FPGA IMPLEMENTATION OF MACHINE LEARNING ALGORITHMS FOR VIBROTACTILE FEEDBACK IN PROSTHESES

by

İsmail ERBAŞ

B.S., in Electrical and Electronics Engineering, Anadolu University, 2019

Submitted to the Institute of Biomedical Engineering
in partial fulfillment of the requirements
for the degree of
Master of Science
in
Biomedical Engineering

Boğaziçi University
2021

To my father...

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my thesis supervisor Prof. Dr. Burak Güçlü for his patience, support, help and motivation through my study. I also would like to thank Prof. Dr. Mehmed Özkan and Prof. Dr. Volkan Patoğlu for their invaluable comments and time.

I would like to thank to my brother and mother for their endless love and support through my life. They make the life meaningful for me.

I would like to thank to the members of Tactile Research Laboratory for their support and friendship. I thank to İpek Karakuş, Deniz Kılınç Bülbül, Begüm Devlet for their frendship, constant motivation, help and support during my thesis. They always helped me patiently whenever I needed. I thank to all participants, who joined the experiments for their time and patience.

Finally, I would like to thank my best friends Kağan Ceylan, Faruk Cürebal, Sercan Gelir, Bilge Alır and Ferhat Demirkıran for their endless friendship and support. They always bring joy to my life.

This work was supported by TÜBİTAK Grant 117F481 within European Union's FLAG-ERA JTC 2017 project GRAFIN.

ACADEMIC ETHICS AND INTEGRITY STATEMENT

I, İsmail ERBAŞ, hereby certify that I am aware of the Academic Ethics and Integrity Policy issued by the Council of Higher Education (YÖK) and I fully acknowledge all the consequences due to its violation by plagiarism or any other way.

Name :

Signature:

Date:

ABSTRACT

FPGA IMPLEMENTATION OF MACHINE LEARNING ALGORITHMS FOR VIBROTACTILE FEEDBACK IN PROSTHESES

This study aimed to apply discrete event-driven vibrotactile feedback using machine learning algorithms in real time. Previously acquired tactile and proprioceptive sensor data were input to an FPGA and classified by multinomial logistic regression (MLR) and decision tree (DT) algorithms. Calibrated force and angle values and their derivatives were used as features. Movement-type (stationary, flexion, contact, extension, release) and object-type (no object, hard object, soft object) classes were predicted as discrete events. Training of the models was performed in MATLAB offline; model parameters were implemented in the FPGA by using NI LabVIEW and FPGA module. Vibrotactile feedback stimuli were generated in the FPGA card according to real-time classification. FPGA outputs were sent to custom-made power amplifiers to drive two actuators (Haptuator) placed on both upper arms of participants. The classes were mapped to discrete prosthesis events by using two frequencies and two magnitudes (relative to each participant). Six able-bodied humans participated in psychophysical experiments for measuring absolute detection thresholds and sequential pattern recognition of vibrotactile feedback. DT performed better than MLR for both object-type (97% vs. 94%) and movement-type (88% vs. 59%) classification in real time. Furthermore, the participants could distinguish vibrotactile feedback signals associated resulting from discrete events with medium recall (0.38 ± 0.08), precision (0.38 ± 0.09), similar to offline identification in our previous work. The presented thesis shows that FPGA implementation of machine learning for vibrotactile feedback is feasible in prostheses. It is expected that human performance for utilizing the feedback may increase during daily use because of additional sensory cues and physical context.

Keywords: FPGA, Somatosensory Feedback, Vibrotactile, Touch, Tactile Sensor, Proprioceptive Sensor, Decision Tree, Multinomial Logistic Regression, Machine Learning, Discrete Event-Driven Sensory Feedback Control.

ÖZET

PROTEZLERDE TİTREŞİM UYARANLI GERİ BİLDİRİM SAĞLAMAK İÇİN MAKİNE ÖĞRENMESİ ALGORİTMALARININ FPGA KARTINDA ÇALIŞTIRILMASI

Bu çalışmada, alanda programlanabilir bir kapı dizisinde (FPGA), dokunsal ve propriyoseptif verilerin makine öğrenme algoritmalarıyla, gerçek zamanlı sınıflandırılması ve protezler için ayrık olay güdümlü titreşimsel geri bildirimi uygulanması amaçlanmıştır. FPGA'ya önceden toplanmış sensör verileri girilerek, multinom lojistik regresyon (MLR) ve karar ağacı algoritmaları (DT) ile, hareket ve nesne tipine göre sınıflandırma yapılmıştır. Öznitelik olarak kalibre edilmiş sensör değerleri ve türevleri kullanılmıştır. Hareket tipi (hareketsiz durum, fleksiyon, temas, ektansiyon, ayrılma) ve nesne tipi (nesnesiz durum, sert nesne, yumuşak nesne) sınıfları ayrık olaylar olarak tahmin edilmiştir. Modeller MATLAB'da eğitilmiştir, model parametreleri NI LabVIEW ve FPGA modülü kullanılarak FPGA'ya girilmiştir. Gerçek zamanlı sınıflandırma sonuçlarına göre FPGA kartında titreşişimsel geri bildirim işaretleri oluşturulmuştur. FPGA çıkışları, katılımcıların her iki koluna yerleştirilmiş iki aktüatörü sürmek için güç amplifikatörlerine gönderilmiştir. Nesne ve hareket tipi sınıfları iki frekans ve iki genlik değeri (katılımcıya göre bağıl olarak) kullanarak kodlanmıştır. Altı sağlıklı insan, psikofiziksel deneylere katılmış ve mutlak algılama eşikleri ile sıralı titreşimsel geri bildirim verilen örüntü tanıma performansı ölçülmüştür. DT, hem nesne tipini (%97'ye karşı %94) hem de hareket tipini (%88'e karşı %59) MLR'den daha iyi sınıflandırmıştır. Ayrıca, katılımcılar hareket ve nesne tiplerini temsil eden sıralı olaylarla ilişkili titreşimsel geri bildirim işaretlerini orta düzeyde duyarlılıkla (0.38 ± 0.08) ve kesinlikle (0.38 ± 0.09) ayırt edebildiler. Sunulan tezde, protezler için FPGA tabanlı makine öğrenmesi kullanılarak titreşimsel geri bildirimin uygulanabilir olduğu gösterilmiştir.

