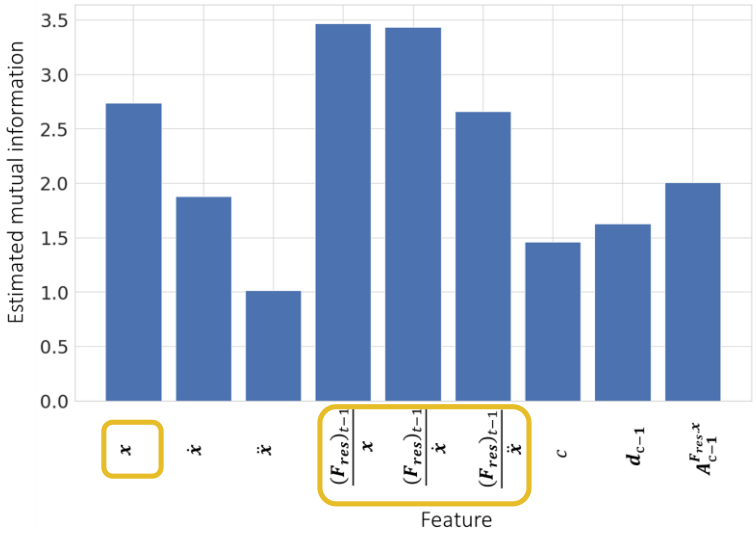


Figure 6.13: Estimated mutual information.

6.3.3. Principal Component Analysis (PCA)

Dimensionality reduction is the main purpose for conducting principal component analysis (PCA). PCA was performed on the datasets to identify the most important data features to be considered and the variance distribution of the principal components (PCs) is shown in Table 6.8. The eigenvalues of covariance matrices and

eigenvectors are used to obtain the principal components and the eigenvalues mentioned in the same table.

Table 6.8: Principal Components (PCs)

PC	Eigenvalues	Variance (%)	Cumulative Variance (%)
1	2.20	24.45	24.45
2	0.21	15.99	40.44
3	1.44	12.75	53.19
4	0.51	11.78	64.98
5	1.15	10.09	75.07
6	0.70	9.17	84.24
7	1.06	7.77	92.01
8	0.83	5.69	97.70
9	0.91	2.30	100.00

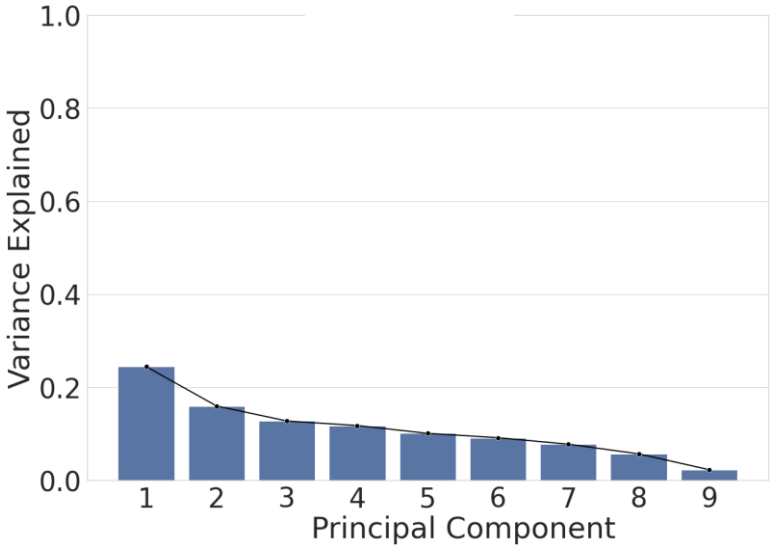


Figure 6.14: Scree plot.

According to the cumulative variance it seemed that seven PC are needed to absorb at least 90% knowledge from the dataset. Moreover, the scree plot which is a line plot of the variance of PCs is shown in Figure 6.14 illustrates the same.

However, when examining the PCA results it seemed that the dataset cannot be represented with fewer number of PCs and almost all the PCs are needed to fully represent the dataset. Thus, use of principal component analysis will not make a considerable impact on reducing computational power.

The results of PCA are summarized in Table 6.9 and significant features with a correlation value greater than 0.60 are highlighted. Thus, it is seemed that x is the most significant feature which has the highest impact on PCs while $A_{c-1}^{F_{res}.x}$ is the lowest significant feature which has lowest impact on the PCs. Thus, it seems that x is the feature most important haptic information.

Table 6.9: Results of PCA

Features	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
x	-0.80	0.00	-0.09	0.50	0.34	-0.10	-0.42	-0.61	-0.80
$\dot{x}$	0.03	0.29	-0.13	0.46	0.44	-0.49	-0.52	0.50	0.41
$\ddot{x}$	0.42	0.51	-0.11	0.23	0.31	-0.26	0.60	-0.32	-0.17
$\frac{(F_{res})_{t-1}}{x}$	0.06	-0.48	0.74	0.03	0.27	-0.42	0.16	-0.02	-0.03
$\frac{(F_{res})_{t-1}}{\dot{x}}$	-0.06	0.57	0.64	0.15	-0.17	0.33	-0.13	0.03	0.01
$\frac{(F_{res})_{t-1}}{\ddot{x}}$	-0.10	0.19	0.01	-0.12	-0.60	-0.62	-0.05	-0.07	-0.07
c	0.15	-0.26	-0.04	0.67	-0.37	0.10	0.11	0.03	0.02
d_{c-1}	0.27	-0.05	0.02	-0.02	-0.01	0.00	-0.32	-0.51	0.27
$A_{c-1}^{F_{res}.x}$	-0.27	0.02	-0.01	0.04	0.01	-0.01	0.18	-0.11	0.29

6.4.AI approach

Several supervised learning regression algorithms were considered to build the AI model. The training dataset was used to train the AI model and the testing set was considered when evaluating the performance of each model to find the best model.

The algorithms considered in this study are,

- Support Vector Regression (SVR)
- Random Forest Regressor

- Linear Regression
- Neural network: Deep Neural network, Recurrent Neural Network

Regression Matrices used to evaluate the performance of these algorithms to choose the best algorithm fit the haptic dataset are,

- R^2 score
- Mean absolute error (MAE)
- Mean squared error (MSE)
- Root mean squared error (RMSE)

6.4.1. Compare same algorithm by changing the input features.

The same algorithm was trained by changing the no. of features in the feature matrix to observe the performance of the model when changing the no.of feature inputs. Random forest algorithm was considered as the algorithm, and the feature matrix was considered as shown below.

Case 1: Feature matrix with only compression depth

$$[\boldsymbol{x}] \tag{6.7}$$

Case 2: Feature matrix with compression depth and velocity

$$\begin{bmatrix} \boldsymbol{x} \\ \dot{\boldsymbol{x}} \end{bmatrix} \tag{6.8}$$

Case 3: Feature matrix with selected features

$$\begin{bmatrix} \boldsymbol{x} \\ \dot{\boldsymbol{x}} \\ \ddot{\boldsymbol{x}} \\ (\boldsymbol{F_{res}})_{t-1}/_{\boldsymbol{x}} \\ (\boldsymbol{F_{res}})_{t-1}/_{\dot{\boldsymbol{x}}} \\ (\boldsymbol{F_{res}})_{t-1}/_{\ddot{\boldsymbol{x}}} \end{bmatrix} \tag{6.9}$$

The random forest algorithm was used to model the AI model by using the feature matrixes defined in above the cases. The variation of the predicted force response and the actual force response are illustrated in Figure 6.15.

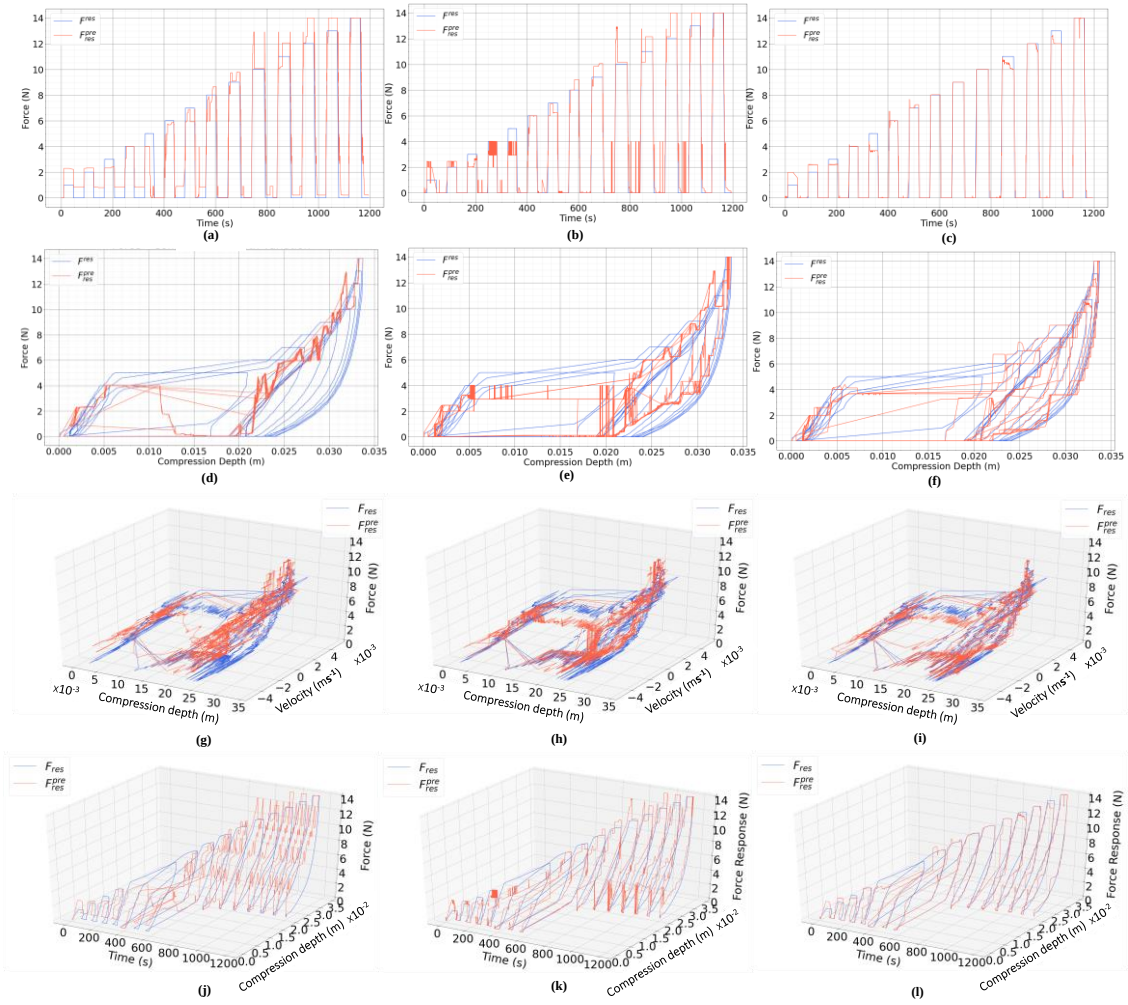


Figure 6.15: Comparison of results for a sponge object - a), b), c) Force response over time d), e), f) Force response over compression depth g), h), i) Force response over motion parameters j), k), l) Force response over compression depth and time for the Case 1, 2, 3

The results obtained are shown in Table 6.8 and it is summarized in Figure 6.16. The model performing well should has a higher R^2 score and lower regression losses denoted by MAE, MSE, RMSE. When comparing these regression matrices, it seems that regression losses namely MAE, MSE, RMSE values reduces while the R^2 score value increases with the increase of the no. of features inputs to the AI model. In this experiment the same AI algorithm, random forest was considered. It seems that even

though the same AI algorithm was used, a higher performance of the AI model can be obtained when increasing the no. of features affecting the target output.

Table 6.10: Compression of Regression Metrices for the 3 cases

Regression Metric	**Case 1**	**Case 2**	**Case 3**
R^2 score	0.97	0.98	0.99
MAE	0.55	0.43	0.16
MSE	0.74	0.59	0.12
RMSE	0.86	0.77	0.34

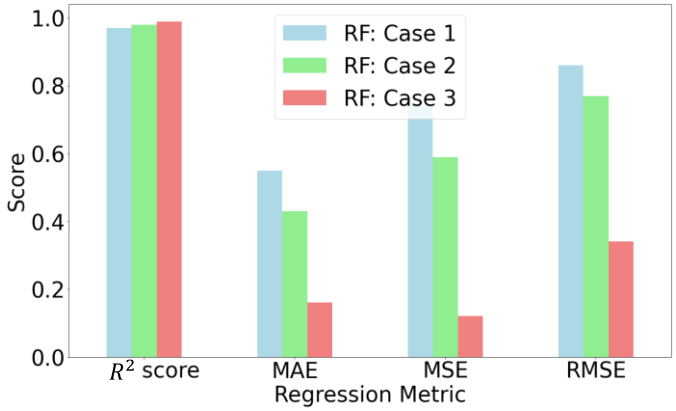


Figure 6.16: Comparison of Regression Metrics for the three cases of random forest models

6.4.2. Comparison of AI algorithms

Different AI algorithms were considered to build the AI model. However, the same feature matrix with the six extracted features was considered as the input for the AI model. The graphs illustrated in Figure 6.17 and Figure 6.18 show how the predicted force response from different AI algorithms varies with the actual force response.

Case 1: LR

Algorithm: Linear regression

Case 2: SVR

Algorithm: SVR algorithm

Table 6.11: Hyperparameters of the SVR Model

Hyperparameter	Value
Kernel Type	'rbf'
Regularization parameter (C)	0.1
Kernel coefficient for 'rbf' (γ)	0.002
Precision (ε)	0.0001

Case 3: RF

Algorithm: Random Forest regression

Table 6.12: Hyperparameters of the random forest Model

Hyperparameter	Value
n_estimators (no.of trees)	1000
max_depth	6

Case 4: NN

Algorithm: Neural network

optimizer = SGD (learning rate = 0.000001)

loss function = MSE

Table 6.13: Hyperparameters of the deep neural network Model

Layer (type)	Output Shape	Hyperparameters
1st dense layer	(None, 150)	units =150 activation = "sigmoid"
1st dropout layer	(None, 150)	rate = 0.2
2nd dense layer	(None, 25)	units =25 activation = "tanh"

2nd dropout layer	(None, 25)	rate = 0.2
3rd dense layer	(None, 1)	units =1 activation = “sigmoid”

Case 3: LSTM

Algorithm: Recurrent neural network – LSTM network

optimizer = Adam (learning rate = 0.000001)

loss function = MAE

Table 6.14: Hyperparameters of the LSTM Model

Layer (type)	**Output Shape**	**Hyperparameters**
LSTM layer	(None, 100)	units =100
Dense Layer	(None, 1)	units = 1

The obtained results were evaluated using the regression metrices and Table 6.9 shows the variation of these matrices with the change of the AI algorithm. Figure 6.19 depicts the same evolution of the performance visually. When observing the graph, it seems that the random forest algorithm shows lower regression losses: MAE, MSE, RMSE than the other algorithms and it has the highest R^2 score too. Thus, random forest algorithm can be chosen as the best matching algorithm for haptic object reconstruction. Hence, this algorithm can be utilized in haptic object reproduction phase.

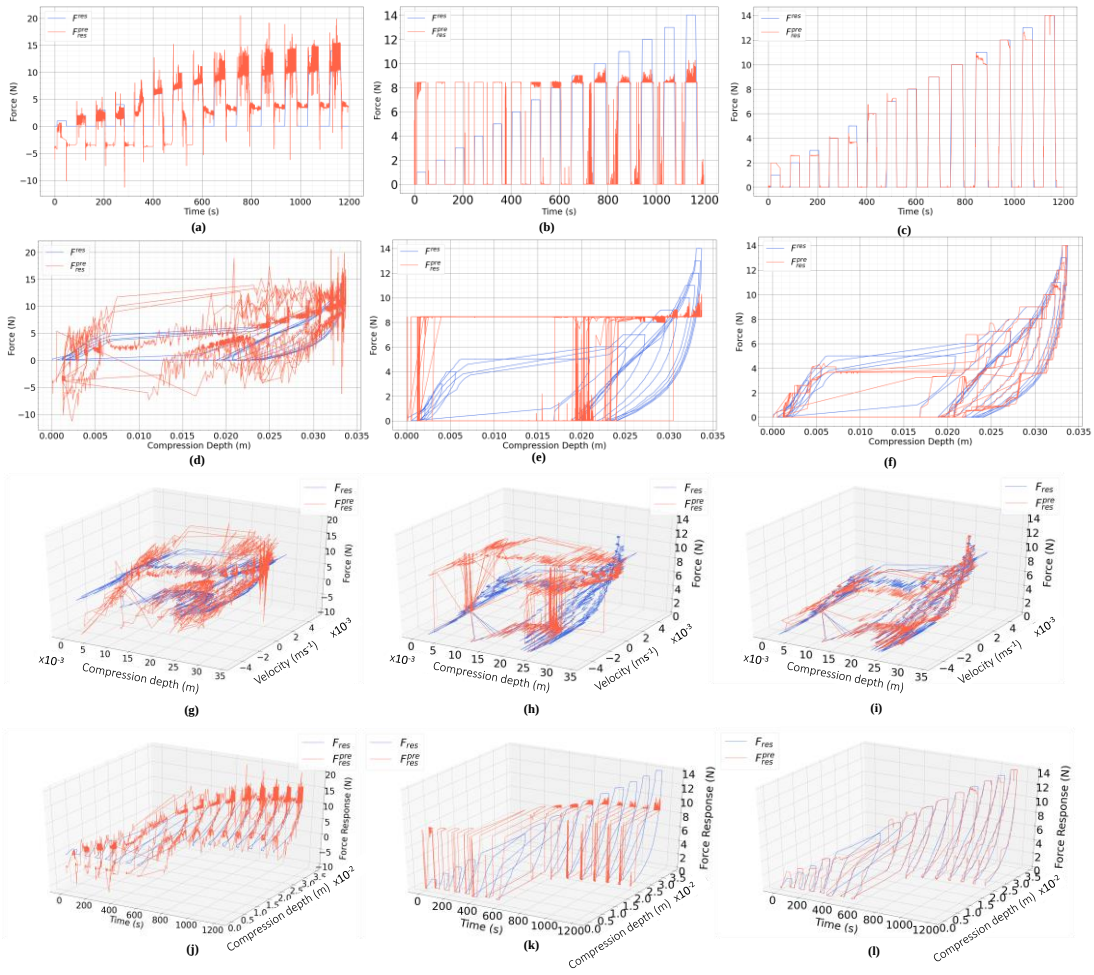


Figure 6.17: Comparison of results for a sponge object - a), b), c) Force response over time d), e), f) Force response over compression depth g), h), i) Force response over motion parameters j), k), l) Force response over compression depth and time for the Case 1, 2, 3 respectively

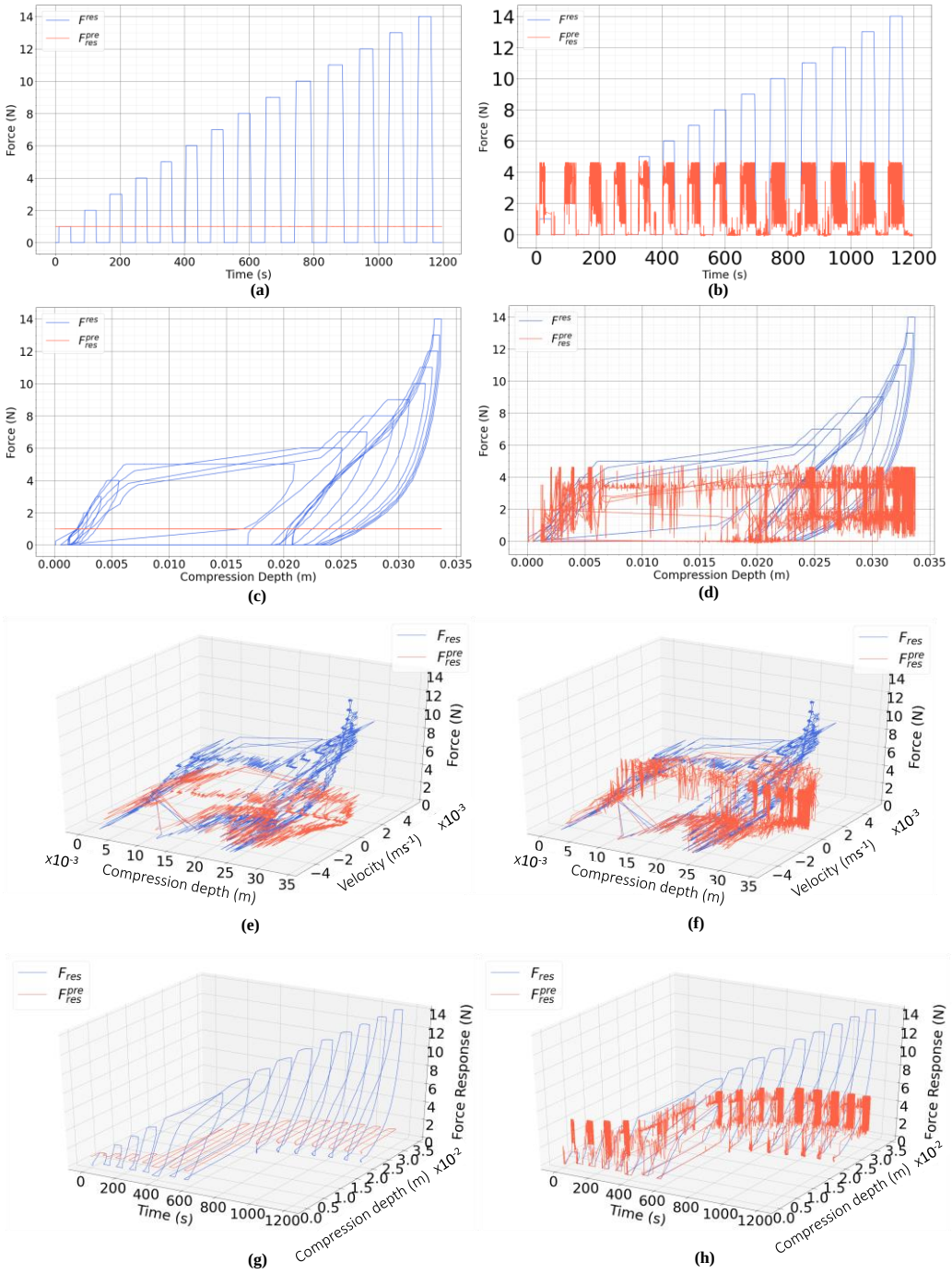


Figure 6.18: Comparison of results for a sponge object - a), b) Force response over time. c), d) Force response over compression depth e), f) Force response over motion parameters g), h) Force response over compression depth and time for the Case 4,5 respectively