Anahtar Sözcükler: FPGA, Somatosensoryel Geri Bildirim, Vibrotaktil, Dokunma, Dokunsal Sensör, Proprioseptif Sensör, Karar Ağacı, Multinom Lojistik Regresyon, Makine Öğrenimi, Ayrık Olay Güdümlü Duyusal Geri Bildirim.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

F	Frequency
M	Magnitude
t	Time
dB	Decibel
kS/s	Kilosamples per second
MS/s	Megasamples per second

LIST OF ABBREVIATIONS

MLR	Multinomial logistic regression
DT	Decision tree
FPGA	Field-programmable gate array
EMG	Electromyography
sEMG	Surface electromyography
DESC	Discrete event-driven sensory feedback control
DRG	Dorsal root ganglion
RA1	Rapidly adapting type 1
RA2	Rapidly adapting type 2
SA1	Slowly adapting type 1
SA2	Slowly adapting type 2
ML	Machine learning
HDL	Hardware definition language
VHDL	Very high-speed integrated circuit hardware description language
DAQ	Data acquisition card
CART	Classification and regression trees
MCP	Metacarpophalangeal
PIP	Proximal interphalangeal
DIP	Distal interphalangeal
SGL	Single-precision floating-point
LUT	Look-up table

1. INTRODUCTION

1.1 Motivation and Aim

The need for tactile feedback is increasing day by day due to the inefficiency of prostheses caused by the lack of sensory information. This inefficiency leads to the rejection of prosthesis use [14, 15]. Quality of life and self-esteem is significantly reduced due to physical and psychosocial deterioration, especially in upper extremity losses. Thus, patients may feel powerless and dependent. Commonly the upper limb losses are the results of traumatic events. Therefore, the patients mostly have no experience living without a limb nor prepared for it [16]. Especially the hand is an essential part of a person's life because of its use in daily activities on complex sensorimotor tasks. However, without sensory feedback, the patients will mostly rely on visual cues, which will increase the cognitive load and adjusting the grasping force will take a longer time to learn [16, 17]. Using prostheses without sensory feedback causes patients to experience it as a foreign body. Therefore, with sensory feedback, patients can accept the prosthesis by feeling it as a part of the body [18, 19]. In addition, the phantom limb pain is reduced with sensory feedback so that discomfort for the amputees are reduced and functionality is increased in daily tasks [20]

Although there are several sensory feedback techniques, DESC based vibrotactile feedback is shown as an effective method in terms of improving the performance of the user [12, 21, 22, 23]. Cipriani et al. (2014) used this feedback for the first time and showed that this feedback improves the performance [22]. Then they showed that DESC based vibrotactile feedback reduces slip [23]. Previously, Karakuş and Güçlü (2020) used psychophysical characterization procedure specific to the each participant and conveyed object and movement type events effectively [12].

In this study, the main goal is to implement machine learning algorithms (specifically MLR and DT) in an FPGA and generate discrete event-driven vibrotactile feedback based on tactile and proprioceptive sensor data in real-time. Without any somatosensory feedback the user would have to rely on visual cues and it is known that

visual cues cause 250 ms latency. However, latency of a tactile cue is 150 ms in average [24]. Therefore using prostheses without somatosensory feedback increases the latency and cognitive burden. So that the user's motor response could not be fast enough to securely control the prosthesis and grab objects [16, 17]. We aimed to reduce latency in somatosensory feedback, and hence improve prostheses' performance. The novelty of this thesis is also the mapping between classified sensor data to the discrete vibrotactile events. We used the sensor data that were previously acquired in our lab and applied the novel mapping method [25]. The data consisted of force and bend sensor values and they were input to the FPGA card by using another computer outputting the data through a different card. Using FPGA card, we calibrated the sensor data to convert voltage values to angle values for bend sensor and gram force for force sensor. The calibrated data were saved to the computer and used for training. The features included the calibrated sensor data and their derivatives. MATLAB was used for training the MLR and Python was used for training DT classification. After training the parameters of both algorithms were written to the FPGA. FPGA classify sensor data in real time and generated vibrotactile stimuli using DESC policy. According to our knowledge this approach is the first in the literature and it was tested on six able-bodied participants. Psychophysical results from real time classification and feedback were compared to previous study [12]. The main difference in our study to the previous study [12] is, they did not use real time classification of the sensor data. For the experiments they presented the vibrotactile feedback by generating the stimuli from a computer without using the sensor data. However, in our study, we used the FPGA to classify tactile and proprioceptive sensor data in real time and generated the stimuli accordingly. Although there are real-time applications of vibrotactile feedback, they used the feedback for posture correction or helping to teach musical instruments efficiently especially the violin [26, 27, 28]. This thesis shows that it is feasible to apply DESC based vibrotactile feedback by real time classification of sensor data. It is hoped that this approach can be used in future prosthetic applications.

1.2 Outline

In this chapter, motivation and aim which includes our novelty and contribution are given. In Chapter 2, the background information about somatosensory feedback, prostheses, the somatosensory system, the FPGA and side components to communicate with the environment are presented. In Chapter 3, the materials and methods that was used in this thesis including the experimental setup, classifying and calibrating the data, the machine learning methods that are used, experimental procedures are presented. In Chapter 4, the results of both classification and psychophysical experiments are presented. Finally, Chapter 5 presents, discussion including previous studies, technical limitations and some other issues, future work, and a conclusion.

2. BACKGROUND

2.1 Short Review of Somatosensory Feedback and Prostheses

A typical hand can perform highly coordinated precise movements and can grasp objects with strength with 27 degrees of freedom and nearly 17000 mechanoreceptors in the skin [29, 30]. The use of the prosthetic arm or hand helps the patient to restore the lost limb partially. Accident, injury, congenital defect and illness can be the reason for loosing limb [31]. A prosthetic hand or arm are created in different shapes, sizes, and designs. They try to imitate the attachment of the limb to a joint or socket by using shafts, sockets or by using cables. Upper limb prostheses can be categorized as passive and active prostheses. Passive prostheses are divided into cosmetic prostheses and functional prostheses in the market. Cosmetic prostheses come with no functionality and they are made to give the limb a natural look. They are made from silicone and to match the amputee's skin tone, hairs or other features, they are painted by an artist. Having a cosmetic device can provide psychosocial benefits to the patients, especially in traumatic events, by helping them feel more whole and it can improve their confidence. On the other hand, functional prostheses can restore partially the use of the hand by helping the patient in specific activities such as in work or sport. All parts that make up the prosthesis are assembled in a detachable way. In case of any problem or damage, prosthetic parts can be easily replaced. The hooks or fingers are movable by the person, they can grip by opening and closing manually. It can perform functions of elbow bending and locking at a certain angle. The advantages of a mechanical prosthesis are that it has very simple mechanics and it is relatively inexpensive [32, 33].