Table 6.15: Compression of Regression Metrices of different AI models

Regression Metric	Case 1	Case 2	Case 3	Case 4	Case 5
R^2 score	0.74	0.66	0.99	-0.53	0.09
MAE	2.15	1.89	0.16	4.51	2.98
MSE	6.45	8.59	0.12	38.17	22.64
RMSE	2.54	2.93	0.34	6.18	4.76

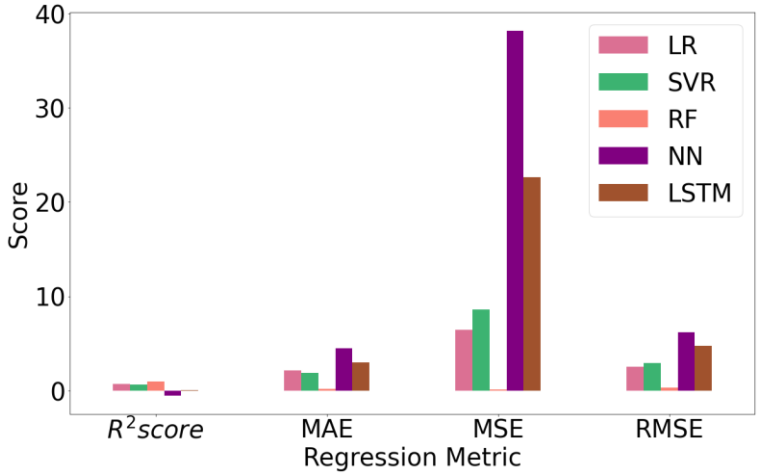


Figure 6.19: Comparison of Regression Metrics for different AI algorithms

The selected algorithm was compared with the conventional spring damper model using regression metrices. Figure 6.20 show this comparison, and it seemed that the random forest algorithm has the best results with a little higher R^2 score and lower MAE, RMSE values. The RMSE value from the random forest model is 0.34 and it is lesser than the RMSE value from the Spring damper model.

Table 6.16: Compression of Regression Metrices of AI model and Spring damper model

Regression Metric	Spring Damper approach	Random Forest Algorithm
R^2 score	0.90	0.99
MAE	1.40	0.16
MSE	2.44	0.12
RMSE	1.56	0.34

The feature importance was found out for the AI model and the comparison of them is shown in Figure 6.21. In case of use of random forest model, the $x, \frac{(F_{res})_{t-1}}{x}$ and $\dot{x}$ show a significant importance and these can be considered as the most important features for building the AI model.

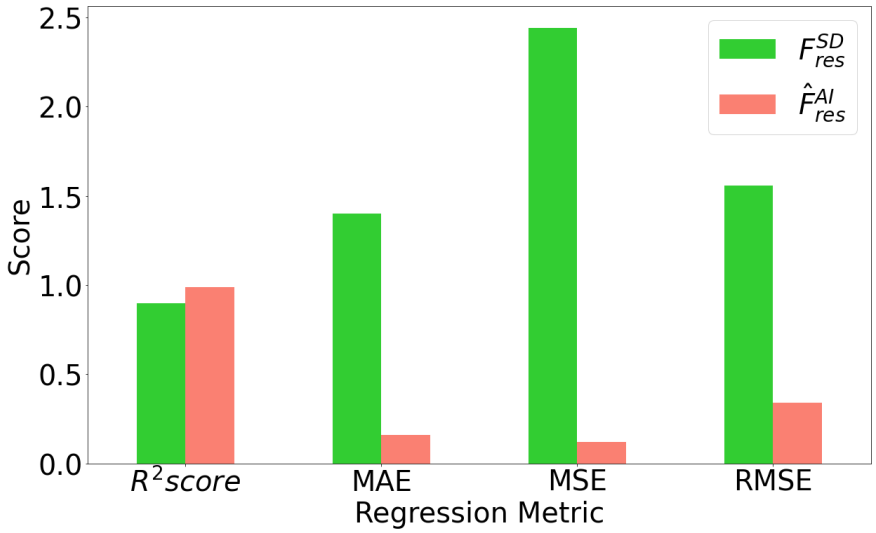


Figure 6.20: Comparison of Regression Metrices of AI approach and Spring damper model

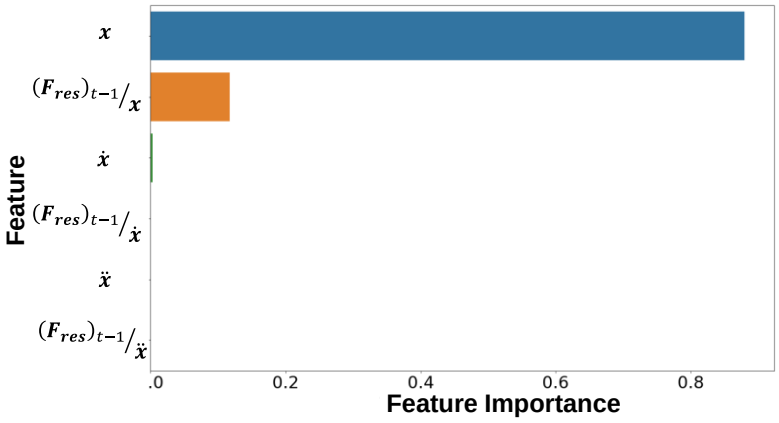


Figure 6.21: Feature Importance

6.4.3. Validate AI approach.

The force was compared for the same position reference command to validate the proposed approach. Position controlling was used in this scenario. The position controlling is illustrated in Fig. (6.22).

Table 6.17: Position control

Parameter	Value
C_p	700.0
C_d	12.0
C_i	1000.0

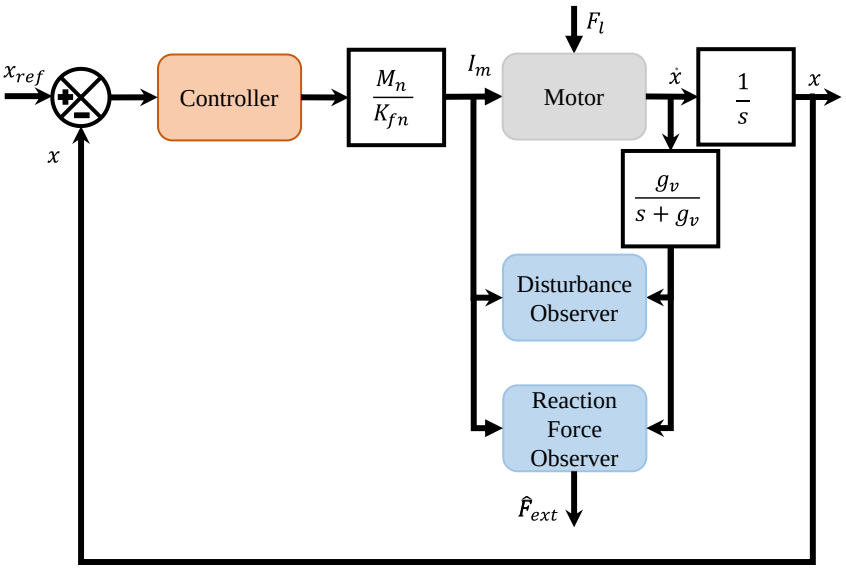


Figure 6.22: Position controlling and force response measurement.

$$x_{err} = x_{ref} - x \tag{6.100}$$

$$dx_{err} = \left(x_{err} - x_{err}^{pre}\right)dt \tag{6.111}$$

$$\int x_{err} = \left(\int x_{err}\right)^{pre} + (x_{err})dt \tag{6.122}$$

$$I_a^{ref} = (C_p x_{err} + C_d dx_{err} + C_i \int x_{err}) \frac{M_n}{K_{fn}} \tag{6.133}$$

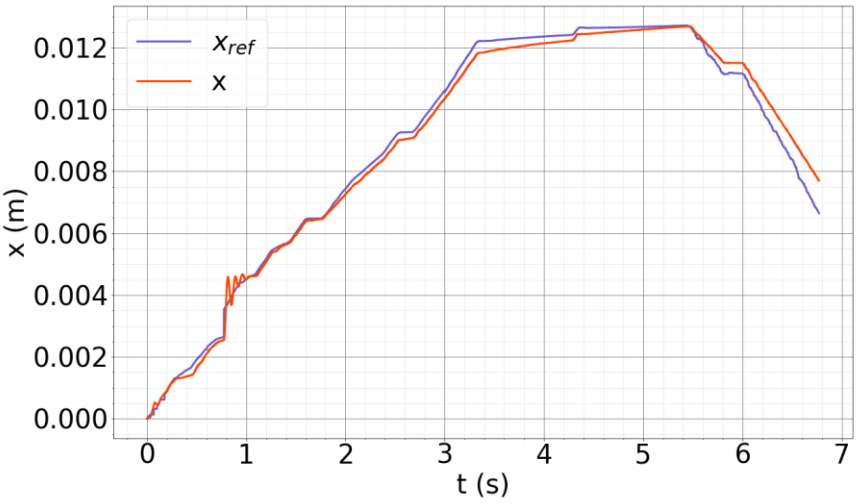


Figure 6.23: Position control

The variation of the force response for the same position variation is illustrated in Fig. (6.24) and comparison of regression metrics is shown in Fig. (6.25). When observing the values, it seems that regression losses are at lower values which leads to these regression metrics prove the validity of using AI for the object reproduction in virtual reality.

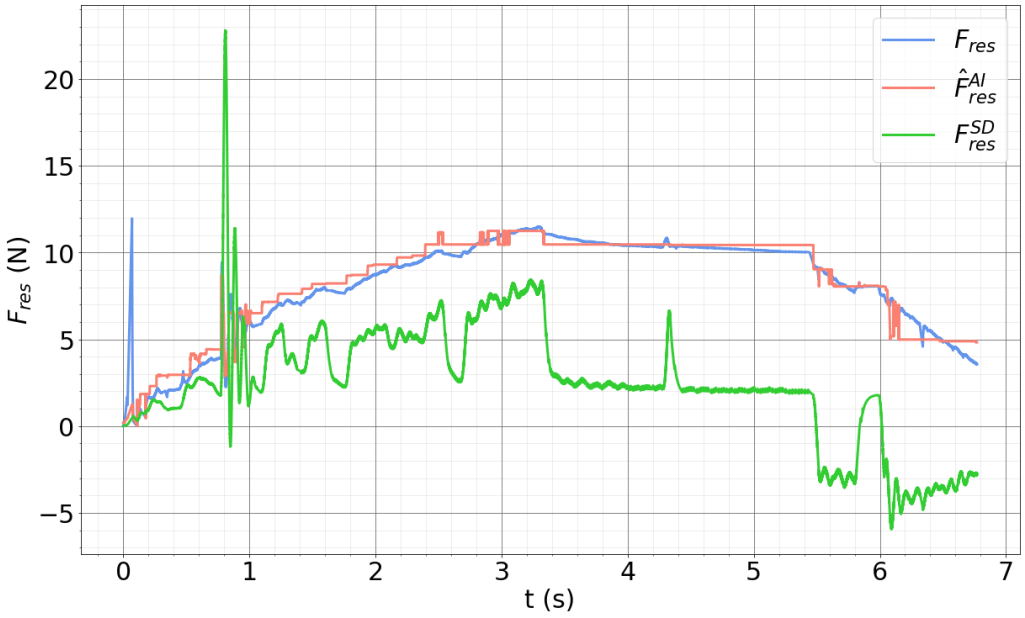


Figure 6.24: Comparison of force responses over time for the sponge real object and virtual object

Table 6.18: Compression of Regression Metrices

Regression Metric	Score
R^2 score	0.95
MAE	0.42
MSE	0.33
RMSE	0.57

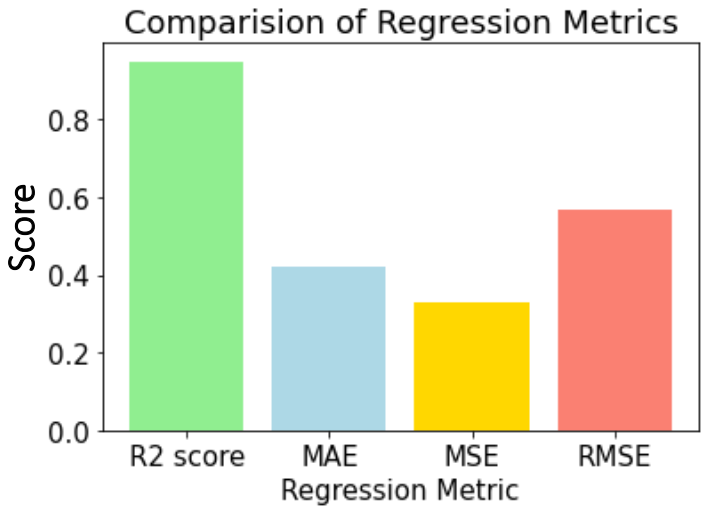


Figure 6.25: Compression of Regression Metrices

6.4.4. Utilize AI model for haptic object Reproduction.

The model was utilized in the real haptic object reproduction. The model was created and converted to PMML, the intermediate model format and integrated the model with hardware using cPMML, C library. The obtained data for a real force application on sponge was recorded and the obtained the response values by the AI model and the calculated force responses for the spring damper relationship to compare the performance of both approach for the real interaction.

Figure 6.26 shows the comparison of actual force response with the predicted force response from the AI approach and the calculated force response from the spring damper model. In this case, spring damper behaviour was assumed by predefining the stiffness and the viscosity as a 2nd order polynomial while the

selected AI algorithm, random forest algorithm was considered in AI approach. It seems that the calculated force response values drastically change while the predicted force values have a closer variation as the actual force response variation. This is also proven when analyzing the regression metrices as outlined in Table 6.19 and the bar graph shown in Figure 6.27. It seemed that regression losses are at a minimum level for AI approach than the conventional approach.

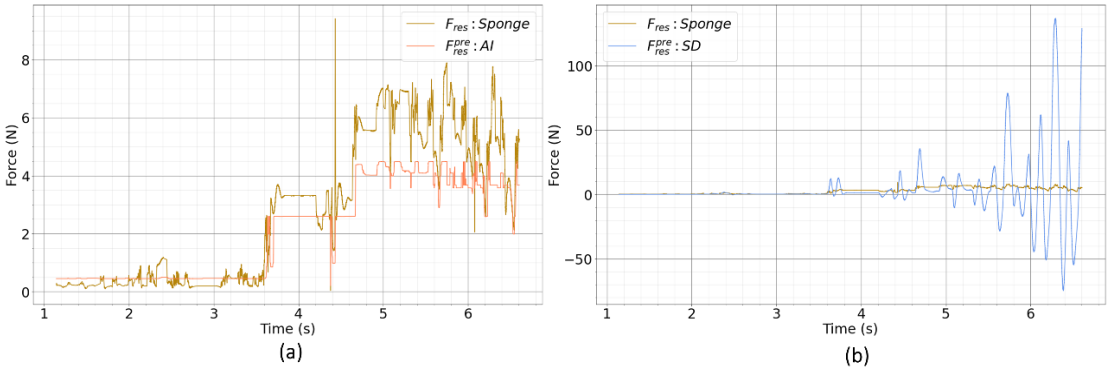


Figure 6.26: Comparison of force responses over time a) Predicted force response from AI approach and force response over time c) Calculated force response from spring damper approach and force response from the sponge over time.

Figure 6.23 illustrates the regression metrices only for the selected AI-based approach. Experimental setup at abstraction and reproduction phase is shown in Figure 6.24.

Table 6.19: Compression of Regression Metrices of AI model and Spring damper model

Regression Metric	**Spring Damper**	**AI approach: Random Forest**
R^2 score	-71.58	0.76
MAE	8.71	0.84
MSE	438.83	1.46
RMSE	20.95	1.21

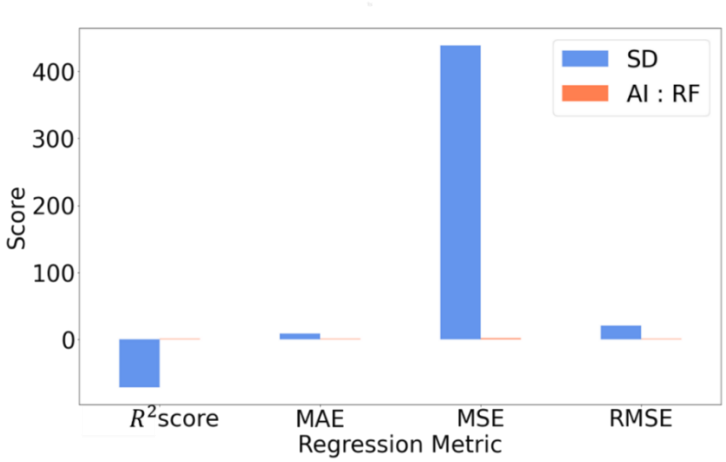


Figure 6.27: Comparison of Regression Metrics for AI approach and Spring damper model

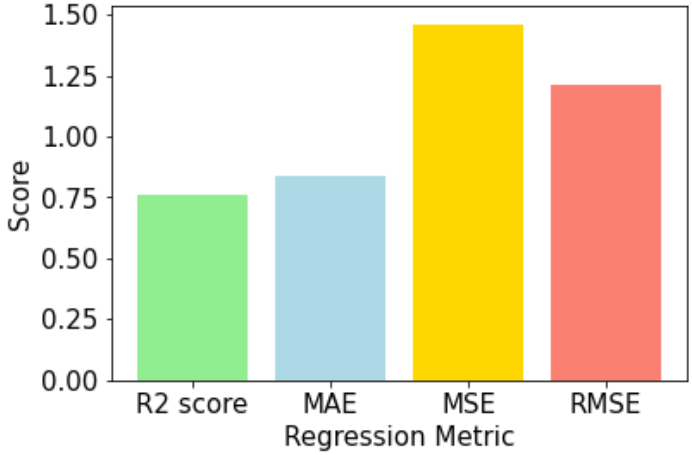
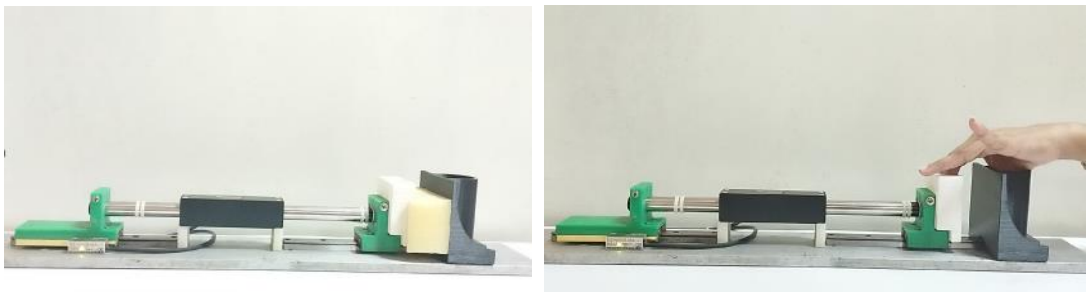


Figure 6.28: Comparison of Regression Metrics for AI approach



(a) **(b)**

Figure 6.29: Experimental setup at 1) Abstraction phase: Squeezing the actual object b) Reproduction phase: Squeezing the virtual object

7. AI ALGORTIHMS FOR PREDICTION OF HAPTIC SENSATION

The proposed AI approach is based on multi-label regression and uses supervised learning. Furthermore, the feature matrix and the output target variable contain only numerical data. Thus, supervised learning regression algorithms are considered in building AI model to replicate the actual environment. The AI algorithms considered in this study are,

- Linear regression
- Random Forest
- Support vector regression (SVR)
- Neural Networks

In this study, python sklearn [46] implementation of the SVR algorithm was utilized.