Active prostheses are divided into body-powered and externally powered prostheses. The body-powered prosthetics are operated with straps that commonly pass over the amputees' shoulders and they move with the person's own mechanical power. By moving shoulder or elbow person can open and close the hand. Using this type of prostheses are hard to get used to therefore, they require training period and lots of

repetition. Body-powered prosthesis are usually used for heavy jobs and farm work. The advantages of the body-powered prostheses are, they can be used in various mechanical environmental conditions, they require less maintenance, they can achieve high performance. The disadvantages are that the control of the prosthesis requires unnatural body movements and a lot of time to get used to the prosthesis which can be a reason to abandon it [34, 35]). Externally powered prostheses use external power source such as battery. The common externally powered prostheses are myoelectric prostheses. A custom fabricated socket is used to connect the myoelectric prosthesis to the residual limb. They are controlled through electromyographic (EMG) signals by using one or more sensors fabricated into the prosthetic socket. Whenever a person intentionally engages specific muscles in residual limb the surface EMG is captured and the signals amplitude is compared to a threshold. By doing so intended movements can be recorded and mapped to a specific event. Then if that specific movement is captured during use, the controller sends command to the myoelectric hand to drive motors and eventually move the joints. Usually, surface electrodes are placed on the skin to get the surface electromyography (sEMG) signal[36, 37, 38]. However, this signal can be very noisy and unstable if it does not have a good skin contact. Also, the electrodes may cause the skin to sweat and disrupt the signal [39]. To overcome this problem invasively implantable electrodes are being used [40, 41]. Compared to a traditional body-powered prosthesis, myoelectric arms can achieve greater comfort, larger functional area, increased range of motion, natural appearance. However, it may cost and weight more than a body-powered prosthesis [42]. Some examples of the commercially available hand prostheses are; The BeBionic v3 made by RSL Steeper, The i-Limb Quantum by Touch Bionics, Michelangelo Hand by Ottobock, DEKA Arm RC by DARPA. Although these prostheses facilitate the life of an amputee and provide several grip types with fully controllable fingers they suffer from lack of sensory feedback. Only Evolution 2 from Vincent Systems is equipped with vibrotactile feedback on the hand.

To apply sensory feedback to the patients, invasive and non-invasive methods are being used. The invasive method uses surgically implanted electrodes to apply electric currents to the central or peripheral nervous system to provoke different tactile sensations [43, 44, 45]. However, the non-invasive method does not require surgery.

Non-invasive feedback is used to stimulate the skin surface by using electrotactile, mechanotactile and vibrotactile feedback. Electrotactile feedback is provided using surface electrodes and stimulators. It is applied as low-level current pulses applied to the skin which travels through the sub-dermal area and stimulates the skin afferents. If the nerve bundles are near the contact point the sensation may spread further [46]. By changing the stimulation parameters, such as amplitude, frequency etc., the sensation intensity and quality can be modulated. In this method the sensation is more natural than the other methods, used stimulator can be compact, power consumption is low and no mechanical components are required. Electrode's location, shape or material may affect the feedback as well as the features of the current, such as frequency, amplitude and duration, and the anatomy of the person. Some deformations caused by burns, electric shock or cuts, even the moisture below the electrode, can change the perception and decrease the performance of the feedback. It can produce sensations such as vibration, pressure, tingling, However, it may also cause pain or fatigue [47, 48]. There is a possibility of the interference with the EMG signals which are used to control the prosthesis [49].

Mechanotactile feedback is used to convey pressure stimulation on body sites to represent the pressure sensed from the sensors on the prosthesis. These stimulations are given by a device named tactor. They are often used to detect the sensory input, detected from the force sensors on the finger of a prosthetic hand. Modality matched sensation is the result of this type of feedback and it improves the feeling of body ownership and lower cognitive burden then others. However, in terms of power consumption and noise level, vibrotactile feedback is better than mechanotactile feedback. In addition, the system's response is slow and require bulky indenters or motors [50, 15]. One example for the mechanotactile can be the system that was proposed by Antfolk et al. [51]. They used silicone pads on the amputation stump connected to the pads on the prosthetic hand, which were expanded whenever corresponding pads were touched on the prosthesis. They conducted experiments on 20 healthy and 12 amputee participants. The results showed that the accuracies were very high in discriminating two levels of pressure and locating the sites of the touch correctly.

Vibrotactile stimulation uses small actuators to vibrate the skin at frequencies between 10 to 500 Hz, usually around 250 Hz because humans are most sensitive at

this frequency [52, 53]. To convey various information, amplitude, frequency, pulse width, shape of the stimulus can be modulated [54, 12]. We used vibrotactile feedback in this study because the integration procedure is not complicated and small portable actuators can be used to convey the feedback, which is easy to use and power-efficient , they are compatible with EMG control and their acceptance rate is higher [52, 55, 56]. Gonzelman et al. first proposed the vibrotactile feedback in prostheses [57]. Then it was widely used due to its advantages. Pylatiuk et al. [58] used vibrotactile feedback to compare the grasp force with and without the vibrotactile feedback. Five transradial amputees which were using myoelectric hands regularly were participated in the experiments. They were asked to grasp and lift a hand dynamometer and hold it. The dynamometer was randomly attached to four different weights. Results showed that the feedback reduced the grasping force significantly. Chatterjee et al. [59] used a prosthesis with angle and force sensors mounted on it. They compared the grasping force by using three grasping force levels. The participants were required to grasp the object and squeeze the object to reach the desired level and hold on that level for ten seconds. Results showed that at all force levels vibrotactile feedback was useful to improve the performance. However, it was also noted that training might be needed to use vibrotactile feedback efficiently. Saunders et al. [60] used I-limb Pulse prosthetic hand by Touch Bionics and they put it on the able-bodied participants' dominant hand. Participants controlled the hand with force-sensing resistors placed on their fingertips. Vibrotactile feedback were applied to the participant's forearm. The results showed that the best result was achieved when both visual and vibrotactile feedbacks were present. Witteveen et al. [61] conducted virtual hand experiments with seven amputees and ten non-amputees. The virtual hand was controlled by a computer mouse for opening and closing. Four different objects having different stiffness level were tested. The results were significantly higher when the opening hand feedback and grasping force feedback were applied than no feedback. In addition, they did not find any difference between amputee and non-amputee subjects. In these systems they applied the continuous vibrotactile feedback by modulating amplitude, frequency or pulse width, however, this system increases the processing and cognitive load and forces the user to continuously focus on the stimuli for changes to identify the events [62, 63, 64, 58, 59, 51, 61, 21]. Instead of giving continuous signals, Cipriani et al. [22]

attempted to look for the changes in events such as contact, lift-off, etc. by using the discrete event-driven sensory feedback control (DESC) policy for the first time. DESC is a neuroscientific theory that was shown to be a better and effective solution. Johansson and Flanagan [65] stated that, in terms of the sensorimotor control, because the object manipulation is dealing with interactions of human body with their environment and different sequential movement stages that create the manipulation tasks, it is an interesting model system. In other words, this model states that in humans, certain coordinated muscle activity and sensory encoded discrete events, characterize the organization motor tasks such as object manipulation. They proposed that, the brain looks for specific events that specify transitions between sequential manipulative task phases, that comprises the crucial control events. During these events brain makes predictions about sensory information which makes them to serve as control points. Then brain applies appropriate control signals from these events. Therefore, for the same task, the continuous stimulation can be replaced by time-discrete signals. With this method the user will not have to continuously focus on the signal and this will decrease the cognitive load as well as processing load [65, 66, 67].

Cipriani et al. [22] used DESC policy to convey vibrotactile feedback for simple grasping tasks. They used a robotic hand named Smart-Hand (Cipriani et al. 2011; Prensilia Srl, Italy). which only the index finger and the thumb were allowed for flexion and extension. Nine healthy participants controlled the robotic hand by using their own index and thumb fingers. They were asked to first grip and lift the test object then they were supposed to replace and release it. The feedback was applied in short durations to the fingers of the participants whenever contact, release, lift-off and replace events occurred. They showed that this policy can be used as an effective method for vibrotactile feedback. Later, they used the virtual egg test by using a DESC glove, which was composed of vibrating motors and sensorized digit thimbles. Five amputees were joined the experiments and they used the equipment at their home for one month. The participants performed grasp and lift task on fragile boxes called Virtual Eggs. They were asked to grasp the boxes and carry to the other side of the wall without breaking them. The results were promising in terms of performance improvement [21]. They also showed that discrete feedback reduces slip in another study by comparing the visual, discrete and continous fedback [23].

However, in these studies, the stimulus duration was short and vibration intensity, frequency and duration were adjusted manually. In addition, the magnitudes were had to be adjusted according to the position on the arm and magnitudes were needed to be scaled, in different frequencies. Recent study used psychophysical characterization procedure specific to the each participant. Thus, for various conditions, they balanced the sensation magnitudes. They also used two actuators each placed on one arm, two magnitudes and two frequencies, to transmit more information to the user, which made possible to convey object and movement type events effectively. They found medium performance for recognizing the events, however, these results are sufficient for the prostheses. Because, for example the user already knows the hand will continue closing in sequential applications [12].

2.2 The Somatosensory System

All animal species need to have sufficient knowledge of both the outside world and their body condition. It is the function of the somatosensory system that provides this information. Animals continue their activities depending on the signals arising from the activity of the receptors reaching the central nervous system. Thermoreception, touch, propriception and nociception are the sub-modalities of the somatosensation. Information of the body's posture and body movements are sent from receptors located in skin, skeletal muscle and joint capsule which refers to proprioception. Lack of sensory feedback from proprioceptors will often cause awkward, poorly coordinated movements and without visual guidance, the person will have a hard time adapting to the complex tasks. The sense of touch is person's direct interaction with the environment and it is important factor in guiding one's behavior. For object identification the sense of touch is important and contact, pressure, stroking, motion and vibration are some examples of the touch. Thermoreception is the sense of heat and cold. In terms of maintaining body's homeostasis, thermoreceptor is an important factor. They provide information on object's heat that touches the skin. Nociception is the sense of pain what responses when the body is either harmed or damaged from external events. They are useful to protect the injured tissues by constantly warning by the sense of

pain [1].

Somatosensory perception begins from the activation of primary sensory neurons. These neurons' cell bodies are positioned in the dorsal root ganglia (DRG). The DRG neurons relay the sensory information to the brain (Figure 1.1). The peripheral branches of dorsal root ganglia cells contain special endings that transmit mechanical change information, called mechanoreceptors, distributed throughout the body [68]. The mechanoreceptors that enable the skin to function as sensory organs are located in the dermis layer and these receptors are sensitive to mechanical changes as pressure, touch, stretching, and motion. With the help of these receptors, the skin is referred to as one of our five sensory organs. Receptors that perceive the sense of touch differ in their function and structure and therefore each has been given different names. There are four types of mechanoreceptors in human glabrous skin: Meissner corpuscles, Pacinian Corpuscles, Merkel disks and Ruffini endings [1].

Meissner corpuscles are associated with the rapidly adapting type 1 (RA1)