7.1. Linear Regression

Linear regression is the most common regression algorithm used in building ML models using AI. This uses the linear representation of input values, X the predicted output values for set of input values y. The hypothesis function for linear regression model can be represented using the matrix notation as,

$$\boldsymbol{y = \gamma X + \varepsilon} \tag{7.1}$$

where X is the matrix comprised of input independent variables while y is the vector of the target variable. γ is the coefficient vector of X and ε is the error variable and it is a random non observable variable which is expected to keep zero when fitting the model [47].

The cost function for a linear regression can be root mean squared error or mean squared error. Thus, to achieve the best fit model the cost function value which the error between predictor value,$\hat{y}$ and the target actual value, y should be minimized.

7.2. Random Forest

Random Forest Regression is an ensemble learning method for regression. Ensemble learning method combines predictions from multiple machine learning algorithms to make a prediction more accurate than a single model. Random Forest constructs several decision trees during training phase and provides the output as the means of the prediction of all the trees [48]. Random forests can rank the importance of variables in regression problems via measures of significance.

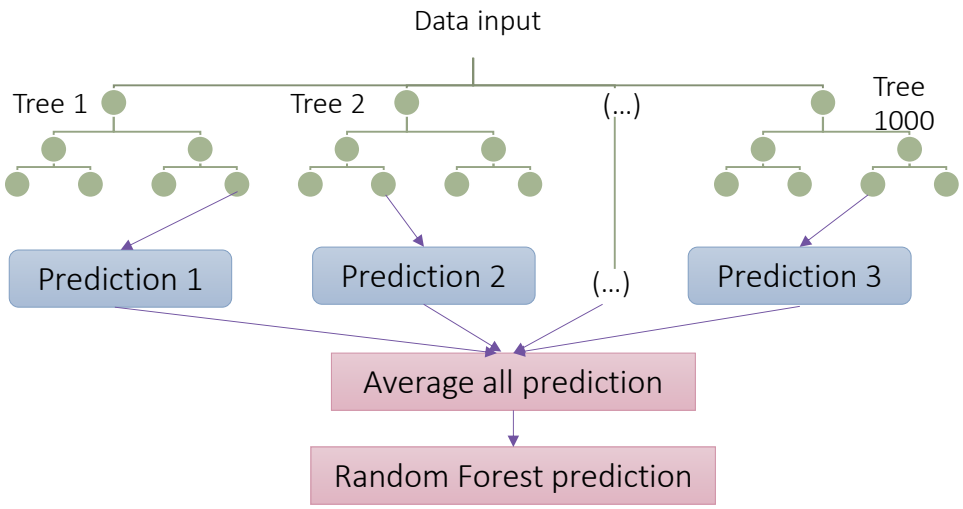


Figure 7.1: Representation of Random Forest

This study shown Random forests was the best performing algorithm to build the AI model.

7.3. Support Vector Regression (SVR)

Support Vector Regression (SVR) [49], [50] is a nonlinear regression model which convex and it guarantees to produce a global solution. It is a kernel-based learning algorithm and a very effective method for learning nonlinear complex functions. For the given training dataset $\{(a_1, b_1), \ldots, (a_n, b_n)\} \subset \mathcal{A} \times \mathbb{R}$ and where $\mathcal{A} \in \mathbb{R}^2$ indicates the input feature space, the SVM algorithm can be defined as is (7.2).

$$\boldsymbol{f(a, w) =< a, w > + d} \tag{7.2}$$

where scaler $d \in \mathbb{R}$, vector $w \in \mathcal{A}$and $<.,.>$ indicates the dot product in A. This algorithm has convex optimization. Its minimization function can be represented as in (7.3).

$$\boldsymbol{L = MIN\ \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)} \tag{7.3}$$

subjected to,

$$b_i - < a_i, \mathrm{w} > - d \leq \varepsilon + \xi_i$$

$$< a_i, \mathrm{w} > + d - b_i \leq \varepsilon + \xi_i^*$$

where the precision is denoted by ε, slack variables, $\xi_i, \xi_i^* \geq 0$ and the constant which determines the trade-off between the regression model's complexity, $C>0$. The kernel functions are used to transform the feature vectors to another space to overcome nonlinearity. Radial basis function (RBF) is the popular kernel function, and it was used in this study as shown in (7.4).

$$\boldsymbol{K(a_i, a_j) = \exp(-\gamma||a_i - a_j||^2)} \tag{7.4}$$

where the kernel coefficient for RBF, $\gamma>0$. The hyperparameters were tuned to define the model which is performing better on the training set. Then, the defined model was utilized to train the SVR model for the object using the training set. Finally, the trained model was employed to predict the force responses, F_{res}^{pred} for the testing set.

7.4. Deep neural network (DNN)

Deep learning architectures can be used to accurately predict the haptic sensation. These are mainly based on neural networks and different types of architectures are available to train any datasets.

The backpropagation (BP) algorithm is the most used algorithm for training the feedforward neural network as shown in Figure 7.2 [51]. It is an iterative process which fine-tunes the weights of a neural network based on the error rate or loss

obtained from the previous iteration which means the epoch [52]. Lower error rates can be achieved through proper tuning of the weights, and it leads to making the model reliable by increasing the degree of generalization. Stochastic gradient descent (SGD) is the commonly used optimization method to adjust the weights by minimizing the error at the output layer [52].

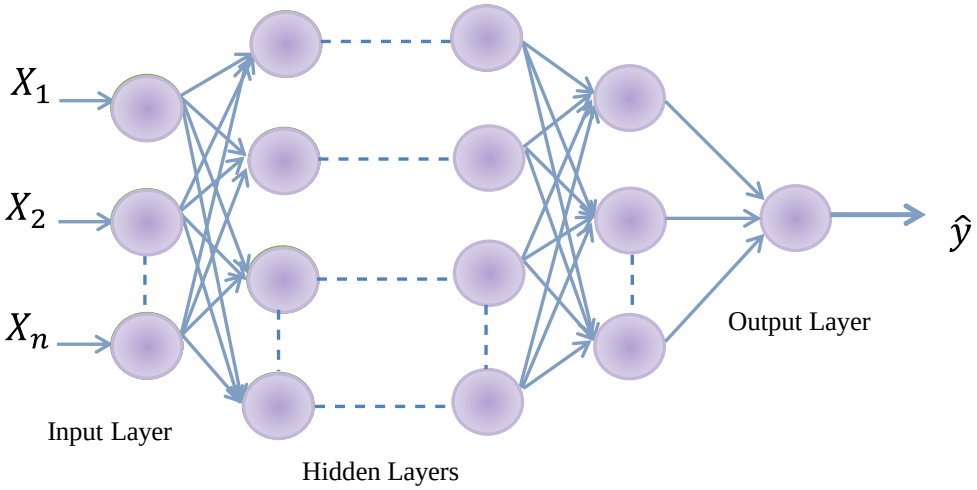


Figure 7.2: Deep learning architecture based on neural network.

The weights of the NN using the following relationship.

$$\Delta \boldsymbol{w_{ij}^{k}}(\boldsymbol{t}) = -\propto \frac{\boldsymbol{\partial}(\boldsymbol{E}(\boldsymbol{X},\boldsymbol{\theta}))}{\boldsymbol{\partial w_{ij}^{k}}}(\boldsymbol{t}) \tag{7.5}$$

where, w_{ij}^{k} is the weight between node i & j in the layer $(k-1)$ and k respectively and $E(X,\theta)$ is the error term which can be derived by using the desired output and the calculated output and $\propto$ is the iteration rate of the NN.

7.4.1. Recurrent neural network (RNN)

RNN is the time-series version of ANN as it analyzes data in a sequence [53]. It has feedback connections which allow flow activations to happen in a loop [53]. The network can perform temporal processing which allows sequence learning. RNN architectures come in a variety of shapes and sizes, and the typical multi-layer

perceptron is the most frequently used. These networks use memory and sophisticated non-linear mapping capabilities.

7.4.1.1. Long short-term memory networks (LSTM)

LSTM networks consist of memory capacity that can preserve the state over long periods. Thus, the major drawbacks of RNN, which is the vanishing gradient problem can be solved by using the LSTM networks [54]. A typical LSTM network consists of various memory blocks called cells. The cell state, C_t and the hidden state h_t are two states that pass to the next cell. The information is stored in memory blocks and the three types of gates are responsible for memory manipulation [54].

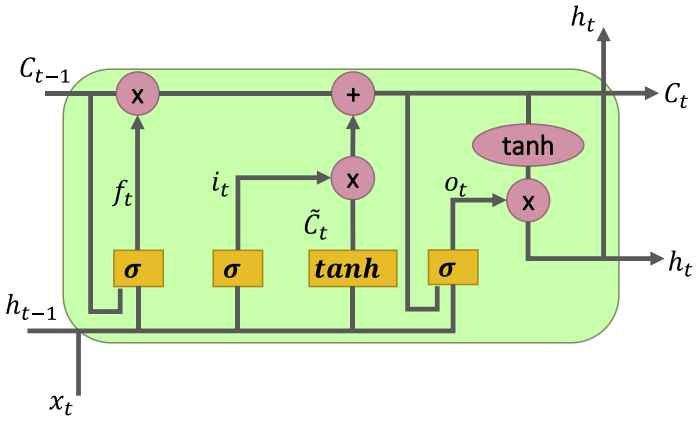


Figure 7.3: LSTM Network

where x_t, f_t,and o_t are the input gate, forget gate, output gate respectively and $\tilde{C}_t$ is the cell update.

The data which is less important or no longer required are removed from the cell state. This is done by the forget gate layer, a sigmoid layer. It observes the values of h_{t-1} and x_t to generate a value between the range of 0 and 1, corresponding to each cell state C_{t-1}. The value '1' indicates to keep completely while the value '0' indicates to completely forget. Then, the cell state is multiplied by this output vector generated from the sigmoid function. Then f_t can be calculated using (7.6).

$$\boldsymbol{f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right)} \tag{7.6}$$

where h_{t-1} is the hidden state of the previous cell. The input gate layer is used to select the new data which is to be stored in the cell. The values which should be updated are determined by sigmoid layer, and a vector $\tilde{C}_t$ which is formed by the tanh layer contains every possible value that can be included. Then the product of the output value of the sigmoid gate with the created vector is taken and added to the cell state by the operation. i_t , and $\tilde{C}_t$ of the input gate is calculated by using.

$$\boldsymbol{i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)} \tag{7.7}$$

$$\boldsymbol{\widetilde{C}_t = tanh(W_c[h_{t-1}, x_t] + b_c)} \tag{7.8}$$

Then the data is added to the cell state. Then the new cell state, C_t can be derived using C_{t-1} , the old cell state as shown in (7.9). Finally, the output is decided, and it depends on the cell state and it's also a filtered version. The output can be identified by executing using the data through the sigmoid layer. Then, as illustrated in equations (7.10) and (7.11), cell state is inserted via the tanh layer and multiplied the output by the sigmoid layer to determine the final output.

$$\boldsymbol{C_t = C_{t-1} \times f_t + i_t \times \widetilde{C}_t} \tag{7.9}$$

$$\boldsymbol{O_t = \sigma(W_o[h_{t-1}, x_t] + b_o)} \tag{7.10}$$

$$\boldsymbol{h_t = O_t \times tanh(C_t)} \tag{7.11}$$

The data discarded from the network and dependency control on individual input are controlled by the values of i_t, f_t,and $\tilde{C}_t$. Also, it is helpful in minimizing the effect of vanishing gradient. Memory allocation to each cell and modification of inputs with new inputs help to maintain the dependencies from the previous input and for the conservation of state during long training session [54].

7.5.Machine learning frameworks.

Various kinds of frameworks and libraries can be used to simplify the training algorithms, depending on the task [55]. Some popular frameworks used in this study are discussed below.

7.5.1. TensorFlow

This is the most popular open-source platform which is JavaScript-based and is a collection of many tools and resources to support the learning process. TensorFlow's Core tool is used to create and execute AI models in browsers, whereas TensorFlow Lite is used with mobile and embedded devices, and TensorFlow Extended is used to train ML or deep learning models in large production systems. TensorFlow can be employed with a variety of programming languages, including JavaScript, Java, C++, and C# but Python is the most popular language [56]. TensorFlow is mainly used to train various kinds of DNN models.

7.5.2. Keras

This is an open-source python deep learning library [57]. It is a high-level, user-friendly API which makes learning and model building process of deep learning algorithms easy. Moreover, it supports multiple GPUs and distributed training. Keras can be used for various applications by integrating with other frameworks and TensorFlow is most used with Keras [57].

7.5.3. Scikit-learn

This is an advanced and efficient yet simple framework that can be used with supervised and unsupervised tasks [46]. It is based on NumPy, SciPy and matplotlib libraries for Python language and often used with lower level, lesser complex data science tasks than deep learning [58].

7.6.Analysis of Features

7.6.1. Variance, Covariance and Correlation

These are the common statistical measures that are considered to understand the relationship within the features of the feature matrix. The dispersion of a data set variable around its mean value is defined as the variance.

$$\boldsymbol{\sigma_Y^2 = E[Y - E[Y]]^2} \tag{7.12}$$

where σ^2 represents the variance of the variable, y while $E[Y]$ is the expected value similar to the mean.

Covariance and Correlation are the basic techniques that is used in data analysis to understand the relationship between variables. These statistical measures explain how two variables change together. The covariance of two variables reveals how they differ, and it is a measure of directional relationship between two variables [43]. The correlation shows how two variables are related. The Pearson correlation coefficient is the most used and it assumes a Gaussian distribution to each variable and measures the strength of the linear relationship between two features. Correlation coefficient is typically a value between -1 and 1 with 0 representing no relationship.

- Positive Correlation: both variables change in the same direction, proportionally.
- Neutral Correlation: No identifiable relationship between the variables.
- Negative Correlation: Inversely correlated which means the variables change in opposite directions.

The covariance and the correlation can be expressed as in (7.13) and (7.14).

$$\boldsymbol{cov}\,(\boldsymbol{X},\boldsymbol{Y}) = \boldsymbol{E}[(\boldsymbol{X} - \boldsymbol{E}(\boldsymbol{X}))(\boldsymbol{Y} - \boldsymbol{E}(\boldsymbol{Y}))] \tag{7.13}$$

$$\boldsymbol{corr}\,(\boldsymbol{X},\boldsymbol{Y}) = \frac{\boldsymbol{cov}\,(\boldsymbol{X},\boldsymbol{Y})}{\boldsymbol{\sigma_X \sigma_Y}} \tag{7.14}$$

where $cov\,(X,Y)$ and $corr\,(X,Y)$ represents the covariance and correlation for the pair of variables, X and Y.

However, the Pearson correlation coefficient is primarily used to understand how strong the linear relationship between two features is. Thus, when variables are related by nonlinear relationship, and considered as a non-Gaussian distribution, Spearman's correlation is used [43]. This measures the strength of a monotonic relationship. The data is considered monotonic when the one variable increase or decreases the variable will increase or decreases.

7.6.2. Mutual Information

This relies on nonparametric methods that are based on entropy estimation from k-nearest neighbors' distances [59],[60] and the idea base for these methods were originally proposed in 1980's [61]. This is a measure of the reduction in uncertainty of a variable when given a known value of the other variable and is calculated between two variables. Feature selection can be done using this information and the larger the value is better.

7.6.3. Principal Component Analysis (PCA)

Introducing PCA is considered as the beginning of AI. PCA is a dimensionality reduction method which allows discarding the redundant variables and representing the knowledge in a compact way [62]. Thus, it allows to help to lower the dimensional space without losing information. Hence, it paves way for resource management while reducing computational power and infrastructure. PCA is commonly used to reduce the dimension by defining a few orthogonal linear combinations (principal components) of original features with a higher variance. Furthermore, PCA helps to identify the most important data features. There are principal components as many as the number of original features. The first principal component takes most of the variance in the data. The second principal component is orthogonal to the first principal component and comprehends the remaining variance.

7.7.Analysis of AI algorithms

7.7.1. Performance Indices

The performance of the approaches used in the analysis was evaluated using performance indices. Since regression algorithms are considered regression metrics are used as performance indices. They are used to decide the best algorithm for haptic sensation prediction. Thus, regression metrics were considered for evaluation of model performance are,

- R^2 score
- Mean squared error (MSE)

- Root mean squared error (RMSE)
- Mean absolute error (MAE)

The model should have a higher R^2 score and lower regression losses indicated by MSE, RMSE, and MAE for a model to perform well.

7.7.1.1. Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and Accuracy

These metrices are the most used performance indices to compare the AI models. The MSE is the square means of all errors which are derived from the predicted and the actual value. The RMSE is the square root of the mean squared error. MSE and RMSE estimated over the n samples can be represented as in (7.15) and (7.16) respectively.

$$\boldsymbol{MSE}\ (\boldsymbol{y_i}, \boldsymbol{\hat{y}_i}) = \frac{\mathbf{1}}{\boldsymbol{n}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (\boldsymbol{y_i} - \boldsymbol{\hat{y}_i})^{\mathbf{2}} \tag{7.15}$$

$$\boldsymbol{RMSE}\ (\boldsymbol{y_i}, \boldsymbol{\hat{y}_i}) = \sqrt{\boldsymbol{MSE}\ (\boldsymbol{y_i}, \boldsymbol{\hat{y}_i})}\ = \sqrt{\frac{\mathbf{1}}{\boldsymbol{n}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (\boldsymbol{y_i} - \boldsymbol{\hat{y}_i})^{\mathbf{2}}} \tag{7.16}$$

where,

n= total no. of samples

y_i= actual value

$\hat{y}_i$= corresponding predicted value of the ith sample

Thus, the accuracy can be defined from MSE and RMSE as,

$$\boldsymbol{Accuracy} = \mathbf{1} - \boldsymbol{RMSE} = \mathbf{1} - \sqrt{\boldsymbol{MSE}} \tag{7.17}$$

7.7.1.2. R^2 score

The estimated R^2 is expressed as,

$$\boldsymbol{R^2}(\boldsymbol{y_i}, \boldsymbol{\hat{y}_i}) = \mathbf{1} - \frac{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (\boldsymbol{y_i} - \boldsymbol{\hat{y}_i})^{\mathbf{2}}}{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (\boldsymbol{y_i} - \boldsymbol{\bar{y}_i})^{\mathbf{2}}} \tag{7.18}$$

where,

n= total no. of samples

y_i= actual value

$\hat{y}_i$= corresponding predicted value of the i[th] sample

$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$

7.7.1.3. Mean Absolute Error (MAE)

MAE is the mean of absolute values of individual error values. It estimated over the n samples can be expressed as (7.19).

$$\boldsymbol{MAE}(\boldsymbol{y_i}, \boldsymbol{\hat{y}_i}) = \frac{\mathbf{1}}{\boldsymbol{n}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} |\boldsymbol{y_i} - \boldsymbol{\hat{y}_i}| \tag{7.19}$$

where,

n= total no. of samples

y_i= actual value

$\hat{y}_i$= corresponding predicted value of the i[th] sample

8. DISCUSSION

Conventional approaches for modelling haptic objects have been used since the beginning of haptic studies and force sensors were often employed to get force response. Furthermore, only motion parameters: compression depth and velocity were often considered while stiffness and viscosity were predefined by considering their behaviour as simple and using mathematical representations and often higher order polynomial representations were considered. Even though recent studies on haptics incorporated AI, these studies primarily utilized force sensors despite their issues. Thus, this study focused on introducing AI approach for the recreation of haptic objects.

This proposed approach followed three phases: Abstraction, Reconstruction and Reproduction. An adequate amount of data was abstracted by squeezing a sponge object by applying force on it with different ramp rates and the obtained dataset was preprocessedx by removing abnormal data points. The whole dataset was divided in to two independent sets for training and testing. Then the training set was utilized in further analysis and building the AI model while testing set was considered in the evaluation process of AI algorithms. The factors affecting for response were found out through a statistical analysis and correlation statistic, mutual information statistics and Principal component analysis were considered to analyze this data set with nine features, however, six features were identified as important features affecting for the feedback from the object. They are $\boldsymbol{x}$, $\dot{\boldsymbol{x}}$, $\ddot{\boldsymbol{x}}$, $\frac{(\boldsymbol{F_{res}})_{t-1}}{\boldsymbol{x}}$, $\frac{(\boldsymbol{F_{res}})_{t-1}}{\dot{\boldsymbol{x}}}$, and $\frac{(\boldsymbol{F_{res}})_{t-1}}{\ddot{\boldsymbol{x}}}$, however, $\boldsymbol{x}$ showed a highest important than any other features. These extracted features were considered as the feature matrix for the input of AI model. According to the PCA it seemed that seven PC are essential to absorb at least 90% knowledge from the dataset as the variance change is smaller from one PC to another. Thus, when considering the PCA results, it seemed that considerable no.of PC are needed to represent the whole dataset. Therefore, PCA doesn't generate important outcomes as it will not impact greatly on reduction of computational power.