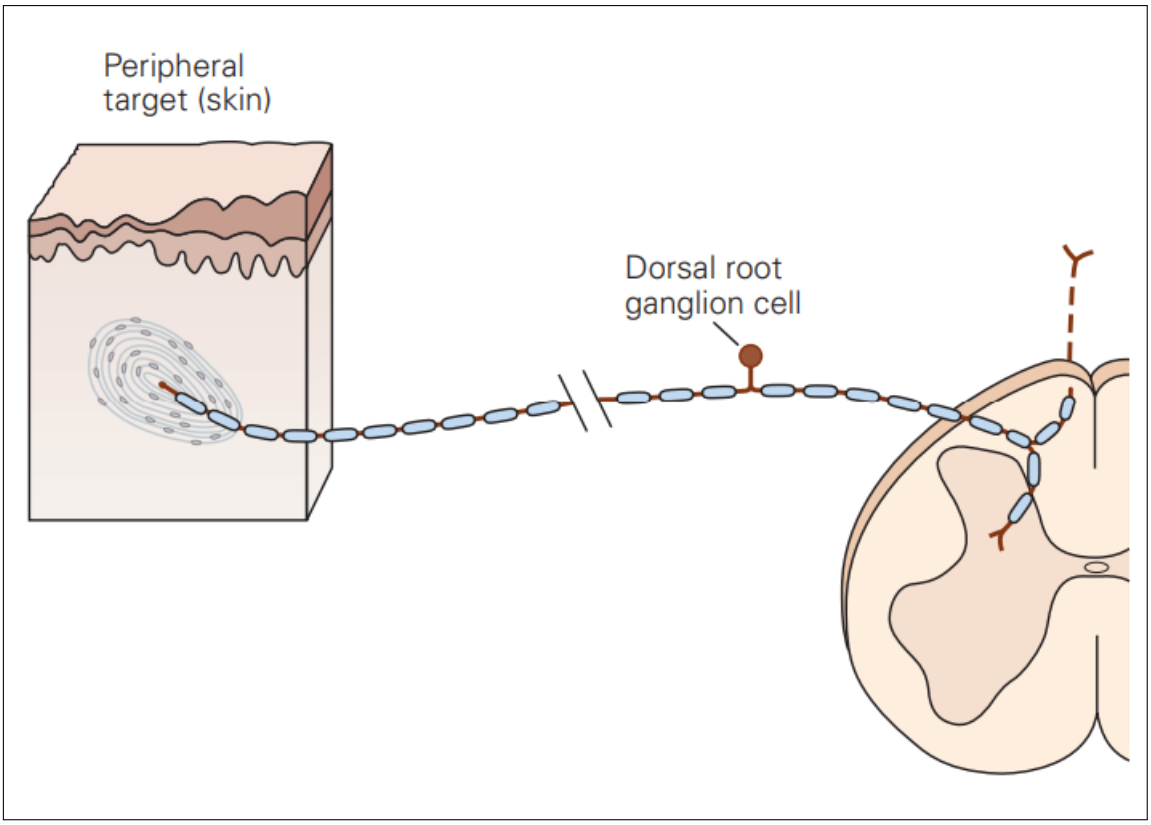

Figure 2.1 Dorsal root ganglion and nerve endings [1].

fibers. They mostly sense the low-frequency vibrations. They can detect hand motion

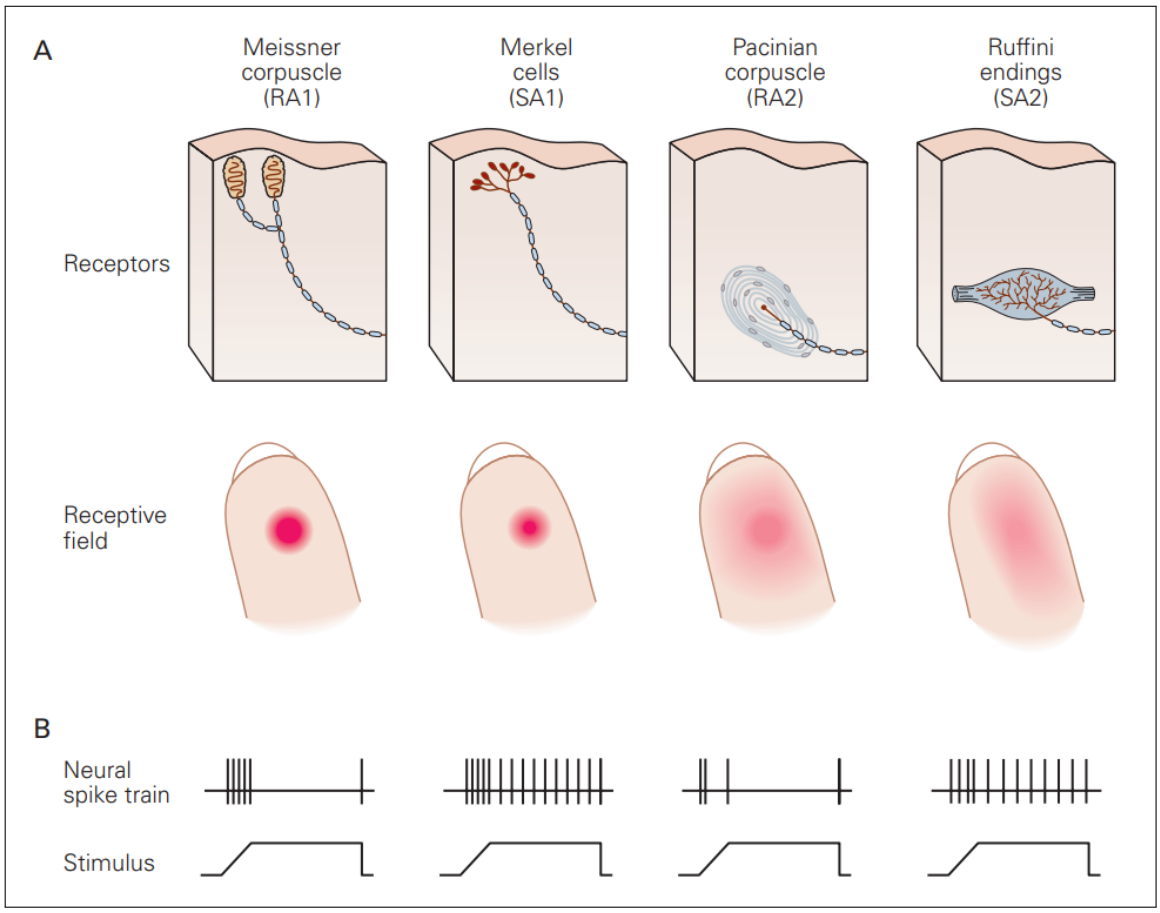

Figure 2.2 (A) The location of each receptor and their receptive fields. (B) The responses of the receptors when activated under constant pressure [1].

over textured surfaces. They can also detect object contact and slippage. Meissner corpuscles are relatively large clusters of cells located in dermal projections just below the epidermis. The RA1 fibers fire the action potential whenever there is a physical deformation. The rate of action potentials will decreases fast while a static stimulus exists [1].

Pacinian corpuscles, which are associated with rapidly adapting type 2 (RA2) fibers, mostly detect the high-frequency vibrations in handheld objects, tools or probes. Similar to Meissner's corpuscles, Pacinian corpuscles are also fast adaptive receptors with encapsulated nerve endings and are located deeper in the skin's dermis. However, they are fewer than Meissner corpuscles. Pacinian corpuscles are very sensitive, they have large receptive fields on the skin's surface [69, 1].

Merkel cells are associated with slowly adapting type (SA1) fibers. They are located in the basal layer of the epidermis and they sense the corners, edges and points.

Braille alphabet can be recognized and read by Merkel cells. They can respond to the pressure over a long time.

The Ruffini endings are associated with slowly adapting type 2 (SA2) fibers. These receptors can mostly sense the shape of large handheld objects and respond to the skin's stretch. The places of the mechanoreceptors and their receptive fields are shown in Figure 1.2 (a). When constant pressure is applied, these mechanoreceptors respond, as in Figure 1.2 (b).

Unlike glabrous skin, hairy skin includes hair follicles instead of the Meissner corpuscle. Although their function is similar to the Meissner corpuscle, they sense the movements of hair located on the skin. In hairy skin, there are other types of mechanoreceptors such as C mechanoreceptors, field receptors, hair-guard receptors and hair-down receptors [2, 1]. Figure 1.3 shows the locations of the receptors and the differences between hairy skin and glabrous skin.

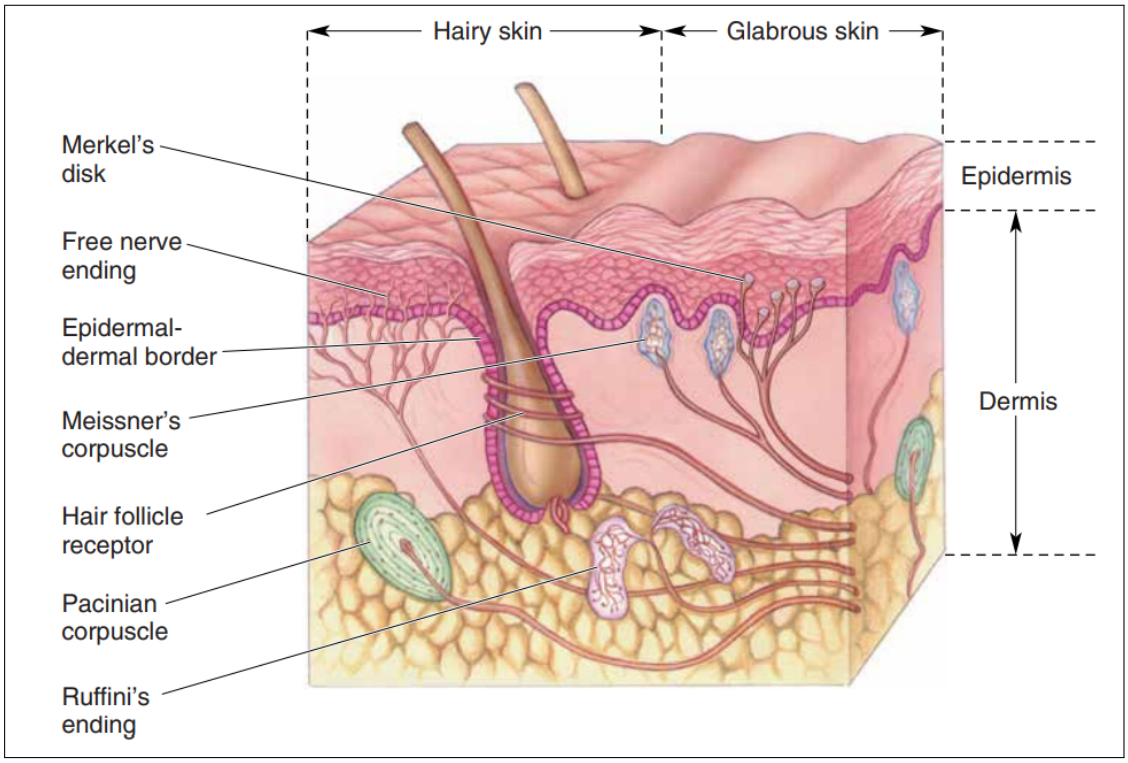

Figure 2.3 Locations of the mechanoreceptors in the glabrous and hairy skin [2].

Psychophysical channels are important in investigating sensory systems. The mechanoreceptors mediate four channels which are called Pacinian (P), Non-Pacinian(NPI, NPII, NPIII) channels in the glabrous skin. Bolanowski et al. [70] discovered these

channels by conducting experiments on detection thresholds in frequencies between 0.4 Hz to 500 Hz. They used both large contactor with 2.9 cm^2 diameter and small contactor with 0.008 cm^2 diameter. In the early stages of the perception, these channels process information and then they combine their outputs within the CNS [71]. Mostly Pacinian corpuscles mediate the P channel. This channel are most sensitive to the vibrations from 40 to 500 Hz. It follows U-shape at these frequencies and it is most sensitive at 250 Hz. Skin temperature changes affect this channel largely [72]. NPI channel is most sensitive in between 2-40 Hz. It is mostly mediated by the Meissner corpuscles. Temperature changes do not affect the NPI channel [72]. Ruffini endings mostly meditate the NPII channel and it's sensitivity region is in between 100-500 Hz which is similar with the P channel. However, in the large stimulation area, their sensitivity is much lower. The NPIII channel, which is mostly mediated with Merkel's cells, is sensitive between 0.4 Hz and 2 Hz. Figure 2.4 shows the frequency responses of the four channels.

Among all these channels only the P channel has the ability of spatial and temporal