Different AI models were trained including linear regression, random forest regression, support vector regression and neural networks and their performance were analyzed by evaluating the performance matrices on testing dataset. Through this analysis, random forest was identified as the AI model, best fit for haptic sensation recreation with the lowest MSE. The AI algorithm was deployed with the hardware using a model intermediate format, PMML and the whole hardware set up was modified to be utilized in the reproduction phase. Though out this study force sensors were not employed, and force measurements were relied on a sensorless force control mechanism based on DOB and RFOB.

In this study, different spring damper model approaches were also analyzed with the AI approach and different AI algorithms were evaluated for exploring the best AI model. Furthermore, the validity of the introduced method is proven by comparing the force response deviation from the actual force response for the same compression depth variation. Ultimately, this study has proven the fact that using AI based approaches along with a sensorless force control mechanism produces haptic sensation with a far higher performance than the conventional approaches that are already been utilized.

9. CONCLUSION

This research proposed an AI based for replicating the actual behavior of the object by taking its nonlinear behavior of responses into account. A typical spring damper model only consider compression depth and velocity as the features affecting for haptic object recreation, however, even with these both features alone AI based approach proved to have a higher performance that conventional approach. Furthermore, it seemed that nonlinear features like hysteresis can be observed, and they cannot be interpreted using simple mathematical relationship and the traditional spring damper model fails in abstracting these features. Thus, a complete statistical analysis was conducted and $\boldsymbol{x}$, $\dot{\boldsymbol{x}}$, $\ddot{\boldsymbol{x}}$, $\frac{(\boldsymbol{F_{res}})_{t-1}}{\boldsymbol{x}}$, $\frac{(\boldsymbol{F_{res}})_{t-1}}{\dot{\boldsymbol{x}}}$, and $\frac{(\boldsymbol{F_{res}})_{t-1}}{\ddot{\boldsymbol{x}}}$, are identified as the most important features. These extracted features were considered as the feature matrix to input the AI model to predict force response. Although a PCA was conducted on the dataset, it doesn't generate important outcomes as almost all the PCs are needed to represent the most knowledge of dataset. Thus, use of PCA instead of raw features will not make a considerable impact to reduce computational power and management of resources. The random forest algorithm was trained by using different feature matrixes to identify the impact of feature matrix. Thus, it seemed that even with the same AI algorithm, a higher performance of the AI model can be obtained when increasing the no. of features affecting the target output. Several AI algorithms were evaluated to find the best algorithm for haptic object reproduction and random forest regression is discovered to be the algorithm with the highest performance. This algorithm was compared with the conventional spring damper model, however, the AI based approach has shown the highest performance with the lowest regression losses, including a RMSE value of 0.34. The model with the higher no.of features in the feature matrix is proven to have a better performance.

Thus, it proved that the AI-based approach outperforms the traditional model-based approaches in replicating the object behavior and as well as in haptic object reproduction. Furthermore, the introduced AI approach is verified by the results obtained under same compression depth variation. Therefore, more promising results can be achieved when reproducing realistic haptic feedback using haptic objects

identified and reproduced through AI-based approaches than conventional model-based approaches.

REFERENCES

[1] Culbertson, H., Schorr, S., & Okamura, A. (2018). Haptics: The Present and Future of Artificial Touch Sensation. Annual Review Of Control, Robotics, And Autonomous Systems, 1(1), 385-409. doi: 10.1146/annurev-control-060117-105043.

[2] Lederman, S., & Klatzky, R. (2009). Haptic perception: A tutorial. Attention, Perception &Amp; Psychophysics, 71(7), 1439-1459. doi: 10.3758/app.71.7.1439.

[3] Lederman, S., & Klatzky, R. (1987). Hand movements: A window into haptic object recognition. Cognitive Psychology, 19(3), 342-368. doi: 10.1016/0010-0285(87)90008-9.

[4] Marti, P., Parlangeli, O., Recupero, A., Guidi, S., & Sirizzotti, M. (2021). Mid-air haptics for shape recognition of virtual objects. Ergonomics, 65(5), 775-793. doi: 10.1080/00140139.2021.1992019.

[5] Shimono, T., Katsura, S., & Ohnishi, K. (2007). Abstraction and Reproduction of Force Sensation From Real Environment by Bilateral Control. IEEE Transactions On Industrial Electronics, 54(2), 907-918. doi: 10.1109/tie.2007.892744.

[6] Abeykoon, A. H. S., & Ohnishi, K. (2006, December). Bilateral control interacting with a virtual model and environment. In 2006 IEEE International Conference on Industrial Technology (pp. 1320-1325). IEEE.

[7] Schmidts, A. M., Lee, D., & Peer, A. (2011, September). Imitation learning of human grasping skills from motion and force data. In 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 1002-1007). IEEE.

[8] Rychlewski, J.: On hooke's law. Journal of Applied Mathematics and Mechanics 48(3), 303–314 (1984)

[9] Ruwanthika, R. M. M., & Abeykoon, A. H. S. (2015, April). 3d environmental force: Position impedance variation for different motion parameters. In 2015 Moratuwa Engineering Research Conference (MERCon) (pp. 112-117). IEEE.

[10] Katsura, S., Matsumoto, Y., & Ohnishi, K. (2007). Modeling of Force Sensing and Validation of Disturbance Observer for Force Control. IEEE Transactions On Industrial Electronics, 54(1), 530-538. doi: 10.1109/tie.2006.885459.

[11] Khalil, I. S., & Sabanovic, A. (2011). Sensorless torque/force control. Advances in Motor Torque Control, 49-69.

[12] Benko, H., Holz, C., Sinclair, M., Ofek, E.: Normaltouch and texturetouch: High-fidelity 3d haptic shape rendering on handheld virtual reality controllers. In: Proceedings of the 29th Annual Symposium on User Interface Software and Technology, pp. 717–728 (2016)

[13] Munoz, J., Gutierrez, G., Sanchis, A.: A human-like torcs controller for thesimulated car racing championship. In: Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games, pp. 473–480 (2010). IEEE.

[14] Lee, J., Sinclair, M., Gonzalez-Franco, M., Ofek, E., Holz, C.: Torc: A virtual reality controller for in-hand high-dexterity finger interaction. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, pp. 1–13 (2019)

[15] Ohnishi, K., Shibata, M., & Murakami, T. (1996). Motion control for advanced mechatronics. IEEE/ASME Transactions On Mechatronics, 1(1), 56-67. doi: 10.1109/3516.491410.

[16] Shimono, T., Katsura, S., & Ohnishi, K. (2005, November). Improvement of operationality for bilateral control based on nominal mass design in disturbance observer. In 31st Annual Conference of IEEE Industrial Electronics Society, 2005. IECON 2005. (pp. 6-pp). IEEE.

[17] Murakami, T., Yu, F., & Ohnishi, K. (1993). Torque sensorless control in multidegree-of-freedom manipulator. IEEE Transactions On Industrial Electronics, 40(2), 259-265. doi: 10.1109/41.222648.

[18] Katsura, S., Yamanouchi, W., & Yokokura, Y. (2012). Real-World Haptics: Reproduction of Human Motion. IEEE Industrial Electronics Magazine, 6(1), 25-31. doi: 10.1109/mie.2012.2182854.

[19] Ohnishi, K., & Mizoguchi, T. (2017). Real haptics and its applications. IEEJ Transactions On Electrical And Electronic Engineering, 12(6), 803-808. doi: 10.1002/tee.22562.

[20] Sun, X., Nozaki, T., Murakami, T., & Ohnishi, K. (2019, March). Grasping point estimation based on stored motion and depth data in motion reproduction system. In 2019 IEEE International Conference on Mechatronics (ICM) (Vol. 1, pp. 471-476). IEEE..

[21] Smith, A., Mobasser, F., & Hashtrudi-Zaad, K. (2006). Neural-Network-Based Contact Force Observers for Haptic Applications. IEEE Transactions On Robotics, 22(6), 1163-1175. doi: 10.1109/tro.2006.882923.

[22] Sun, H., & Martius, G. (2018, November). Robust affordable 3D haptic sensation via learning deformation patterns. In 2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids) (pp. 846-853). IEEE.

[23] T. Bhattacharjee, G. Lee, H. Song, and S. S. Srinivasa, “Towards robotic feeding: Role of haptics in fork-based food manipulation,” IEEE Robotics and Automation Letters, vol. 4, no. 2, pp. 1485–1492, 2019.

[24] T. Matsunaga, K. Ohnishi, N. Wada, and Y. Kitagawa, “Development of small-diameter haptic flexible gripping forceps robot,” Electrical Engineering in Japan, vol. 211, no. 1-4, pp. 47–54, 2020.

[25] E. M. Overtoom, T. Horeman, F.-W. Jansen, J. Dankelman, and H. W. Schreuder, “Haptic feedback, force feedback, and force-sensing in simulation training for Laparoscopy: A systematic overview,” Journal of Surgical Education, vol. 76, no. 1, pp. 242–261, 2019.

[26] D. Panariello, T. Caporaso, S. Grazioso, G. Di Gironimo, A. Lanzotti, S. Knopp, L. Pelliccia, M. Lorenz, and P. Klimant, “Using the KUKA LBR iiwa robot as haptic device for virtual reality training of hip replacement surgery,” 2019 Third IEEE International Conference on Robotic Computing (IRC), 2019.

[27] A. M. Okamura, “Haptic feedback in robot-assisted minimally invasive surgery,” Current Opinion in Urology, vol. 19, no. 1, pp. 102–107, 2009.

[28] A. Turolla, O. A. Daud Albasini, R. Oboe, M. Agostini, P. Tonin, S. Paolucci, G. Sandrini, A. Venneri, and L. Piron, “Haptic-based neurorehabilitation in poststroke patients: A feasibility prospective multicentre trial for Robotics Hand Rehabilitation,” Computational and Mathematical Methods in Medicine, vol. 2013, pp. 1–12, 2013.

[29] P. Y. Chua, T. Ilschner, and D. G. Caldwell, “Robotic manipulation of Food Products – A Review,” Industrial Robot: An International Journal, vol. 30, no. 4, pp. 345–354, 2003.

[30] G. LIU, X. GENG, L. LIU, and Y. WANG, “Haptic based teleoperation with master-slave motion mapping and haptic rendering for space exploration,” Chinese Journal of Aeronautics, vol. 32, no. 3, pp. 723–736, 2019.

[31] Knopp, S., Lorenz, M., Pelliccia, L., & Klimant, P. (2018, March). Using industrial robots as haptic devices for VR-training. In 2018 IEEE conference on virtual reality and 3D user interfaces (VR) (pp. 607-608). IEEE.

[32] González, C., Solanes, J. E., Munoz, A., Gracia, L., Girbés-Juan, V., & Tornero, J. (2021). Advanced teleoperation and control system for industrial robots based on augmented virtuality and haptic feedback. Journal of Manufacturing Systems, 59, 283-298.

[33] Walia, A., Goel, P., Kairon, V., & Jain, M. (2020, April). HapTech: Exploring Haptics in Gaming for the Visually Impaired. In Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems (pp. 1-6).

[34] Chen, Y. S., Han, P. H., Hsiao, J. C., Lee, K. C., Hsieh, C. E., Lu, K. Y., ... & Hung, Y. P. (2016, October). Soes: Attachable augmented haptic on gaming controller for immersive interaction. In Adjunct Proceedings of the 29th Annual ACM Symposium on User Interface Software and Technology (pp. 71-72).

[35] A. Turolla, O. A. Daud Albasini, R. Oboe, M. Agostini, P. Tonin, S. Paolucci, G. Sandrini, A. Venneri, and L. Piron, “Haptic-based neurorehabilitation in poststroke patients: A feasibility prospective multicentre trial for Robotics Hand

Rehabilitation," Computational and Mathematical Methods in Medicine, vol. 2013, pp. 1–12, 2013.

[36] Morris, D., Joshi, N., & Salisbury, K. (2004, March). Haptic battle pong: High-degree-of-freedom haptics in a multiplayer gaming environment. In Experimental gameplay workshop, GDC (Vol. 192).

[37] Sung, G. T., & Gill, I. S. (2001). Robotic laparoscopic surgery: a comparison of the da Vinci and Zeus systems. Urology, 58(6), 893-898.

[38] Bethea, B. T., Okamura, A. M., Kitagawa, M., Fitton, T. P., Cattaneo, S. M., Gott, V. L., ... & Yuh, D. D. (2004). Application of haptic feedback to robotic surgery. Journal of Laparoendoscopic & Advanced Surgical Techniques, 14(3), 191-195.

[39] Freschi, C., Ferrari, V., Melfi, F., Ferrari, M., Mosca, F., & Cuschieri, A. (2013). Technical review of the da Vinci surgical telemanipulator. The International Journal of Medical Robotics and Computer Assisted Surgery, 9(4), 396-406.

[40] Perret, J., Vander Poorten, E.: Touching virtual reality: a review of haptic gloves. In: ACTUATOR 2018; 16th International Conference on New Actuators, pp. 1–5 (2018). VDE

[41] Guazzelli, A., Zeller, M., Lin, W., & Williams, G. (2009). PMML: An Open Standard for Sharing Models. The R Journal, 1(1), 60. doi: 10.32614/rj-2009-010.

[42] Dewapura, P. W., Jayawardhana, K. M., & Abeykoon, A. H. S. (2021, September). Object Identification using Support Vector Regression for Haptic Object Reconstruction. In 2021 3rd International Conference on Electrical Engineering (EECon) (pp. 144-150). IEEE

[43] Brownlee, J. (2018). Statistical methods for machine learning: Discover how to transform data into knowledge with Python. Machine Learning Mastery.

[44] Kuhn, M., & Johnson, K. (2019). Feature engineering and selection: A practical approach for predictive models. CRC Press.

[45] Zheng, A., & Casari, A. (2018). Feature engineering for machine learning: principles and techniques for data scientists. " O'Reilly Media, Inc.".

[46] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research 12.

[47] Gross, J., & Groß, J. (2003). Linear regression (Vol. 175). Springer Science & Business Media.

[48] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

[49] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.

[50] Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. Statistics and computing, 14(3), 199-222.

[51] Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. In Neural networks for perception (pp. 65-93). Academic Press.

[52] Kamada, S., & Ichimura, T. (2018). Fast training of adaptive structural learning method of deep learning for multi modal data. International Journal Of Computational Intelligence Studies, 7(3/4), 169. doi: 10.1504/ijcistudies.2018.10017446.

[53] Bhardwaj, A., Di, W., & Wei, J. (2018). Deep Learning Essentials: Your hands-on guide to the fundamentals of deep learning and neural network modeling. Packt Publishing Ltd.

[54] Waheeb, W., & Ghazali, R. (2016). Chaotic Time Series Forecasting Using Higher Order Neural Networks. International Journal On Advanced Science, Engineering And Information Technology, 6(5), 624. doi: 10.18517/ijaseit.6.5.958.

[55] Bahrampour, S., Ramakrishnan, N., Schott, L., & Shah, M. (2015). Comparative study of deep learning software frameworks. arXiv preprint arXiv:1511.06435.

[56] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Zheng, X. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.

[57] Tantawi, I., Abushariah, M., & Hammo, B. (2021). A deep learning approach for automatic speech recognition of The Holy Qur'ān recitations. International Journal Of Speech Technology, 24(4), 1017-1032. doi: 10.1007/s10772-021-09853-9.

[58] Polyzotis, N., Roy, S., Whang, S., & Zinkevich, M. (2018). Data Lifecycle Challenges in Production Machine Learning. ACM SIGMOD Record, 47(2), 17-28. doi: 10.1145/3299887.3299891.

[59] Kraskov, A., Stögbauer, H., & Grassberger, P. (2004). Estimating mutual information. Physical Review E, 69(6). doi: 10.1103/physreve.69.066138.

[60] Ross, B. (2014). Mutual Information between Discrete and Continuous Data Sets. Plos ONE, 9(2), e87357. doi: 10.1371/journal.pone.0087357.

[61] Kozachenko, L. F., & Leonenko, N. N. (1987). Sample estimate of the entropy of a random vector. Problemy Peredachi Informatsii, 23(2), 9-16.

[62] Goodfellow, I., Bengio, Y., & Courville, A. (2017). Deep learning (adaptive computation and machine learning series). Cambridge Massachusetts, 321-359.

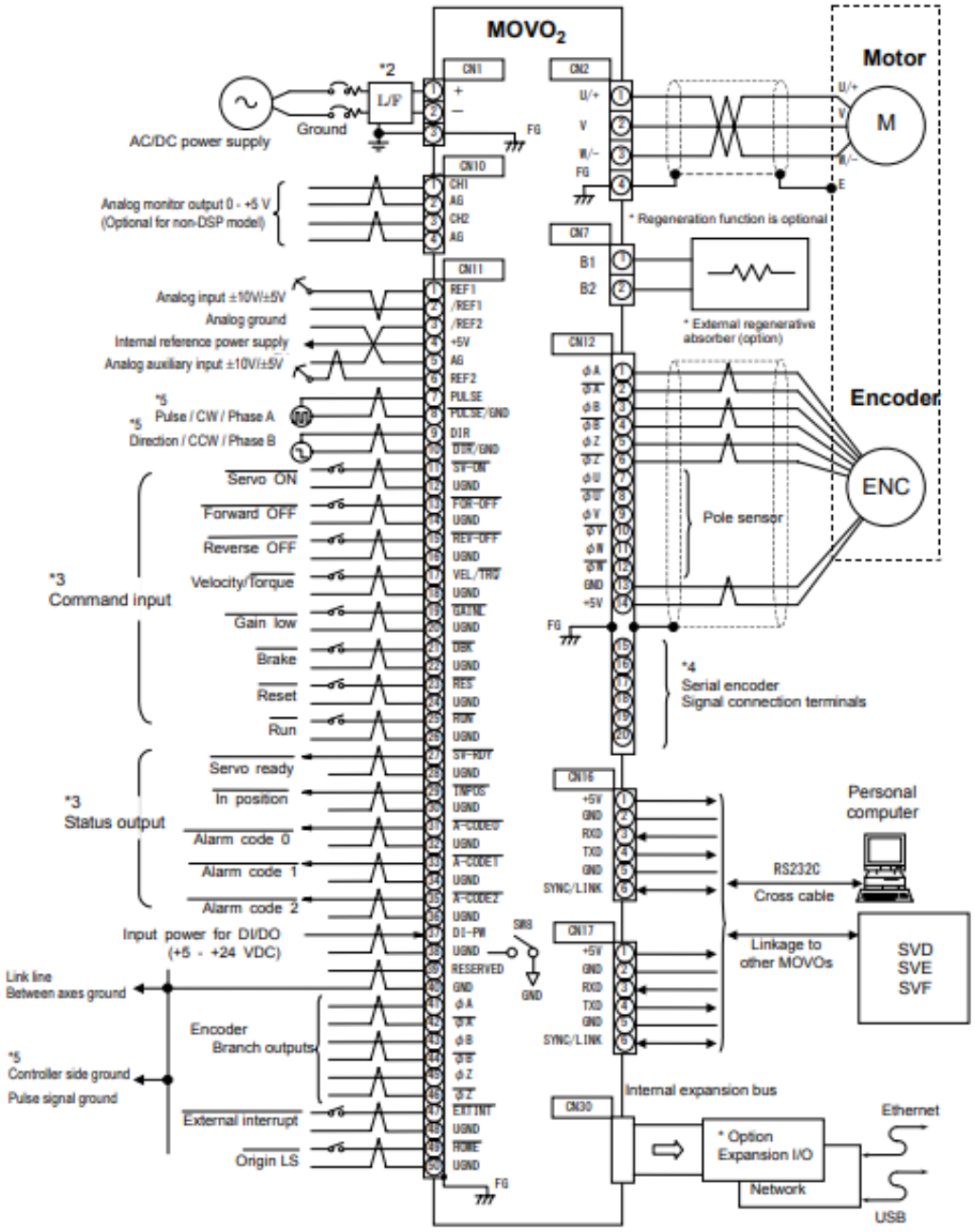
[63] "What is tinyml? - technical articles," All About Circuits. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/what-is-tinyml/. [Accessed: 12-Mar-2023].

APPENDICES

APPENDIX A Hardware Block diagram of the Motor Driver, MOVO2

SVF series (models other than SVFH10, 20 and 40, release ver. D and F drivers)

SFVM, SVFH (1-phase), SVFL (124, 224, 448) *1



*1 SVFL (battery type) is in planning stage and not available yet.

*2 Install a noise filter

*3 For input/output circuit specifications, refer to "4.1 Interface circuit"

*4 Connection point differs depending on the type of serial encoder. For details refer to the wiring diagrams for serial encoder manufacturers.

*5 For using differential input of pulse command signal or for using encoder branch output signal, join the control-side grounding line of pulse command signal with 40 pin (GND) of CN11.

APPENDIX B Parameter List Values of the Motor Driver, MOVO2

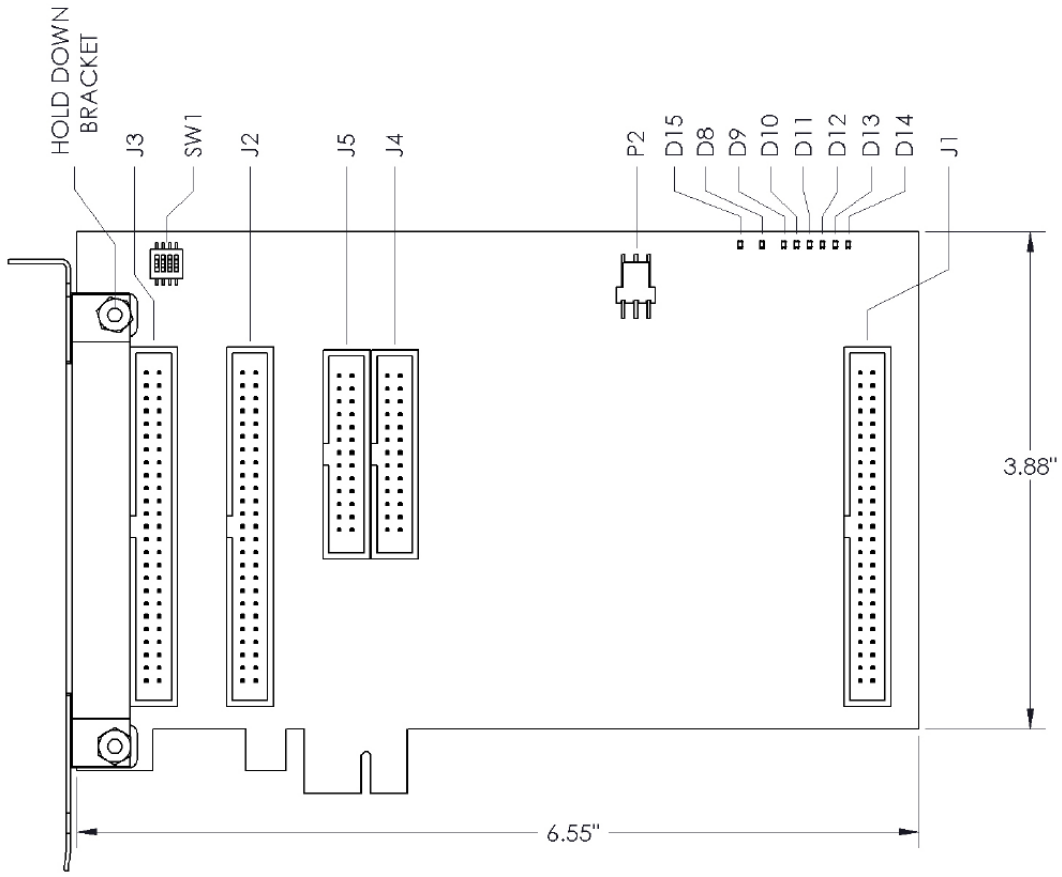
Parameter No.	Parameter	Value
#000	Encoder basic resolution	6000
#001	Linear Motor reference length	120
#002	Number of poles	4
#003	Motor rated current	6
#004	Current limit value	100
#007	Maximum output speed	4000
#008	Detection mode	40
#009	Command mode	1110
#010	Pole sensor position	0
#011	I/O type	101
#012	Initial mode	37
#013	Electronic gear denominator	100
#014	Electronic gear numerator	100
#015	Electronic volume	20800
#016	Electronic trimmer	60
#017	Position loop time constant	14
#018	Velocity loop time constant	5
#019	Acceleration loop gain	20
#020	Low gain	100
#021	MOV/2 mode	12
#022	Set velocity	100
#023	Jog velocity	100
#024	Creep velocity	10
#025	Acceleration / Deceleration time	60
#026	Jerk time	20
#027	Overrun limit	1000
#028	In position width	1

#029	Origin position	0
#030	Origin mode	0
#031	Thermal time constant	16
#032	Switch	110
#033	Motor axis name	88
#034	Maintenance code	0
#035	Friction	0
#036	Thermal value	149
#037	Optional encoder type	0
#038	Motor type	102
#039	Acceleration filter	8
#040	Driver type	120012
#041	Driver maximum current	21
#042	Phase U offset	-2395
#043	Phase U gain	7360
#044	Phase V offset	-2401
#045	Phase V gain	7311
#046	Gain of power factor detection	64
#047	Parameter check sum	9096
#048	Electronic volume 2	15000
#049	Electronic trimmer 2	0
#050	CH1 offset (Analog Monitor)	-1902
#051	CH1 gain (Analog Monitor)	7049
#052	CH2 offset (Analog Monitor)	-1871
#053	CH2 gain (Analog Monitor)	7018
#054	Deceleration time	0
#055	Deceleration jerk time	0
#064	Proportional gain (current loop)	1063
#065	Integral gain (current loop)	1024
#066	Differential gain (current loop)	0
#067	PWM frequency	15

#068	Encoder jitter filter	0
#069	Excessive error setting	10
#070	Axes synchronization option	0
#072	Adaptive control time constant	0
#074	Standard DI port logic inversion	0
#075	Standard DO port logic inversion	0
#076	Velocity magnification	1
#077	Alarm record trigger select	0
#078	Servo cycle	16
#079	Filter mode	0
#080	Proportional gain (Velocity PID)	500
#081	Integral gain (Velocity PID)	10
#082	Differential gain (Velocity PID)	1
#083	Velocity filter	2
#084	Pulse filter	3
#085	DI port filter	0
#086	Tachometer on rate	3
#087	Encoder branch division ratio	0
#089	Electronic gear 2	0
#090	Bus voltage curve	357
#091	Bus voltage gain	539
#092	Driver power class	12616
#093	Switch 2	0
#096	Current mode	36
#104	ITC Driver resistance	0
#105	ITC/ITU time constant	0
#106	ITC DC bus voltage	0
#107	ITC motor resistance	0
#108	ITC motor inductance	0
#109	ITC motor EMF constant	0
#110	Notch switch	0

#111	Notch mode	0
#112	Notch frequency (First channel)	0
#113	Notch band width (First channel)	0
#114	Notch frequency (Second channel)	0
#115	Notch band width (Second channel)	0
#116	Notch frequency (Third channel)	0
#117	Notch band width (Third channel)	0
#120	Regeneration type	0
#121	Regeneration rated power	0
#122	Regeneration resistance	0
#123	Regeneration thermal time constant	0
#124	Correction start select	0
#130	BSC communication speed	0
#131	ANI master	0
#132	Analog input dead zone	0
#133	Reserved	0
#134	BSC communication mode	0
#137	Torque monitor select	0
#138	Torque monitor gain	0
#139	Torque monitor offset	0

APPENDIX C Board Layout of Sensoray's Model 826



APPENDIX D C & C++ codes

- Abstraction

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// SENSORAY MODEL 826 PROGRAMMING EXAMPLES
// This file contains simple functions that show how to program the 826.
// Copyright (C) 2012 Sensoray
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// MSc setup - Abstraction
// Praveena Dewapura
// 2022.03.30
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Header functions declarations
#ifndef _LINUX
#include <windows.h>
#include <conio.h>
#endif
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#ifndef _LINUX
#include "..\826api.h"
#else
#include "826api.h"
#endif

// Helpful macros for DIOs
#define DIO(C)              ((uint64)1 << (C))                          // convert dio channel number to uint64 bit mask
#define DIOMASK(N)          {(uint)(N) & 0xFFFFFF, (uint)((N) >> 24)}   // convert uint64 bit mask to uint[2] array
#define DIOSTATE(STATES,CHAN)   ((STATES[CHAN / 24] >> (CHAN % 24)) & 1)    // extract dio channel's
boolean state from uint[2] array

////////////////////////////////////////////////////////////////////////////////
// ERROR HANDLING
// These examples employ very simple error handling: if an error is detected, the example functions will immediately return an
error code. This behavior may not be suitable for some
// real-world applications but it makes the code easier to read and understand. In a real application, it's likely that additional
actions would need to be performed. The examples use
// the following X826 macro to handle API function errors; it calls an API function and stores the returned value in errcode,
then returns immediately if an error was detected.
#define X826(FUNC)   if ((errcode = FUNC) != S826_ERR_OK) { printf("\nERROR: %d\n", errcode); return errcode;}

// end of hearder fuction declarations

////////////////////////////////////////////////////////////////////////////////
// Global variables declaration

// Encoder read
uint counts = 0;                    // encoder counts when the snapshot was captured
uint counts_pre =0;
uint timestamp;                     // time the snapshot was captured in us. converted to s by /10^6
uint timestamp_start;
uint timestamp_previous;            // time the previous snapshot was captured
uint aout = 0;                      // aout= analogue output channel no.0 , output duplicate waveform on this dac channel
uint chan = 4;                      // +++ Check the connection 41 - current- analog o/p chan 4
uint setpoint;
uint range;
int errcode = S826_ERR_OK;
int steps = 0;

//for loop oparation
long long int i ;
long long int j ;
char file_name[1000];
int k = 0;

float function_exicution_time = 0.0;
```

```
float dt = 0.0;
float F_ref = 0.0;
float F_ref_max = 0.0;
float F_ref_min = 0.0;
float ramp_rate = 0.0;

// motor drive oparation
unsigned int analog_out_motor;
float I_a_ref = 0.0;
float I_motor = 0.0;

// motor parameters
float K_fn = 24.0;                      // motor force constent
float M_n = 0.643;                      // Mass : (Motor shaft, 2 x linera bearings, 1 x Long green bearing and shft connector 1 x
short green                             //bearing and shft connector)=600g and (whight hammer head)=55g .   Mass : (Motor
shaft, 2 x linera bearings, 1 x Long green bearing and shft connector 1 x short green bearing and shft connector)=600g and
(whight hammer head)=55g . Changed the hammer head mass to 43 from 57 from elctronic balance, thus the new mass =  0.655
- 55 + 43 =0.643g
float friction = 0.0;                   // +++ Eventhough friction is assumed as 0 there is a 1N, the motor shaft doesn't move tll
there is 1N value shown in the                  //spring balance

// position, velocity and acceleration mesurment
float velocity_sum = 0.0;
float velocity = 0.0;                   // motor velocity
float g_velocity = 30.0;                // velocity filter constant
float position_x = 0.0;                 // motor physical position
float position_x_prevous = 0.0;
float position_dx = 0.0;
float acc_sum =0.0;
float acceleration = 0.0;
float g_acc =100.0;

// DOb
float dob_input = 0.0;
float dob_filter_input = 0.0;
float dob_filter_output = 0.0;
int const g_dis = 300.0;                //Consider g_rec = g_dis
float dob_force = 0.0;

// RTOB
float rtob_filter_input = 0.0;
float rtob_filter_output = 0.0;
float rtob_force = 0.0;

// PID
float K_p = 2.0;

//Generate force response
// float K_s = 2000.0

// end of veriable declarations

////////////////////////////////////////////////////////////////////////////////////////
// Function declaration

static float VelocityCalculation();
static int ControlLoop(uint board);
static int DemoWatchdog(uint board);
static int MotorOutDAC(uint board);
static float DoB();
static float RToB();
static float dtCalculation();
static float AccelerationCalculation();
static float MainCalculation(uint board);

// end of function declaration

////////////////////////////////////////////////////////////////////////////////////////
// Main function.
int main(int argc, char **argv)
{
```

```
   uint board      = 0;                          // 1 board, named as 0, change this if want to use other than board number 0
   int errcode     = S826_ERR_OK;
   int boardflags  = S826_SystemOpen();                  // open 826 driver and find all 826 boards

   if (argc > 1)
      board = atoi(argv[1]);

   if (boardflags < 0)
         errcode = boardflags;                                  // problem during open
      else if ((boardflags & (1 << board)) == 0)
   {
      int i;
         printf("TARGET BOARD of index %d NOT FOUND\n",board);            // driver didn't find board you want to use
         for (i = 0; i < 8; i++)
      {
         if (boardflags & (1 << i))
         {
            printf("board %d detected. try \"./s826demo %d\"\n", i, i);
               }
         }
   } else
   {
      // running functions
      X826( ControlLoop(board)          );                  // read the endocer value, output anlog value and file write
      X826( DemoWatchdog(board)         );                  // watchdog timer
   }

   switch (errcode)
   {
      case S826_ERR_OK:          break;
      case S826_ERR_BOARD:       printf("Illegal board number"); break;
      case S826_ERR_VALUE:       printf("Illegal argument"); break;
      case S826_ERR_NOTREADY:    printf("Device not ready or timeout"); break;
      case S826_ERR_CANCELLED:   printf("Wait cancelled"); break;
      case S826_ERR_DRIVER:      printf("Driver call failed"); break;
      case S826_ERR_MISSEDTRIG:  printf("Missed adc trigger"); break;
      case S826_ERR_DUPADDR:     printf("Two boards have same number");break;S826_SafeWrenWrite(board, 0x02);
      case S826_ERR_BOARDCLOSED: printf("Board not open"); break;
      case S826_ERR_CREATEMUTEX: printf("Can't create mutex"); break;
      case S826_ERR_MEMORYMAP:   printf("Can't map board"); break;
      default:                   printf("Unknown error"); break;
   }

#ifndef _LINUX
   printf("\nKeypress to exit ...\n\n");
   while (!_kbhit());
   _getch();
#endif

   S826_SystemClose();
   return 0;
}
// end of main function

////////////////////////////////////////////////////////////////////////////////////////
// ENCODER READ ANLOG OUT AND FILE WRITE FUCNTION
// JKD - 10.08.2020

static int ControlLoop(uint board)
{
   // ***Configure interfaces and start them running.
   X826( S826_DacRangeWrite(board, aout, S826_DAC_SPAN_0_5, 0));       // program dac output range: -5V to +5V , motor
0 value is 0V / for motor drive operation , IF SO SHOULD CHANGE FROM THE MOTOR DRIVER'S SIDE
   //X826( S826_DacRangeWrite(board, chan, S826_DAC_SPAN_0_5, 0));     // program dac output range: -0V to +5V , motor
0 value is 2.5V / for motor drive operation ,
   X826( S826_CounterModeWrite(board, 0, S826_CM_K_QUADX4));          // Configure counter0 as incremental encoder
interface. quadrature
   X826( S826_CounterStateWrite(board, 0, 1));                  // Start tracking encoder position.

   I_motor = 0.0;
   position_x = 0.0;
```

```
   // Injecting nevagive current, stopping cycle
   F_ref = 0.0;
   I_motor = 0.0;
   I_a_ref = 0.0;                                          //stop the mortor input current

   MotorOutDAC(board);
   // X826( S826_CounterStateWrite(board, 0, 0)    );              // Stop tracking encoder position.
   dtCalculation();
   VelocityCalculation();
   AccelerationCalculation();
   X826( S826_CounterSnapshot(board, 0) );                                  // Trigger snapshot on counter 0.
   X826( S826_CounterSnapshotRead(board, 0, &counts, &timestamp_start, NULL, 0));      // Read the snapshot: from the
counter0

   // force loop
   F_ref_min = 0.0;
   F_ref = 0.0;
   I_a_ref = 0.0;
   //K_s=0.002;

   timestamp_previous = timestamp_start;
   X826( S826_CounterSnapshot(board, 0)                    );              // Trigger snapshot on counter 0.
   X826( S826_CounterSnapshotRead(board, 0, &counts, &timestamp, NULL, 0)  );      // Read the snapshot:receive the
snapshot info here;no need to wait for snapshot;it's already been captured
   // Important to calculate dt, velocity before dob, rtob calculations
   dtCalculation();
   VelocityCalculation();
   AccelerationCalculation();


////////////////////////////////////////////////////////////////////////////
   ramp_rate =6.2;     // 2.1 , 2.4 , 2.7 , 3.0                        // to change the force ramping
   F_ref_max = 14.0;       // 6 , 9 , 12 , 15

   // ****file write operation
   k=1;
   if(abs(position_x)<0.05){
      for(j=0;j<14;j++){
         FILE *fp;
         snprintf(file_name, sizeof(char) * 32, "test_04/testV04_6.2_%i.csv", k);
         fp = fopen(file_name, "a+");                                  // file pointer file file operation
         // fp= fopen("testdata_V02_.txt", "a+");

fprintf(fp,"timestamp_start,timestamp,timestamp_previous,function_exicution_time,dt,F_ref,rtob_force,dob_force,counts,steps,
position_x,velocity,I_motor,I_a_ref,analog_out_motor,acceleration\n");

         // force min =0
         for(i=0;i<200000;i++)
         {
            F_ref = 0.0;
            MainCalculation(board);
            fprintf(fp,"%d,%d,%d,%f,%f,%f,%f,%f,%d,%d,%f,%f,%f,%f,%u,%f\n",
timestamp_start,timestamp,timestamp_previous,function_exicution_time,dt,F_ref,rtob_force,dob_force,counts,steps,position_x,
velocity,I_motor,I_a_ref,analog_out_motor,acceleration);

         }

         // force increment
         for(i=0;i<2000000;i++)
         {
            if (F_ref < F_ref_max)
            {
               //F_ref = -K_s*position_x;
               F_ref = F_ref + (dt * ramp_rate);
               printf("position :%f",position_x);
               printf("    F_ref :%f",F_ref);

            }else
            {
               F_ref = F_ref;
```

```
            }
            MainCalculation(board);
            fprintf(fp,"%d,%d,%d,%f,%f,%f,%f,%f,%d,%d,%f,%f,%f,%f,%u,%f\n",

timestamp_start,timestamp,timestamp_previous,function_exicution_time,dt,F_ref,rtob_force,dob_force,counts,steps,position_x,
velocity,I_motor,I_a_ref,analog_out_motor,acceleration);

         }

         // force max
         for(i=0;i<200000;i++)
         {
            F_ref = F_ref;
            MainCalculation(board);
            fprintf(fp,"%d,%d,%d,%f,%f,%f,%f,%f,%d,%d,%f,%f,%f,%f,%u,%f\n",

timestamp_start,timestamp,timestamp_previous,function_exicution_time,dt,F_ref,rtob_force,dob_force,counts,steps,position_x,
velocity,I_motor,I_a_ref,analog_out_motor,acceleration);
         }

         // force reduction
         for(i=0;i<2000000;i++)
         {
            if (F_ref > 0.0)
            {
               F_ref = F_ref - (dt * ramp_rate);
            }else
            {
               F_ref = 0.0;
            }
            MainCalculation(board);
            fprintf(fp,"%d,%d,%d,%f,%f,%f,%f,%f,%d,%d,%f,%f,%f,%f,%u,%f\n",

timestamp_start,timestamp,timestamp_previous,function_exicution_time,dt,F_ref,rtob_force,dob_force,counts,steps,position_x,
velocity,I_motor,I_a_ref,analog_out_motor,acceleration);
         }

         // force min =0
         for(i=0;i<200000;i++)
         {
            F_ref = 0.0;
            MainCalculation(board);
            fprintf(fp,"%d,%d,%d,%f,%f,%f,%f,%f,%d,%d,%f,%f,%f,%f,%u,%f\n",

timestamp_start,timestamp,timestamp_previous,function_exicution_time,dt,F_ref,rtob_force,dob_force,counts,steps,position_x,
velocity,I_motor,I_a_ref,analog_out_motor,acceleration);
         }

         F_ref_max = F_ref_max+1.0 ;
         k=k+1;
         fclose(fp);
      }
   }else{

      // Injecting nevagive current, stopping cycle
      F_ref = 0.0;
      I_motor = 0.0;
      I_a_ref = 0.0;                                    //stop the mortor input current
      MotorOutDAC(board);
      X826( S826_CounterStateWrite(board, 0, 0)   );            // Stop tracking encoder position.
      return errcode;

   }
////////////////////////////////////////////////////////////

   // Injecting nevagive current, stopping cycle
   F_ref = 0.0;
   I_motor = 0.0;
   I_a_ref = 0.0;                              //stop the mortor input current
   MotorOutDAC(board);
   X826( S826_CounterStateWrite(board, 0, 0)   );            // Stop tracking encoder position.
```

```
    return errcode;
}
// end of main function
////////////////////////////////////////////////////////////////////////////////////////
// dt calculation
static float dtCalculation()
{
    function_exicution_time = (float)(timestamp - timestamp_start)/1000000;     //Timestamp in us , /10^6 to convert to s
    dt =(float)(timestamp - timestamp_previous)/1000000;
    timestamp_previous = timestamp;
    return 0;
}
// velocity calculation
static float VelocityCalculation()
{
    steps = 1 * counts;                                          // Correcting the encorder read , + to the direction of squeezing
    position_x = steps*0.000005;
    velocity_sum = velocity_sum + velocity*dt;
    velocity = g_velocity * (position_x - velocity_sum);
    counts_pre = counts;
    return 0;
}
//Acceleration Calculation
static float AccelerationCalculation()
{
    // steps = 1 * counts;                          // Correcting the encorder read , + to the direction of squeezing
    // position_x = steps*0.000005;
    acc_sum = acc_sum + acceleration*dt;
    acceleration = g_acc * (velocity - acc_sum);
    // counts_pre = counts;
    return 0;
}
// DOB
static float DoB()
{
    dob_input = velocity * g_dis * M_n;
    dob_filter_input = K_fn * I_motor + dob_input;
    dob_filter_output = dob_filter_output + g_dis*(dob_filter_input - dob_filter_output)*dt;
    dob_force = dob_filter_output - dob_input;
    return 0;
}
// RTOB
static float RToB()
{
    rtob_filter_input = dob_filter_input - friction;
    rtob_filter_output = rtob_filter_output + g_dis*(rtob_filter_input - rtob_filter_output)*dt;
    rtob_force = rtob_filter_output - dob_input;
    return 0;
}
// Motor drive input voltage genaration function, using dac
static int MotorOutDAC(uint board)
{
    if (I_motor < -0.59)
    {
        I_motor = -0.59;
    }else if(I_motor > 0.59)
    {
        I_motor = 0.59;
    }
    analog_out_motor = (unsigned int)((65535*0.5) + ((I_motor*65535)/(0.6*2))); // DAC senstivity 16 bit (2**16)movo zero
vlaue 0 -->  2.5V out in dac
    //analog_out_motor = (unsigned int)(32768 + (I_motor/0.6)*32768);          // DAC senstivity 16 bit (2**16)movo zero vlaue
0 --> 0V out in dac (-5 - 5V)
    X826( S826_DacDataWrite(board, aout, analog_out_motor, 0) );               // Analog value out from dac
    //X826( S826_DacDataWrite(board, chan, analog_out_motor, 0) );              // Analog value out from dac
    return 0;                                                  // negative current - negative cont, Positive current positive counter
incriment
}

static float MainCalculation(uint board)
```

```
{
   I_a_ref = K_p*(F_ref - rtob_force)*(M_n/K_fn);          // Calculate from the previous response/rtob_tyorque and dob_torque
to get the I_motor adjusted with the error
   I_motor = I_a_ref + (dob_force/K_fn);                   // the mortor input current
   //printf("  counts :%d",counts);