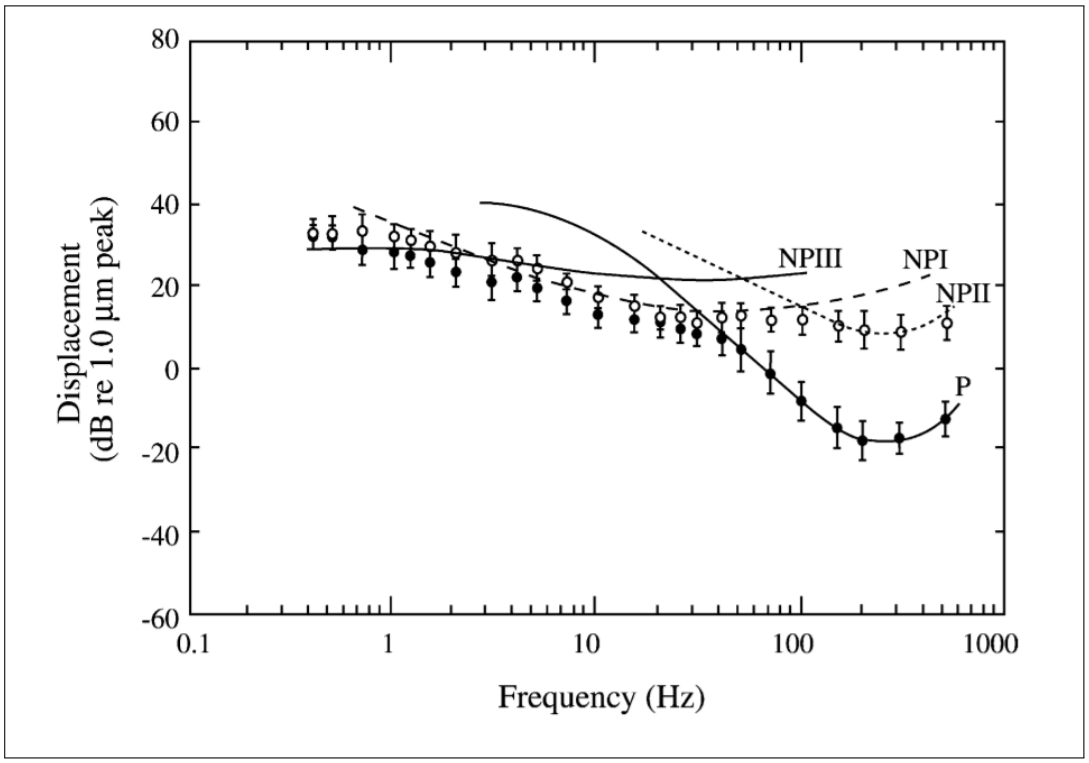

Figure 2.4 Frequency ranges of the four-channel model of mechanoreception glabrous skin [3].

summation. Having spatial summation means that when the contactor size increases the detection threshold decreases and having temporal summation means that, increase in stimulus duration decreases the detection threshold [73, 71]. However, the spatial summation effect of the P channels are negligible with the small contactor sizes [74]. The vibrotactile thresholds follow a flat shape other than U-shape for small contactor sizes. Figure 2.5 shows the thresholds of different contactor sizes in various frequencies in hairy skin. In addition, the hairy skin thresholds are higher than glabrous skin [4, 75]. The static indentation is also an important part of the tactile thresholds because the thresholds decrease when the indentation increases. This effect is shown in Figure 2.6. The temporal gap of two stimuli, spatial distance between the stimulation sites and masking effect also affect the tactile sensitivity for multiple stimuli [76, 77, 78].

To understand the relationship between stimuli (physical or chemical) and the

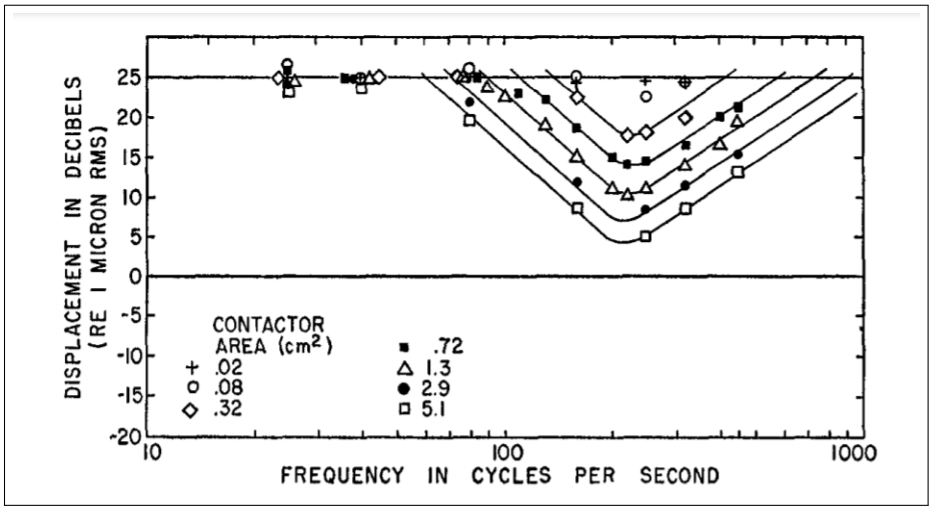

Figure 2.5 Vibrotactile thresholds of a hairy skin for different contactor areas [4].

sensory responses that was triggered by these stimuli, a scientific method called psychophysics is used. Psychophysics was established by a German physiologist, physicist and philosopher Gustav T. Fechner in the nineteenth century. Fechner described some procedures in his book to examine the relationship between sensation and stimuli by setting out psychophysical measurement principles [79]. Any sensory system can be examined by the psychophysics. The main components of psychophysics are; stimulus, task, method, analysis and measure. The stimulus must be adapted to a specific question, therefore, it is the least generic component. The stimuli are generally applied

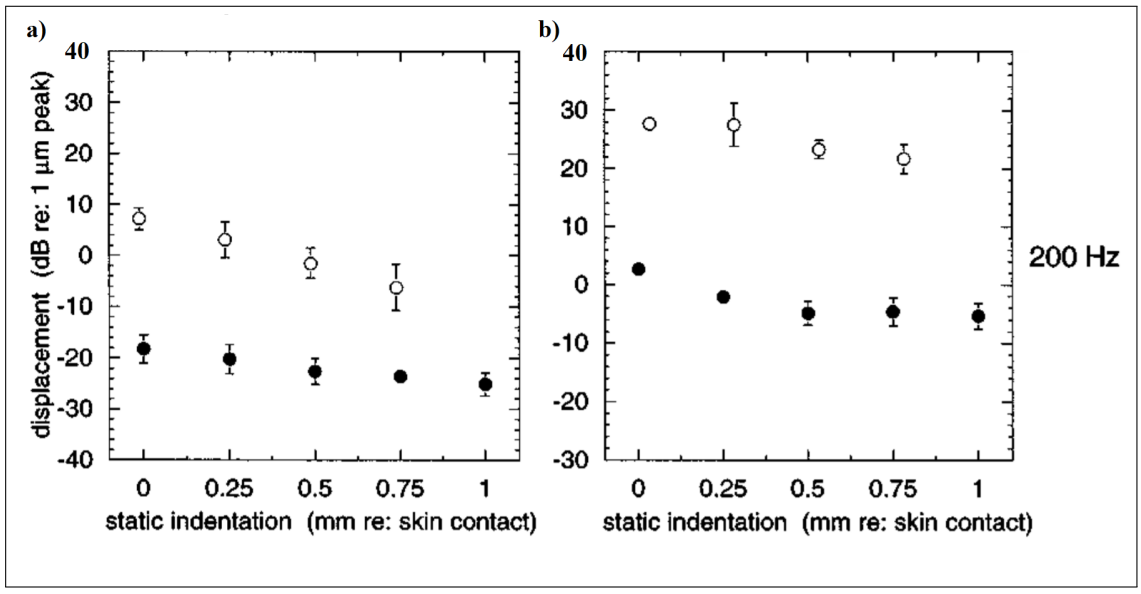

Figure 2.6 The static indentation effect on thresholds. Filled dots represent 2.9 cm^2 contactor and open dots represent 0.008 cm^2 contactor. (a) glabrous skin (b) hairy skin [5].