   MotorOutDAC(board);                                     // DAC o/p motor current --> motor v
   X826( S826_CounterSnapshot(board, 0) );                 // Trigger snapshot on counter 0.
   X826( S826_CounterSnapshotRead(board, 0, &counts, &timestamp, NULL, 0) );  // Read the snapshot:receive the snapshot
info here; no need to wait for snapshot; it's already been captured
   //values for current state
   dtCalculation();                                        // calculate dt, velocity, rotob_force, dob_force according to the existing force
command
   VelocityCalculation();
   AccelerationCalculation();
   DoB();
   RToB();
   return 0;
}

////////////////////////////////////////////////////////////////////////////////////////////
// The demo.
static int DemoWatchdog(uint board)
{
   int rc;
   uint timing[5];
   // set timer 1 time-out to around 1 second
   timing[0] = 55000000;
   printf("\nDemoWatchdog\n");
   // enable writing to watchdog
   rc = S826_SafeWrenWrite(board, 0x02);
   if (rc != 0)
   {
      printf("failed to enable wren for watchdog\n");
      return rc;
   }
   rc = S826_WatchdogConfigWrite(board, 0x00, timing);

   if (rc != 0)
   {
      printf("failed to configure WD\n");
      return rc;
   }

   rc = S826_WatchdogEnableWrite(board, 1);
   if (rc != 0)
   {
      printf("failed to enable WD\n");
      return rc;
   }

   // commented out.  for testing watchdog cancel
   // CreateThread(NULL, 4096, testThread, board, 0, NULL);
   // watch indefinitely for watchdog
   rc = S826_WatchdogEventWait(board, S826_WAIT_INFINITE);

   switch (rc)
   {
      case 0:
         printf("watchdog successfully expired\n");
         break;
      case S826_ERR_NOTREADY:
         printf("WD wait timed out\n");
         break;
      case S826_ERR_CANCELLED:
         printf("WD wait cancelled\n");
         break;
      default:
      printf("error %d\n", rc);

   }
```

```
   // disable watchdog
   rc = S826_WatchdogEnableWrite(board, 0);
   if (rc != 0)
   {
      printf("failed to disable WD\n");
      return rc;
   }
   return 0;
}
```

- Reconstruction

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
// SENSORAY MODEL 826 PROGRAMMING EXAMPLES
// This file contains simple functions that show how to program the 826.
// Copyright (C) 2012 Sensoray
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
// MSc setup - Force Reproduction
// Praveena Dewapura
// 2022.05.30
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
// Header functions declarations
#ifndef _LINUX
#include <windows.h>
#include <conio.h>
#endif
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#ifndef _LINUX
#include "..\826api.h"
#else
#include "826api.h"
#endif

// Helpful macros for DIOs
#define DIO(C)              ((uint64)1 << (C))                          // convert dio
channel number to uint64 bit mask
#define DIOMASK(N)          {(uint)(N) & 0xFFFFFF, (uint)((N) >> 24)}
// convert uint64 bit mask to uint[2] array
#define DIOSTATE(STATES,CHAN)   ((STATES[CHAN / 24] >> (CHAN % 24))
& 1)            // extract dio channel's boolean state from uint[2] array

////////////////////////////////////////////////////////////////////////////////////////
```

```
// ERROR HANDLING
// These examples employ very simple error handling: if an error is detected, the example functions will immediately return an error code. This behavior may not be suitable for some
// real-world applications but it makes the code easier to read and understand. In a real application, it's likely that additional actions would need to be performed. The examples use
// the following X826 macro to handle API function errors; it calls an API function and stores the returned value in errcode, then returns immediately if an error was detected.
#define X826(FUNC)   if ((errcode = FUNC) != S826_ERR_OK) {
printf("\nERROR: %d\n", errcode); return errcode;}
// end of hearder fuction declarations

////////////////////////////////////////////////////////////////////////////////////////
// Global variables declaration

// Encoder read
uint counts = 0;                    // encoder counts when the snapshot was captured
uint counts_pre =0;
uint timestamp;                     // time the snapshot was captured in us. converted to s by /10^6
uint timestamp_start;
uint timestamp_previous;                // time the previous snapshot was captured
uint aout = 0;                      // aout= analogue output channel no.0 , output duplicate waveform on this dac channel
uint chan = 4;                      // +++ Check the connection 41 - current- analog o/p chan 4
uint setpoint;
uint range;
int errcode = S826_ERR_OK;
int steps = 0;

float function_exicution_time = 0.0;
float dt = 0.0;
float F_ref = 0.0;
float F_pred = 0.0;
float F_ref_max = 0.0;
float F_ref_min = 0.0;
float ramp_rate = 0.0;

// motor drive oparation
unsigned int analog_out_motor;
float I_a_ref = 0.0;
float I_motor = 0.0;

// motor parameters
```

```
float K_fn = 24.0;                    // motor force constent
float M_n = 0.643;                    // Mass : (Motor shaft, 2 x linera bearings, 1 x
Long green bearing and shft connector 1 x short green
//bearing and shft connector)=600g and (whight hammer head)=55g .   Mass : (Motor
shaft, 2 x linera bearings, 1 x Long green bearing and shft connector 1 x short green
bearing and shft connector)=600g and (whight hammer head)=55g . Changed the
hammer head mass to 43 from 57 from elctronic balance, thus the new mass =  0.655
- 55 + 43 =0.643g
float friction = 0.0;                 // +++ Eventhough friction is assumed as 0 there is
a 1N, the motor shaft doesn't move tll there is 1N value shown in the
//spring balance

// Velocity mesurment
float velocity_sum = 0.0;
float velocity = 0.0;                 // motor velocity
float g_velocity = 30.0;              // velocity filter constent
float position_x = 0.0;               // motor physical position
float position_x_prevous = 0.0;
float position_dx = 0.0;
float position_nom =0.0;
float force_predmodel=0.0;

// Acceleration mesurement
float acc_sum = 0.0;
float acceleration= 0.0;              // motor velocity
float g_acc = 100.0;

// DOb
float dob_input = 0.0;
float dob_filter_input = 0.0;
float dob_filter_output = 0.0;
int const g_dis = 300.0;              //Consider g_rec = g_dis
float dob_force = 0.0;

// RTOB
float rtob_filter_input = 0.0;
float rtob_filter_output = 0.0;
float rtob_force = 0.0;
float F_res =0.0;
float F_x =0.0;
float F_v=0.0;
float F_a=0.0;

// PID
float K_p = 2.0;

//Generate force response
```

```
// float K_s = 2.0;

// end of veriable declarations

////////////////////////////////////////////////////////////////////////////////////////////
// Function declaration

static int dtCalculation();
static float VelocityCalculation();
static float AccelerationCalculation();
static int ControlLoop(uint board);
static int DemoWatchdog(uint board);
static int MotorOutDAC(uint board);
static int DoB();
static int RToB();
static float MainCalculation(uint board);
// end of function declaration

////////////////////////////////////////////////////////////////////////////////////////////
// Main function.
int main(int argc, char **argv)
{
   uint board      = 0;                          // 1 board, named as 0, change this if want
to use other than board number 0
   int errcode     = S826_ERR_OK;
   int boardflags  = S826_SystemOpen();                  // open 826 driver and find all
826 boards

   if (argc > 1)
      board = atoi(argv[1]);

   if (boardflags < 0)
         errcode = boardflags;                                  // problem during open
      else if ((boardflags & (1 << board)) == 0)
   {
      int i;
         printf("TARGET BOARD of index %d NOT FOUND\n",board);              //
driver didn't find board you want to use
         for (i = 0; i < 8; i++)
      {
         if (boardflags & (1 << i))
         {
            printf("board %d detected. try \"./s826demo %d\"\n", i, i);
               }
         }
   } else
   {
```

```
      // running functions
      X826( ControlLoop(board)          );                  // read the endocer value,
output anlog value and file write
      X826( DemoWatchdog(board)         );                  // watchdog timer
   }

   switch (errcode)
   {
      case S826_ERR_OK:          break;
      case S826_ERR_BOARD:       printf("Illegal board number"); break;
      case S826_ERR_VALUE:       printf("Illegal argument"); break;
      case S826_ERR_NOTREADY:    printf("Device not ready or timeout"); break;
      case S826_ERR_CANCELLED:   printf("Wait cancelled"); break;
      case S826_ERR_DRIVER:      printf("Driver call failed"); break;
      case S826_ERR_MISSEDTRIG:  printf("Missed adc trigger"); break;
      case S826_ERR_DUPADDR:     printf("Two boards have same
number");break;S826_SafeWrenWrite(board, 0x02);
      case S826_ERR_BOARDCLOSED: printf("Board not open"); break;
      case S826_ERR_CREATEMUTEX: printf("Can't create mutex"); break;
      case S826_ERR_MEMORYMAP:   printf("Can't map board"); break;
      default:                   printf("Unknown error"); break;
   }

#ifndef _LINUX
   printf("\nKeypress to exit ...\n\n");
   while (!_kbhit());
   _getch();
#endif

   S826_SystemClose();
   return 0;
}
// end of main function

////////////////////////////////////////////////////////////////////////////////////////////
// ENCODER READ ANLOG OUT AND FILE WRITE FUCNTION
// JKD - 10.08.2020

static int ControlLoop(uint board)
{
   // ***Configure interfaces and start them running.
   X826( S826_DacRangeWrite(board, aout, S826_DAC_SPAN_0_5, 0));      //
program dac output range: -5V to +5V , motor 0 value is 0V / for motor drive
operation , IF SO SHOULD CHANGE FROM THE MOTOR DRIVER'S SIDE
   //X826( S826_DacRangeWrite(board, chan, S826_DAC_SPAN_0_5, 0));    //
program dac output range: -0V to +5V , motor 0 value is 2.5V / for motor drive
operation ,
```

```
   X826( S826_CounterModeWrite(board, 0, S826_CM_K_QUADX4));          //
Configure counter0 as incremental encoder interface. quadrature
   X826( S826_CounterStateWrite(board, 0, 1));                    // Start tracking
encoder position.

   I_motor = 0.0;                                          //stop the mortor input current
   position_x = 0.0;
   MotorOutDAC(board);  //- remove
   X826( S826_CounterSnapshot(board, 0) );                                  // Trigger
snapshot on counter 0.
   X826( S826_CounterSnapshotRead(board, 0, &counts, &timestamp_start, NULL,
0));      // Read the snapshot: from the counter0

   // Force loop
   F_ref_min = 0.0;
   ramp_rate = 2.1;        // 2.1 , 2.4 , 2.7 , 3.0              // to change the force ramping
   F_ref = 0.0;
   // K_s=2000.0;
   timestamp_previous = timestamp_start;
   I_a_ref = 0.0;                                        //stop the mortor input current
   MotorOutDAC(board);
   // X826( S826_CounterStateWrite(board, 0, 0)    );            // Stop tracking
encoder position.

   dtCalculation();
   VelocityCalculation();
   AccelerationCalculation();
   X826( S826_CounterSnapshot(board, 0)                    );              // Trigger
snapshot on counter 0.
   X826( S826_CounterSnapshotRead(board, 0, &counts, &timestamp, NULL, 0)  );
// Read the snapshot:receive the snapshot info here;no need to wait for snapshot;it's
already been captured
   // Important to calculate dt, velocity before dob, rtob calculations
   dtCalculation();
   VelocityCalculation();
   AccelerationCalculation();


//////////////////////////////////////////////////////////////////////////////
   cpmml::Model model("Force_Response_adj_pmml.pmml");
   F_ref_max = 14.0 ;      // 6 , 9 , 12 , 15
   // ****file write operation
   FILE *fp;                                 // file pointer file file operation
   fp = fopen("test_pmml/testdata_ml_pmml_new_6_2022.csv", "a+");
   // fp = fopen("test_pmml/testdata_ml_new_30_pva_V_0.3.csv", "a+");

fprintf(fp,"timestamp_start,timestamp,timestamp_previous,function_exicution_time,
```

```
dt,F_ref,rtob_force,dob_force,counts,steps,position_x,velocity,I_motor,I_a_ref,analo
g_out_motor,F_pred,acceleration,F_x,F_v,F_a\n");

   if(abs(position_x)<0.05){

      for(i=0;i<5;i++)
      {
         F_pred=0.0;
         F_ref = F_pred;
         MainCalculation(board);
         fprintf(fp,"%d,%d,%d,%f,%f,%f,%f,%f,%d,%d,%f,%f,%f,%f,%u,
%f,%f,%f,%f,%f\n",

timestamp_start,timestamp,timestamp_previous,function_exicution_time,dt,F_ref,rto
b_force,dob_force,counts,steps,position_x,velocity,I_motor,I_a_ref,analog_out_mot
or,F_pred,acceleration,F_x,F_v,F_a);

      }

      // force increment
      for(i=0;i<200000;i++)
      {

         if (abs(F_pred) < F_ref_max)
         {

            if (position_x!=0.0){
               F_x = (float)(F_res/position_x);
            }else{
               F_x=F_x;
            }
            if (velocity!=0.0){

               F_v = (float)(F_res/velocity);
            }else{
               F_v=F_v;
            }
            if (acceleration!=0.0){

               F_a = (float)(F_res/acceleration);
            }else{
               F_a=F_a;
            }
            unordered_map<string, string> sample;
            sample["position_x"]= to_string(position_x);
            sample["velocity"]= to_string(velocity);
            sample["acceleration"]= to_string(acceleration);
```

```
            sample["f_previous_x"]= to_string(F_x);
            sample["f_previous_v"]= to_string(F_v);
            sample["f_previous_a"]= to_string(F_a);

            force_predmodel= stof(model.predict(sample));
            F_pred =(float) (force_predmodel);
            // printf("F_pred:%f", F_pred);
            // F_pred = -10.0;
            F_ref = -F_pred;

         }else

         {
            F_pred = F_pred;
            F_ref = -F_pred;

         }
         F_res= F_pred;
         MainCalculation(board);
         fprintf(fp,"%d,%d,%d,%f,%f,%f,%f,%f,%d,%d,%f,%f,%f,%f,%u,
%f,%f,%f,%f,%f\n",

timestamp_start,timestamp,timestamp_previous,function_exicution_time,dt,F_ref,rto
b_force,dob_force,counts,steps,position_x,velocity,I_motor,I_a_ref,analog_out_mot
or,F_pred,acceleration,F_x,F_v,F_a);

      }

      for(i=0;i<5;i++)
      {
         F_pred=0.0;
         F_ref = F_pred;
         MainCalculation(board);
         fprintf(fp,"%d,%d,%d,%f,%f,%f,%f,%f,%d,%d,%f,%f,%f,%f,%u,
%f,%f,%f,%f,%f\n",

timestamp_start,timestamp,timestamp_previous,function_exicution_time,dt,F_ref,rto
b_force,dob_force,counts,steps,position_x,velocity,I_motor,I_a_ref,analog_out_mot
or,F_pred,acceleration,F_x,F_v,F_a);
      }

      fclose(fp);
   }else{
      // Injecting nevagive current, stopping cycle
      F_ref = 0.0;
      I_motor = 0.0;
      I_a_ref = 0.0;                              //stop the mortor input current
```

```
        MotorOutDAC(board);
        X826( S826_CounterStateWrite(board, 0, 0)   );            // Stop tracking encoder
position.
        return errcode;

    }
////////////////////////////////////////////////////////////
    // Injecting nevagive current, stopping cycle
    F_ref = 0.0;
    I_motor = 0.0;
    I_a_ref = 0.0;                          //stop the mortor input current
    MotorOutDAC(board);
    X826( S826_CounterStateWrite(board, 0, 0)   );            // Stop tracking encoder
position.

    return errcode;
}
// end of main function
////////////////////////////////////////////////////////////////////////////////////////////
// dt calculation
static float dtCalculation()
{
    function_exicution_time = (float)(timestamp - timestamp_start)/1000000;
//Timestamp in us , /10^6 to convert to s
    dt =(float)(timestamp - timestamp_previous)/1000000;
    timestamp_previous = timestamp;
    return 0;
}
// velocity calculation
static float VelocityCalculation()
{
    steps = 1 * counts;                                          // Correcting the encorder
read , + to the direction of squeezing
    position_x = steps*0.000005;
    velocity_sum = velocity_sum + velocity*dt;
    velocity = g_velocity * (position_x - velocity_sum);
    counts_pre = counts;
    return 0;
}
//Acceleration Calculation
static float AccelerationCalculation()
{
    // steps = 1 * counts;                      // Correcting the encorder read , + to the
direction of squeezing
    // position_x = steps*0.000005;
    acc_sum = acc_sum + acceleration*dt;
    acceleration = g_acc * (velocity - acc_sum);
```

```
   // counts_pre = counts;
   return 0;
}
// DOB
static float DoB()
{
   dob_input = velocity * g_dis * M_n;
   dob_filter_input = K_fn * I_motor + dob_input;
   dob_filter_output = dob_filter_output + g_dis*(dob_filter_input -
dob_filter_output)*dt;
   dob_force = dob_filter_output - dob_input;
   return 0;
}
// RTOB
static float RToB()
{
   rtob_filter_input = dob_filter_input - friction;
   rtob_filter_output = rtob_filter_output + g_dis*(rtob_filter_input -
rtob_filter_output)*dt;
   rtob_force = rtob_filter_output - dob_input;
   return 0;
}
// Motor drive input voltage genaration function, using dac
static int MotorOutDAC(uint board)
{
   if (I_motor < -0.59)
   {
      I_motor = -0.59;
   }else if(I_motor > 0.59)
   {
      I_motor = 0.59;
   }
   analog_out_motor = (unsigned int)((65535*0.5) + ((I_motor*65535)/(0.6*2))); //
DAC senstivity 16 bit (2**16)movo zero vlaue 0 -->  2.5V out in dac
   //analog_out_motor = (unsigned int)(32768 + (I_motor/0.6)*32768);          // DAC
senstivity 16 bit (2**16)movo zero vlaue 0 --> 0V out in dac (-5 - 5V)
   X826( S826_DacDataWrite(board, aout, analog_out_motor, 0) );               //
Analog value out from dac
   //X826( S826_DacDataWrite(board, chan, analog_out_motor, 0) );             //
Analog value out from dac
   return 0;                                                  // negative current - negative
cont, Positive current positive counter incriment
}

static float MainCalculation(uint board)
{
```

```
   I_a_ref = K_p*(F_ref - rtob_force)*(M_n/K_fn);          // Calculate from the
previous response/rtob_tyorque and dob_torque to get the I_motor adjusted with the
error
   I_motor = I_a_ref + (dob_force/K_fn);                  // the mortor input current
   //printf("  counts :%d",counts);
   //values for previous state
   dtCalculation();                                  // calculate dt, velocity, rotob_force,
dob_force according to the existing force command
   VelocityCalculation();
   AccelerationCalculation();
   DoB();
   RToB();
   MotorOutDAC(board);                               // DAC o/p motor current --> motor
v
   X826( S826_CounterSnapshot(board, 0) );                // Trigger snapshot on
counter 0.
   X826( S826_CounterSnapshotRead(board, 0, &counts, &timestamp, NULL, 0) );
// Read the snapshot:receive the snapshot info here; no need to wait for snapshot; it's
already been captured

   return 0;
}

////////////////////////////////////////////////////////////////////////////////////////////
// The demo.
static int DemoWatchdog(uint board)
{
   int rc;
   uint timing[5];
   // set timer 1 time-out to around 1 second
   timing[0] = 55000000;
   printf("\nDemoWatchdog\n");
   // enable writing to watchdog
   rc = S826_SafeWrenWrite(board, 0x02);
   if (rc != 0)
   {
      printf("failed to enable wren for watchdog\n");
      return rc;
   }
   rc = S826_WatchdogConfigWrite(board, 0x00, timing);

   if (rc != 0)
   {
      printf("failed to configure WD\n");
      return rc;
   }
```

```
    rc = S826_WatchdogEnableWrite(board, 1);
    if (rc != 0)
    {
        printf("failed to enable WD\n");
        return rc;
    }

    // commented out.  for testing watchdog cancel
    // CreateThread(NULL, 4096, testThread, board, 0, NULL);
    // watch indefinitely for watchdog
    rc = S826_WatchdogEventWait(board, S826_WAIT_INFINITE);

    switch (rc)
    {
        case 0:
            printf("watchdog successfully expired\n");
            break;
        case S826_ERR_NOTREADY:
            printf("WD wait timed out\n");
            break;
        case S826_ERR_CANCELLED:
            printf("WD wait cancelled\n");
            break;
        default:
        printf("error %d\n", rc);

    }
    // disable watchdog
    rc = S826_WatchdogEnableWrite(board, 0);
    if (rc != 0)
    {
        printf("failed to disable WD\n");
        return rc;
    }
    return 0;
}
```

- Position control

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// SENSORAY MODEL 826 PROGRAMMING EXAMPLES
// This file contains simple functions that show how to program the 826.
// Copyright (C) 2012 Sensoray
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// MSc setup - Position control
// Praveena Dewapura
// 2022.03.30
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
// Header functions declarations

#ifndef _LINUX
#include <windows.h>
#include <conio.h>
#endif

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

#ifndef _LINUX
#include "..\826api.h"
#else
#include "826api.h"
#endif

#include <iostream>
#include <unordered_map>

#include "cPMML.h"
using namespace std;
#include <sstream>
#include <string.h>

#include <vector>

// Helpful macros for DIOs
#define DIO(C)              ((uint64)1 << (C))                          // convert dio channel number to uint64 bit mask
#define DIOMASK(N)          {(uint)(N) & 0xFFFFFF, (uint)((N) >> 24)}   // convert uint64 bit mask to uint[2] array
#define DIOSTATE(STATES,CHAN)   ((STATES[CHAN / 24] >> (CHAN % 24)) & 1)    // extract dio channel's
boolean state from uint[2] array

////////////////////////////////////////////////////////////////////////////////////////
// ERROR HANDLING
// These examples employ very simple error handling: if an error is detected, the example functions will immediately return an
error code. This behavior may not be suitable for some
// real-world applications but it makes the code easier to read and understand. In a real application, it's likely that additional
actions would need to be performed. The examples use
// the following X826 macro to handle API function errors; it calls an API function and stores the returned value in errcode,
then returns immediately if an error was detected.
#define X826(FUNC)   if ((errcode = FUNC) != S826_ERR_OK) { printf("\nERROR: %d\n", errcode); return errcode;}
// end of hearder fuction declarations

////////////////////////////////////////////////////////////////////////////////////////
// Global variables declaration

// Encoder read

uint counts = 0;                    // encoder counts when the snapshot was captured
uint counts_pre =0;
uint timestamp;                     // time the snapshot was captured in us. converted to s by /10^6
uint timestamp_start;
uint timestamp_previous;            // time the previous snapshot was captured
uint aout = 0;                      // aout= analogue output channel no.0 , output duplicate waveform on this dac channel
uint chan = 4;                      // +++ Check the connection 41 - current- analog o/p chan 4
uint setpoint;
uint range;
int errcode = S826_ERR_OK;
int steps = 0;

//for loop oparation
long long int i ;
long long int j ;
char file_name[1000];
// char num[50];
int k = 0;
float position_err=0.0;
float position_err_pre=0.0;
float position_err_sum=0.0;
```

```
float function_exicution_time = 0.0;
float dt = 0.0;
float F_ref = 0.0;
float F_ref_max = 0.0;
float F_ref_min = 0.0;
float ramp_rate = 0.0;
float F_pred = 0.0;

float time_execution=0.0;
// motor drive oparation
unsigned int analog_out_motor;
float I_a_ref = 0.0;
float I_motor = 0.0;

// motor parameters
float K_fn = 24.0;                    // motor force constent
float M_n = 0.643;                    // Mass : (Motor shaft, 2 x linera bearings, 1 x Long green bearing and shft connector 1 x
short green                           //bearing and shft connector)=600g and (whight hammer head)=55g .   Mass : (Motor
shaft, 2 x linera bearings, 1 x Long green bearing and shft connector 1 x short green bearing and shft connector)=600g and
(whight hammer head)=55g . Changed the hammer head mass to 43 from 57 from elctronic balance, thus the new mass =  0.655
- 55 + 43 =0.643g
float friction = 0.0;                 // +++ Eventhough friction is assumed as 0 there is a 1N, the motor shaft doesn't move tll there
is 1N value shown in the                        //spring balance

// Velocity mesurment
float velocity_sum = 0.0;
float velocity = 0.0;                 // motor velocity
float g_velocity = 30.0;              // velocity filter constent
float position_x = 0.0;               // motor physical position
float position_x_prevous = 0.0;
float position_dx = 0.0;

float acc_sum =0.0;
float acceleration = 0.0;
float g_acc =100.0;
// DOb
float dob_input = 0.0;
float dob_filter_input = 0.0;
float dob_filter_output = 0.0;
int const g_dis = 300.0;              //Consider g_rec = g_dis
float dob_force = 0.0;

// RTOB
float rtob_filter_input = 0.0;
float rtob_filter_output = 0.0;
float rtob_force = 0.0;

// PID
float K_p = 2.0;

//Generate force response
float K_s = 2000.0;
float C_p=0.0;
float C_d=0.0;
float C_i=0.0;

float position_ex =0.0;
float position_ref =0.0;
// end of veriable declarations

std::vector< float > force_arr;
// std::vector< float > force_arr;
std::vector< float > position_arr;
std::vector< float > time_arr;
std::vector< float > dt_arr;

////////////////////////////////////////////////////////////////////////////////////////
// Function declaration

static float VelocityCalculation();
static int ControlLoop(uint board);
```

```
static int DemoWatchdog(uint board);
static int MotorOutDAC(uint board);
static float DoB();
static float RToB();
static float dtCalculation();
static float AccelerationCalculation();
static float timeCalculation();

// end of function declaration


////////////////////////////////////////////////////////////////////////////////////////
// Main function.
int main(int argc, char **argv)
{
   uint board     = 0;                          // 1 board, named as 0, change this if want to use other than board number 0
   int errcode    = S826_ERR_OK;
   int boardflags  = S826_SystemOpen();                  // open 826 driver and find all 826 boards

   if (argc > 1)
      board = atoi(argv[1]);

   if (boardflags < 0)
         errcode = boardflags;                                  // problem during open
      else if ((boardflags & (1 << board)) == 0)
   {
      int i;
         printf("TARGET BOARD of index %d NOT FOUND\n",board);             // driver didn't find board you want to use
         for (i = 0; i < 8; i++)
      {
         if (boardflags & (1 << i))
         {
            printf("board %d detected. try \"./s826demo %d\"\n", i, i);
               }
         }
   } else
   {
      // running functions
      X826( ControlLoop(board)          );                  // read the endocer value, output anlog value and file write
      X826( DemoWatchdog(board)         );                   // watchdog timer
   }

   switch (errcode)
   {
      case S826_ERR_OK:          break;
      case S826_ERR_BOARD:       printf("Illegal board number"); break;
      case S826_ERR_VALUE:       printf("Illegal argument"); break;
      case S826_ERR_NOTREADY:    printf("Device not ready or timeout"); break;
      case S826_ERR_CANCELLED:   printf("Wait cancelled"); break;
      case S826_ERR_DRIVER:      printf("Driver call failed"); break;
      case S826_ERR_MISSEDTRIG:  printf("Missed adc trigger"); break;
      case S826_ERR_DUPADDR:     printf("Two boards have same number");break;S826_SafeWrenWrite(board, 0x02);
      case S826_ERR_BOARDCLOSED: printf("Board not open"); break;
      case S826_ERR_CREATEMUTEX: printf("Can't create mutex"); break;
      case S826_ERR_MEMORYMAP:   printf("Can't map board"); break;
      default:                   printf("Unknown error"); break;
   }

#ifndef _LINUX
   printf("\nKeypress to exit ...\n\n");
   while (!_kbhit());
   _getch();
#endif

   S826_SystemClose();
   return 0;
}
// end of main function

////////////////////////////////////////////////////////////////////////////////////////
// ENCODER READ ANLOG OUT AND FILE WRITE FUCNTION
```

```
// JKD - 10.08.2020

static int ControlLoop(uint board)
{
  // ***Configure interfaces and start them running.
  X826( S826_DacRangeWrite(board, aout, S826_DAC_SPAN_0_5, 0));         // program dac output range: -5V to +5V , motor
0 value is 0V / for motor drive operation , IF SO SHOULD CHANGE FROM THE MOTOR DRIVER'S SIDE
  //X826( S826_DacRangeWrite(board, chan, S826_DAC_SPAN_0_5, 0));       // program dac output range: -0V to +5V , motor
0 value is 2.5V / for motor drive operation ,
  X826( S826_CounterModeWrite(board, 0, S826_CM_K_QUADX4));             // Configure counter0 as incremental encoder
interface. quadrature
  X826( S826_CounterStateWrite(board, 0, 1));                          // Start tracking encoder position.

  I_motor = 0.0;
  position_x = 0.0;
  //stop the mortor input current
  MotorOutDAC(board);  //- remove
  X826( S826_CounterSnapshot(board, 0) );                                // Trigger snapshot on counter 0.
  X826( S826_CounterSnapshotRead(board, 0, &counts, &timestamp_start, NULL, 0));     // Read the snapshot: from the
counter0


  F_ref_min = 0.0;
  F_ref = 0.0;
  I_a_ref = 0.0;
  timestamp_previous = timestamp_start;

  X826( S826_CounterSnapshot(board, 0)                    );             // Trigger snapshot on counter 0.
  X826( S826_CounterSnapshotRead(board, 0, &counts, &timestamp, NULL, 0)    );      // Read the snapshot:receive the
snapshot info here;no need to wait for snapshot;it's already been captured
  // Important to calculate dt, velocity before dob, rtob calculations
  dtCalculation();
  VelocityCalculation();
  AccelerationCalculation();

  ////////////////////////////////////////////////////////////////////////////
  ramp_rate =4.2;       // 2.1 , 2.4 , 2.7 , 3.0               // to change the force ramping
    //zero
  F_ref_max = 1.0;      // 6 , 9 , 12 , 15

  // ****file write operation
  k=1;
  // for(j=0;j<15;j++){
  FILE *fp;
  fp= fopen("test_pmml/testdata_ml_pmml_new_2022_working_2.csv", "r");                          // file pointer file file
operation

  if (!fp)
    printf("Can't open file\n");

  else {
    char buffer[1024];

    int row = 0;
    int column = 0;

    while (fgets(buffer,
                 1024, fp)) {
      column = 0;
      row++;

      // To avoid printing of column      // names in file can be changed     // according to need
      if (row == 1)
        continue;

      // Splitting the data
      char* value = strtok(buffer, ", ");

      while (value) {

        k=0;
```

```
      // Column 4
      if (column == 3) {
        time_arr.push_back(stof(value));
      }
      // Column 5
      if (column == 4) {
        dt_arr.push_back(stof(value));
      }
      // Column 7
      if (column == 6) {
        force_arr.push_back(stof(value));
      }
      // Column 11
      if (column == 10) {
        position_arr.push_back(stof(value));
      //   printf("%s", value);
      }

      value = strtok(NULL, ", ");
      column++;
      // k++;
    }
  //   printf("\n");
   }
   fclose(fp);                          // Close the file
  }


  FILE *fp1;
  fp1 = fopen("test_position/testdata_ml_sponge_position_new.csv", "a+");

fprintf(fp1,"timestamp_start,timestamp,timestamp_previous,function_exicution_time,dt,F_ref,rtob_force,dob_force,counts,step
s,position_x,velocity,I_motor,I_a_ref,analog_out_motor,F_pred,acceleration,time_execution,position_ref\n");

  C_p=700.0;
  C_i = 1000.0;
  C_d =12.0;

   for (i = 0; i < 200010; i++){

   if(position_arr[i]>0){
       position_ex=position_arr[i];
   }else{
       position_ex=position_ex;
   }
   position_ref=position_ex;
   //position_ref =0.01;
   position_err = position_ref - position_x;
   position_dx = ((position_err-position_err_pre)/dt);
   position_err_sum =position_err_sum+(position_err*dt);
   I_a_ref = ((C_p*(position_err))+(C_d*position_dx)+(C_i*position_err_sum))*(M_n/K_fn);
   //I_motor = I_a_ref + (dob_force/K_fn);
   I_motor = I_a_ref ;


   time_execution = time_arr[i];
   while(function_exicution_time<time_execution){
       MotorOutDAC(board);                                          // the mortor input current to run
       X826( S826_CounterSnapshot(board, 0)                          );      // Trigger snapshot on counter 0.
       X826( S826_CounterSnapshotRead(board, 0, &counts, &timestamp, NULL, 0)  );      // Read the snapshot:receive the
snapshot info here; no need to wait for snapshot; it's already been captured
       timeCalculation();
   }
   dtCalculation();                                             // calculate dt, velocity, rotob_force, dob_force according to the
existing force command
   VelocityCalculation();
   AccelerationCalculation();
   DoB();
   RToB();
```

```
   fprintf(fp,"%d,%d,%d,%f,%f,%f,%f,%f,%d,%d,%f,%f,%f,%f,%f,%u,%f,%f,%f,%f\n",
timestamp_start,timestamp,timestamp_previous,function_exicution_time,dt,F_ref,rtob_force,dob_force,counts,steps,position_x,
velocity,I_motor,I_a_ref,analog_out_motor,F_pred, acceleration,time_execution,position_ref);
   position_err_pre =position_err;
   }
   fclose(fp1);

  ////////////////////////////////////// ////////////////////

  // Injecting nevagive current, stopping cycle
  F_ref = 0.0;
  I_motor = 0.0;
  I_a_ref = 0.0;                              //stop the mortor input current
  MotorOutDAC(board);
  X826( S826_CounterStateWrite(board, 0, 0) );              // Stop tracking encoder position.

  return errcode;
}
// end of main function
////////////////////////////////////////////////////////////////////////////////////////////
// time calculation
static float timeCalculation()
{
  function_exicution_time = (float)(timestamp - timestamp_start)/1000000;     //Timestamp in us , /10^6 to convert to s
  // dt =(float)(timestamp - timestamp_previous)/1000000;
  // timestamp_previous = timestamp;
  return 0;
}
// dt calculation
static float dtCalculation()
{
  // function_exicution_time = (float)(timestamp - timestamp_start)/1000000;  //Timestamp in us , /10^6 to convert to s
  dt =(float)(timestamp - timestamp_previous)/1000000;
  timestamp_previous = timestamp;
  return 0;
}
// velocitycalculation fucntions with function
static float VelocityCalculation()
{
  steps = -1 * counts;                                          // Correcting the encorder read , + to the direction of squeezing
  position_x = steps*0.000005;
  velocity_sum = velocity_sum + velocity*dt;
  velocity = g_velocity * (position_x - velocity_sum);
  counts_pre = counts;
  return 0;
}
static float AccelerationCalculation()
{
  // steps = 1 * counts;                                          // Correcting the encorder read , + to the direction of squeezing
  // position_x = steps*0.000005;
  acc_sum = acc_sum + acceleration*dt;
  acceleration = g_acc * (velocity - acc_sum);
  // counts_pre = counts;
  return 0;
}
// DOB
static float DoB()
{
  dob_input = velocity * g_dis * M_n;
  dob_filter_input = K_fn * I_motor + dob_input;
  dob_filter_output = dob_filter_output + g_dis*(dob_filter_input - dob_filter_output)*dt;
  dob_force = dob_filter_output - dob_input;
  return 0;
}
// RTOB
static float RToB()
{
  rtob_filter_input = dob_filter_input - friction;
  rtob_filter_output = rtob_filter_output + g_dis*(rtob_filter_input - rtob_filter_output)*dt;
  rtob_force = rtob_filter_output - dob_input;
  return 0;
```

```
}
// Motor drive input volate genarating function, using dac
static int MotorOutDAC(uint board)
{
  if (I_motor < -0.59)
  {
    I_motor = -0.59;
  }else if(I_motor > 0.59)
  {
    I_motor = 0.59;
  }
  analog_out_motor = (unsigned int)((65535*0.5) + ((I_motor*65535)/(0.6*2)));   // DAC senstivity 16 bit (2**16)movo zero
vlaue 0 --> 0V out in dac (-5 - 5V)
  //analog_out_motor = (unsigned int)(32768 + (I_motor/0.6)*32768);            // DAC senstivity 16 bit (2**16)movo zero vlaue
0 -->  2.5V out in dac
  X826( S826_DacDataWrite(board, aout, analog_out_motor, 0) );                // Analog value out from dac
  //X826( S826_DacDataWrite(board, chan, analog_out_motor, 0) );               // Analog value out from dac
  return 0;                                                                   // negativa current - negative cont, Positive current positive counter
incrimet
}
////////////////////////////////////////////////////////////////////////////////////////////
// The demo.
static int DemoWatchdog(uint board)
{
  int rc;
  uint timing[5];
  // set timer 1 time-out to around 1 second
  timing[0] = 55000000;
  printf("\nDemoWatchdog\n");
  // enable writing to watchdog
  rc = S826_SafeWrenWrite(board, 0x02);
  if (rc != 0)
  {
    printf("failed to enable wren for watchdog\n");
    return rc;
  }
  rc = S826_WatchdogConfigWrite(board, 0x00, timing);

  if (rc != 0)
  {
    printf("failed to configure WD\n");
    return rc;
  }

  rc = S826_WatchdogEnableWrite(board, 1);
  if (rc != 0)
  {
    printf("failed to enable WD\n");
    return rc;
  }

  // commented out.  for testing watchdog cancel
  // CreateThread(NULL, 4096, testThread, board, 0, NULL);
  // watch indefinitely for watchdog
  rc = S826_WatchdogEventWait(board, S826_WAIT_INFINITE);

  switch (rc)
  {
  case 0:
    printf("watchdog successfully expired\n");
    break;
  case S826_ERR_NOTREADY:
    printf("WD wait timed out\n");
    break;
  case S826_ERR_CANCELLED:
    printf("WD wait cancelled\n");
    break;
  default:
    printf("error %d\n", rc);

  }
```

```
  // disable watchdog
  rc = S826_WatchdogEnableWrite(board, 0);
  if (rc != 0)
  {
    printf("failed to disable WD\n");
    return rc;
  }
  return 0;
}
```

APPENDIX E AI codes

```
#!pip install tf-nightly-gpu-2.0-preview
import tensorflow as tf
%tensorflow_version 2.x
#import tensorflow as tf

device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
from google.colab import drive
drive.mount('/content/drive')

from __future__ import print_function
from __future__ import division
#Import all required libraries
import pandas as pd #loading data in table form  # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns #visualisation
import matplotlib
from matplotlib import cm
import matplotlib.pyplot as plt #visualisation
#from matplotlib import pyplot as plt
import numpy as np # linear algebra
from sklearn.preprocessing import normalize #machine learning algorithm library

import keras #library for neural network
import keras.backend as kb

#Neural network module
from keras.models import Sequential
from keras.layers import Dense,Activation,Dropout
# from keras.layers.normalization import BatchNormalization
from keras.utils import np_utils
from keras.models import model_from_json
import os

# importing the csv module
import csv



%matplotlib inline

import statsmodels.api as sm
import xgboost as xgb
```

```
from sklearn import tree
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
#from sklearn.metrics import mean_squared_error

import sklearn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import scale
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import Ridge
from sklearn.svm import SVR

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import make_pipeline

import json
from google.colab import files
import gspread
from gspread_dataframe import get_as_dataframe,set_with_dataframe

import warnings # supress warnings
warnings.filterwarnings('ignore')



df = pd.read_csv('/content/drive/MyDrive/Msc/ML Models/dataset/spongedataset.csv')
df.describe()
df.info()

df.isna().sum()
df =df.dropna()
df
r = df.index[np.isinf(df).any(1)]
print(r)
array_remove1 = [x for x in range(df.shape[0]) if x in r] # Only Time_difference ==0, Position_difference ==1/-1
df = df.drop(labels=array_remove1, axis=0)
df



fig, ax = plt.subplots(figsize=(20, 12)) # Create the figure and axes object
# Create another figure
```

```
# plt.figure(figsize=(20, 12))
plt.rcParams.update({'font.size': 30})
# Plot the first x and y axes:
df.plot(x = 'function_exicution_time', y = 'rtob_force',label='$F_{res}$ : Sponge', ax = ax, color ='darkgoldenrod')
df.plot(x = 'function_exicution_time', y = 'predictions_random_forest', label='$F_{res}$ : AI',ax = ax, color ='tomato')
ax.set_ylabel('Force (N)',fontsize=30)
ax.set_xlabel('Time (s)',fontsize=30)
# Show the major grid lines with dark grey lines
plt.grid(b=True, which='major', color='#666666', linestyle='-')
# Show the minor grid lines with very faint and almost transparent grey lines
plt.minorticks_on()
plt.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=0.2)

fig, ax = plt.subplots(figsize=(20, 12)) # Create the figure and axes object
plt.rcParams.update({'font.size': 30})
df.plot(x = 'function_exicution_time', y = 'position_ref',label='Position_ref', ax = ax, color ='blue')
df.plot(x = 'function_exicution_time', y = 'position_x', label='Position',ax = ax, color ='red')
ax.set_ylabel('Compression depth (m)',fontsize=30)
ax.set_xlabel('Time (s)',fontsize=30)
plt.grid(b=True, which='major', color='#666666', linestyle='-')
plt.minorticks_on()
plt.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=0.2)

fig, ax = plt.subplots(figsize=(20, 12)) # Create the figure and axes object
plt.rcParams.update({'font.size': 30})
df.plot(x = 'function_exicution_time', y = 'velocity',label='Velocity', ax = ax, color ='palevioletred')
ax.set_ylabel('Velocity ($\mathregular{ms^{-1}}$)',fontsize=30)
ax.set_xlabel('Time (s)',fontsize=30)
plt.grid(b=True, which='major', color='#666666', linestyle='-')
plt.minorticks_on()
plt.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=0.2)

df['force_previous']=df.rtob_force.shift(1, axis = 0)
df['force_previous'][0]=0
f_x_h = []
f_v_h = []
f_a_h = []
f_x=0.0
f_v=0.0
f_a=0.0
for x in range(0, int(df.shape[0])):
    position = float (df_pred2.position_x.iloc[[x]])
    velocity = float (df.velocity.iloc[[x]])
    acceleration = float (df.acceleration.iloc[[x]])
    force=float (df.force_previous.iloc[[x]])

    if position==0:
```

```
    f_x= f_x
  else:
    f_x=force/position

  if velocity==0:
    f_v= f_v
  else:
    f_v=force/velocity

  if acceleration==0:
    f_a= f_a
  else:
    f_a=force/acceleration

  f_x_h.append(f_x)
  f_v_h.append(f_v)
  f_a_h.append(f_a)


df_v_108_area =pd.DataFrame()
df_v_108_area["Time"] =df_v_108.Time
df_v_108_area["Position"] =df_v_108.Position
df_v_108_area["Force_Response_adj"] =df_v_108.Force_Response_adj
df_v_108_area["Position_Diff"] =df_v_108.Position_difference
df_v_108_area["F*P"] =  (df_v_108.Force_Response_adj * df_v_108_area.Position_Diff*0.000005).abs()
df_v_108_area["F*T"] = df_v_108.Force_Response_adj* df_v_108.Time_Duration
df_v_108_area

df_v_108_area["Cumsum_F*P"]=0
df_v_108_area["Cumsum_F*P"][:(1200000)]= df_v_108_area["F*P"][:(1200000)].cumsum()
df_v_108_area["Cumsum_F*P"][(1200000):(2400000)]= df_v_108_area["F*P"][(1200000):(2400000)].cumsum()
df_v_108_area["Cumsum_F*P"][(2400000):(3800000)]= df_v_108_area["F*P"][(2400000):(3800000)].cumsum()
df_v_108_area["Cumsum_F*P"][(3800000):(5200000)]= df_v_108_area["F*P"][(3800000):(5200000)].cumsum()
df_v_108_area["Cumsum_F*P"][(5200000):(6800000)]= df_v_108_area["F*P"][(5200000):(6800000)].cumsum()
df_v_108_area["Cumsum_F*P"][(6800000):(8400000)]= df_v_108_area["F*P"][(6800000):(8400000)].cumsum()
df_v_108_area["Cumsum_F*P"][(8400000):(10200000)]= df_v_108_area["F*P"][(8400000):(10200000)].cumsum()
df_v_108_area["Cumsum_F*P"][(10200000):(12000000)]= df_v_108_area["F*P"][(10200000):(12000000)].cumsum()
df_cumsum_pt= pd.concat([df_v_108_area["Cumsum_F*P"].iloc[[(1200000-1)]],
  df_v_108_area["Cumsum_F*P"].iloc[[(2400000-1)]], df_v_108_area["Cumsum_F*P"].iloc[[(3800000-1)]],
  df_v_108_area["Cumsum_F*P"].iloc[[(5200000-1)]],df_v_108_area["Cumsum_F*P"].iloc[[(6800000-1)]],
  df_v_108_area["Cumsum_F*P"].iloc[[(8400000-1)]],df_v_108_area["Cumsum_F*P"].iloc[[(10200000-1)]],
  df_v_108_area["Cumsum_F*P"].iloc[[(12000000-1)]]], ignore_index=True)
df_cumsum_pt = pd.DataFrame(df_cumsum_pt)
# df_deformation.columns = [""]
df_cumsum_pt["Datapoint"] = [1200000,2400000,3800000,5200000,6800000,8400000,10200000,12000000]
pt_previous= np.array(df_cumsum_pt["Cumsum_F*P"])
pt_previous = np.insert(pt_previous,0,df_v_108_area["Cumsum_F*P"].iloc[[0]])
```

```
pt_previous = np.resize(pt_previous, df_cumsum_pt["Cumsum_F*P"].size)
df_cumsum_pt['FP_previous'] = pt_previous
df_cumsum_pt['FP_diff']= (df_cumsum_pt['FP_previous'] - df_cumsum_pt["Cumsum_F*P"]).abs()
df_cumsum_pt
cumsum_diff= [0]

for i in range(0, df_cumsum_pt.shape[0],2):
  cumsum_rising = float (df_cumsum_pt['FP_diff'].iloc[[i]])
  cumsum_falling = float (df_cumsum_pt['FP_diff'].iloc[[i+1]])
  cumsum_diff_element = float (cumsum_rising - cumsum_falling)
  # print(i,cumsum_rising, cumsum_falling, cumsum_diff_element)
  cumsum_diff.append(cumsum_diff_element)
cumsum_diff = pd.DataFrame(cumsum_diff)
cumsum_diff

df_deformation= pd.concat([df_v_108.Position.iloc[[0]],
    df_v_108.Position.iloc[[2400000-1]],
    df_v_108.Position.iloc[[5200000-1]],
    df_v_108.Position.iloc[[8400000-1]],
     df_v_108.Position.iloc[[12000000-1]]], ignore_index=True)
df_deformation = pd.DataFrame(df_deformation)
# df_deformation.columns = [""]
df_deformation
position_previous= np.array(df_deformation.Position)
position_previous = np.insert(position_previous,0,0)
position_previous = np.resize(position_previous, df_deformation['Position'].size)
df_deformation['Position_previous'] = position_previous
# haptics_data['Position_previous'] = position_previous*0.005
# haptics_data['Position'] = haptics_data['Position'] *0.005
df_deformation['Deformation']=df_deformation.Position - df_deformation.Position_previous
df_deformation

df_v_1=df_v[:2400000]
df_v_1["Cycle no."] = 1
df_v_2=df_v[2400000:5200000]
df_v_2["Cycle no."] = 2
df_v_3=df_v[5200000:8400000]
df_v_3["Cycle no."] = 3
df_v_4=df_v[8400000:12000000]
df_v_4["Cycle no."] = 4
np.random.seed(42)

# Shuffle the data
# df_train_shuffled = df_train.sample(frac=1)
haptics_data_50_shuffled = haptics_data_50.sample(frac=1)
# data normalization with sklearn
from sklearn.preprocessing import MinMaxScaler
```

```
norm = MinMaxScaler().fit(haptics_data_50_shuffled)
# transform training data
df_norm = norm.transform(haptics_data_50_shuffled)
df_norm = pd.DataFrame(df_norm)
# adding column name to the respective columns
df_norm.columns =haptics_data_50_shuffled.columns
df_norm.describe()

df1= pd.read_csv(urlip11)
df2= pd.read_csv(urlip12)
df4= pd.read_csv(urlip14)
df5= pd.read_csv(urlip15)
df= pd.concat([df1,df2,df4,df5], ignore_index=True)
df = df.drop('Unnamed: 0',axis =1)
df.T

array1=np.arange(0,df.shape[0],2)
array_remove1 = [x for x in range(df.shape[0]) if x not in array1]
df = df.drop(labels=array_remove1, axis=0)
df.T
df.columns

X_tp_train =df_train_shuffled[['function_exicution_time',
      'position_x', 'velocity', 'acceleration',  'f_previous_x',
      'f_previous_v', 'f_previous_a']]
y_tp_train = df_train_shuffled["Force_Response_adj"]

X_tp_test1=X_tp_test
X_tp_test=X_tp_test.drop(["function_exicution_time"],axis=1)
# Split data into X and y
# X_tp = df_norm.drop([ 'Force_Command', 'Force_Response','Force_Command_adj', 'Time_Duration','Position_difference',
'Deformation', 'Area_FP','Force_Response_adj'] ,axis=1)
X_tp = df[["function_exicution_time","position_x","velocity",'acceleration']]
y_tp = df["Force_Response_adj"]

# Split into train & test set
# X_tp_train, X_tp_test, y_tp_train, y_tp_test = train_test_split(X_tp,  y_tp, test_size=0.3)

X_tp_train, X_tp_test, y_tp_train, y_tp_test = train_test_split(X_tp,
                                                y_tp,
                                                test_size=0.3,shuffle = True, random_state=42)

# example of correlation feature selection for numerical data
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression
```

```
from matplotlib import pyplot

# feature selection
def select_features(X_train, y_train, X_test):
  # configure to select all features
  fs = SelectKBest(score_func=f_regression, k='all')
  # learn relationship from training data
  fs.fit(X_train, y_train)
  # transform train input data
  X_train_fs = fs.transform(X_train)
  # transform test input data
  X_test_fs = fs.transform(X_test)
  return X_train_fs, X_test_fs, fs

plt.rcParams.update({'font.size': 30})
fig, ax = plt.subplots(figsize=(20, 12))
# load the dataset
# X, y = make_regression(n_samples=1000, n_features=100, n_informative=10, noise=0.1, random_state=1)
# split into train and test sets

# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)
# feature selection
# X_train_fs, X_test_fs, fs = select_features(X_train, y_train, X_test)
X_train_fs, X_test_fs, fs = select_features(X_tp_train, y_tp_train, X_tp_test)
# what are scores for the features
for i in range(len(fs.scores_)):
  print('Feature %d: %f' % (i, fs.scores_[i]))
# plot the scores
pyplot.bar([i for i in range(len(fs.scores_))], fs.scores_)
pyplot.show()

a= ['position_x', 'velocity', 'acceleration',  'f_previous_x',
      'f_previous_v', 'f_previous_a', 'cycle_No','deformation', 'Area_FP']
plt.rcParams.update({'font.size': 30})
fig, ax = plt.subplots(figsize=(20, 12))
pyplot.xticks(rotation=45)
pyplot.bar(a, fs.scores_)
pyplot.show()

# feature selection
def select_features(X_train, y_train, X_test):
  # configure to select all features
  fs = SelectKBest(score_func=mutual_info_regression, k='all')
  # learn relationship from training data
  fs.fit(X_train, y_train)
  # transform train input data
  X_train_fs = fs.transform(X_train)
```

```
 # transform test input data
 X_test_fs = fs.transform(X_test)
 return X_train_fs, X_test_fs, fs
plt.rcParams.update({'font.size': 30})
fig, ax = plt.subplots(figsize=(20, 12))
# load the dataset
# X, y = make_regression(n_samples=1000, n_features=100, n_informative=10, noise=0.1, random_state=1)
# split into train and test sets
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)
# feature selection
X_train_fs, X_test_fs, fs = select_features(X_tp_train, y_tp_train, X_tp_test)
# what are scores for the features
for i in range(len(fs.scores_)):
  print('Feature %d: %f' % (i, fs.scores_[i]))
# plot the scores
pyplot.bar([i for i in range(len(fs.scores_))], fs.scores_)
pyplot.show()
plt.rcParams.update({'font.size': 30})
fig, ax = plt.subplots(figsize=(20, 12))
pyplot.xticks(rotation=45)

pyplot.bar(a, fs.scores_)
pyplot.show()

df_corr = df_train_shuffled[['position_x', 'velocity',   'acceleration', 'f_previous_x', 'f_previous_v',
'f_previous_a','cycle_No','deformation', 'Area_FP','Force_Response_adj']]
# Let's make our correlation matrix a little prettier
corr_matrix = df_corr.corr()
plt.rcParams.update({'font.size': 20})
fig, ax = plt.subplots(figsize=(15, 10))
ax = sns.heatmap(corr_matrix,
                 annot=True,
                 linewidths=0.5,
                 fmt=".2f",
                 cmap="YlGnBu");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)

correlations = df_corr.corr()
plt.rcParams.update({'font.size': 20})
fig, ax = plt.subplots(figsize=(15, 10))

# San Juan
(correlations
    .Force_Response_adj
    .drop('Force_Response_adj') # don't compare with myself
    .sort_values(ascending=False)
```

```
    .plot
    .barh())

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# file = 'https://aegis4048.github.io/downloads/notebooks/sample_data/unconv_MV_v5.csv'
# df = pd.read_csv(file)
# df = df.iloc[:, 1:-1]

corr = df_corr.corr(method='spearman')
# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
fig, ax = plt.subplots(figsize=(30, 20))
# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True, sep=100)
# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmin=-1, vmax=1, center=0, linewidths=.5,
            annot=True,
            fmt=".2f",)
fig.suptitle('Correlation matrix of features', fontsize=15)
ax.text(0.77, 0.2, 'aegis4048.github.io', fontsize=13, ha='center', va='center',
        transform=ax.transAxes, color='grey', alpha=0.5)
fig.tight_layout()

# k-fold CV (using all the 13 variables)
lm = LinearRegression()
# scores = cross_val_score(lm, X_train, y_train, scoring='r2', cv=5)
# scores
# lm.fit(X, y)
lm.fit(X_tp_train, y_tp_train)

# predictions_linear_reg = lm.predict(X_tp_test.drop(["Time"],axis=1))
predictions_linear_reg = lm.predict(X_tp_test1.drop(["function_exicution_time"],axis=1))

from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
print("R2 score : %.2f" % r2_score(y_tp_test,predictions_linear_reg))
print("mean_absolute_error : %.2f" % mean_absolute_error(y_tp_test,predictions_linear_reg))
print("mean_squared_error : %.2f" % mean_squared_error(y_tp_test,predictions_linear_reg))
print("root_mean_squared_error : %.2f" % np.sqrt(mean_squared_error(y_tp_test,predictions_linear_reg)))
X_tp_test=X_tp_test.sort_values(by=['function_exicution_time'])
X_tp_test
```

```
X_tp_test1.to_csv("/content/drive/MyDrive/Msc/ML_Models/mlmodel_prashhaptics/dataset/test_lm.csv")

# Instantiate model with 1000 decision trees
rf = RandomForestRegressor(n_estimators=1000, max_depth=9,random_state=50)
# rf = RandomForestRegressor()
# Train the model on training data
rf.fit(X_tp_train, y_tp_train);

import os
import joblib
import numpy as np
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
joblib.dump(rf,
"/content/drive/MyDrive/Msc/ML_Models/mlmodel_prashwi/dataset/test_02/adj/x_test_all_features_rf_5var.joblib",
compress=3)  # compression is ON!
print(f"Compressed Random Forest:
{np.round(os.path.getsize('/content/drive/MyDrive/Msc/ML_Models/mlmodel_prashwi/dataset/test_02/adj/x_test_all_features_
rf_5var.joblib') / 1024 / 1024, 2) } MB")
# >>> Compressed Random Forest: 0.03 MB

loaded_model =
joblib.load('/content/drive/MyDrive/Msc/ML_Models/mlmodel_prashwi/dataset/test_02/adj/x_test_all_features_rf_5var.joblib')
# result = loaded_model.score(X_test, Y_test)
# print(result)

show_scores(rf)
# Create evaluation function (the competition uses RMSLE)
from sklearn.metrics import mean_squared_log_error, mean_absolute_error, r2_score, mean_squared_error

# def rmsle(y_test, y_preds):
#     """
#     Caculates root mean squared log error between predictions and
#     true labels.
#     """
#     return np.sqrt(mean_squared_log_error(y_test, y_preds))

def rmse(y_test, y_preds):
    """
    Caculates root mean squared log error between predictions and
    true labels.
    """
    return np.sqrt(mean_squared_error(y_test, y_preds))
# Model evaluation metrics documentation - https://scikit-learn.org/stable/modules/model_evaluation.html
# R^2 (pronounced r-squared) or coefficient of determination.
# Mean absolute error (MAE)
# Mean squared error (MSE)
```

```
# Create function to evaluate model on a few different levels
def show_scores(model):
    train_preds = model.predict(X_tp_train)
    test_preds = model.predict(X_tp_test)
    scores = {"Training MAE": mean_absolute_error(y_tp_train, train_preds),
              "Test MAE": mean_absolute_error(y_tp_test, test_preds),
              # "Training RMSLE": rmsle(y_train, train_preds),
              # "Valid RMSLE": rmsle(y_valid, val_preds),
              "Training MSE": mean_squared_error(y_tp_train, train_preds),
              "Test MSE": mean_squared_error(y_tp_test, test_preds),
              "Training RMSE": rmse(y_tp_train, train_preds),
              "Test RMSE": rmse(y_tp_test, test_preds),
              "Training R^2": r2_score(y_tp_train, train_preds),
              "Test R^2": r2_score(y_tp_test, test_preds)}
    return scores

# svr = SVR()
# svr = SVR(kernel='rbf', degree=3, gamma='scale', coef0=0.0, tol=0.001, C=0.1, epsilon=0.1, shrinking=True,
cache_size=200, verbose=False, max_iter=-1)
# svr = SVR(kernel='rbf', gamma=0.002, tol=0.001, C=0.1, epsilon=0.0001)
svr = SVR(kernel='rbf', gamma=0.002, C=0.1, epsilon=0.0001)
# Train the model on training data
svr.fit(X_tp_train, y_tp_train);
# create NN model
forcemodel_tp = Sequential()
forcemodel_tp.add(Dense(150, input_dim=6, activation='sigmoid'))
forcemodel_tp.add(Dropout(0.2))
forcemodel_tp.add(Dense(25, activation='tanh'))
forcemodel_tp.add(Dropout(0.2))
forcemodel_tp.add(Dense(1, activation='sigmoid'))

forcemodel_tp.summary()
```

```
forcemodel_tp.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 dense_3 (Dense)              (None, 150)               1050

 dropout_2 (Dropout)          (None, 150)               0

 dense_4 (Dense)              (None, 25)                3775

 dropout_3 (Dropout)          (None, 25)                0

 dense_5 (Dense)              (None, 1)                 26

=================================================================
Total params: 4,851
Trainable params: 4,851
Non-trainable params: 0
_________________________________________________________________
```

```
# Compile model
opt = tf.keras.optimizers.SGD(learning_rate=0.000001)
forcemodel_tp.compile(loss='mean_squared_error', optimizer=opt)
# Fit the model
%%time
forcemodel_tp.fit(x=X_tp_train, y=y_tp_train, epochs=100, batch_size=5, validation_split=0.20)

opt = tf.keras.optimizers.Adam(learning_rate=0.000001)


model.summary()
```

```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 lstm (LSTM)                  (None, 100)               47600

 dense_6 (Dense)              (None, 1)                 101

=================================================================
Total params: 47,701
Trainable params: 47,701
Non-trainable params: 0
_________________________________________________________________
```

```
from keras.layers.recurrent import LSTM
# design network
model = Sequential()
model.add(LSTM(100, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))
model.compile(loss='mae', optimizer=opt)
# fit network
history = model.fit(train_X, train_y, epochs=50, batch_size=72, validation_data=(test_X, test_y), verbose=2, shuffle=False)
# plot history
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
plt.legend()
plt.show()
```

APPENDIX F Important R codes

- Reconstruction

```
install.packages('devtools')
install.packages("keras")
install.packages("mlbench")
install.packages("dplyr")
install.packages("magrittr")
install.packages("neuralnet")
install.packages("nnet")
install.packages("tensorflow")
install.packages("pmml")
install.packages("keras2pmml")
install.packages("r2pmml")
install.packages("pmmlTransformations")
install.packages("randomForest")
install.packages("readr") #https://cran.r-project.org/web/packages/readr/readme/README.html
#  Libraries
library(randomForest)
library(XML)
library(pmml)
library("readr")
# data
path <- "/content/testdata_train_shuffle.csv"
# path <-"/home/linux/Documents/Rstudio/data/testdata_train_shuffle.csv"

# reading contents of csv file
# content <- read_csv(path, col_names = TRUE, row.names = FALSE)
content <- read_csv(path, col_names = TRUE)

# Buid a rf model
# keeps <- c("position_x","velocity","acceleration","f_previous_x","f_previous_v","f_previous_a","Force_Response_adj")
# content = content[keeps]
print (content)
#Working in r and c++
## Not run:
# Build a randomForest model
force_rf <- randomForest(Force_Response_adj ~ ., data = content, ntree = 1000)
# fit <- nnet(Species ~ ., data = iris, size = 4)
# Convert to pmml
# force_rf_pmml <- pmml(force_rf)
force_rf_pmml <- pmml(force_rf)
rm(force_rf) #delete objects from the memory.
save_pmml(force_rf_pmml, "Force_Response_adj_pmml.pmml")
```