from a computer. The task refers to the action which the observer must perform in each trial. The analysis, explains how the data is converted in to the measurements. The analysis result is called measure [80]. Sensory threshold is the main concept of the psychophysics. The method is the way of presenting the stimuli and the way the observer's reactions are recorded. Weber and Fechner developed the classical experimental methods to measure minimum detectable sensations which is defined as absolute threshold and difference between just discriminable two thresholds which is defined as difference threshold or in other word just noticeable difference (JND) which can be used for other stimulation parameters such as frequency, duration etc. Weber and Fechner defined the relationship between JND and stimulus intensity as, the size of JND is and stimulus intensity are linearly related [73]. This relationship is called as Weber's law. Fechner showed that a logarithmic function can approximate the relationship between strength and intensity, if JNDs obeyed Weber's Law [73, 81]. Later, Stevens [1] showed that rather than logarithmic function, power function best describes the relationship between strength and intensity.

There are three main psychophysical methods; constant stimuli, methods of limits and staircase method [73, 1]. The constant stimuli method applies the predefined stimulus intensities randomly. In the method of limits, the intensity is changed

in ascending or descending order continuously. The participant reports the intensity when it is no longer detectable (descending order) or it is detectable (ascending order). Absolute threshold is found by averaging this two intensity levels. The staircase method, which is used in our study, applies a stimulus in detectable level and adjusted ascending or descending by the responses that user gives [73, 1]. In this thesis study modified three-down one-up rule is used. In this rule the correct responses decreased the stimulus level after three correct responses which are not necessarily consecutive and incorrect responses increased the level by one step [82]. These methods can be applied to the subjects using two methods. First one is the forced choice task in which the subject has to choose one of the multiple options. For instance in the two-interval forced-choice task, one of the two intervals has to be chosen by the subject for the given stimulus. The other one is the yes/no task in which the subject has to answer if they could detect the stimulus when presented. In this study we used two-interval forced-choice task with adaptive tracking (staircase) method [73].

Psychometric functions are fundamental model of the psychophysics and they

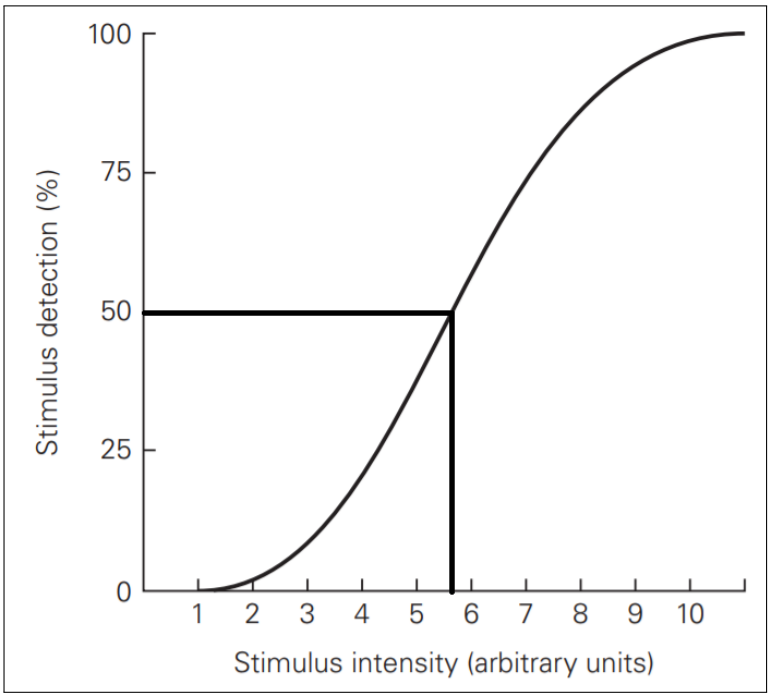

Figure 2.7 Psychometric function. Modified from [1].

relate given psychophysical task's responses to intensity level of stimulus. This function is created by fitting sigmoid function to the proportion of the responses to the inten-

sity level of the stimuli. An example of the psychometric function is given in Figure 2.7. Usually stimulus intensity where 50% probability of stimulus detection matches the stimulus intensity is chosen as the threshold. However, in this study the stimulus intensity at 75% probability of correct stimulus detection was chosen for the threshold as done in tactile psychophysics.

The gross structure of hand is shown in Figure 2.8. The hand consists, five metacarpal bones, 14 finger bones, the wrist joint and the small carpal bones in 8 bones that articulate with each other. The nerves that come out of the cervical vertebra first make the brachial plexus and extend to the hand as three more nerves. Median, radial nerve and ulnar nerve are the main nerves in hand. Bones are essential elements for stability and movement. Movable regions where bones meet each other form joints. There are cartilage covers on the faces of the bones that join with each other. The radius is located on the thumb line between the elbow and wrist, called the forearm, and ulna bone is located on the little finger line. There are five metacarpal bones, one towards the base of each finger. Finally, there are phalanx bones that make up our fingers. Two phalanges are located in the thumb and the other fingers have three phalanges each. They are named proximal phalanx, middle phalanx and distal phalanx according to their location. The bones and their joints are shown in Figure 1.5.

Muscles are contractile elements that provide movement. The end extensions where the muscles attach to the bones are called tendons. Muscles and tendons generally cause flexion, extension, abduction, adduction and rotation movements. There are separate muscle groups specialized for each movement as flexor muscles and extensor muscles. When for any reason, the nerve transmission is disabled (such as nerve cuts, compression), the movement and proprioceptive sensation caused by these nerves cannot be possible. Nerves can be motor or sensory nerves separately, or they can have mixed functions. The functions of the three main nerves (radial nerve, median nerve, ulnar nerve) carry electrical signals from the brain to the hand muscles and carry the sense of pain, heat, touch and proprioception to the brain. Two arteries, called radial artery and ulnar artery, supply blood to the hand. At the level of the palm and fingers, the branches of the radial artery and the ulnar artery first join and then give their blood supply to these areas. Veins are much more in number than arteries and their

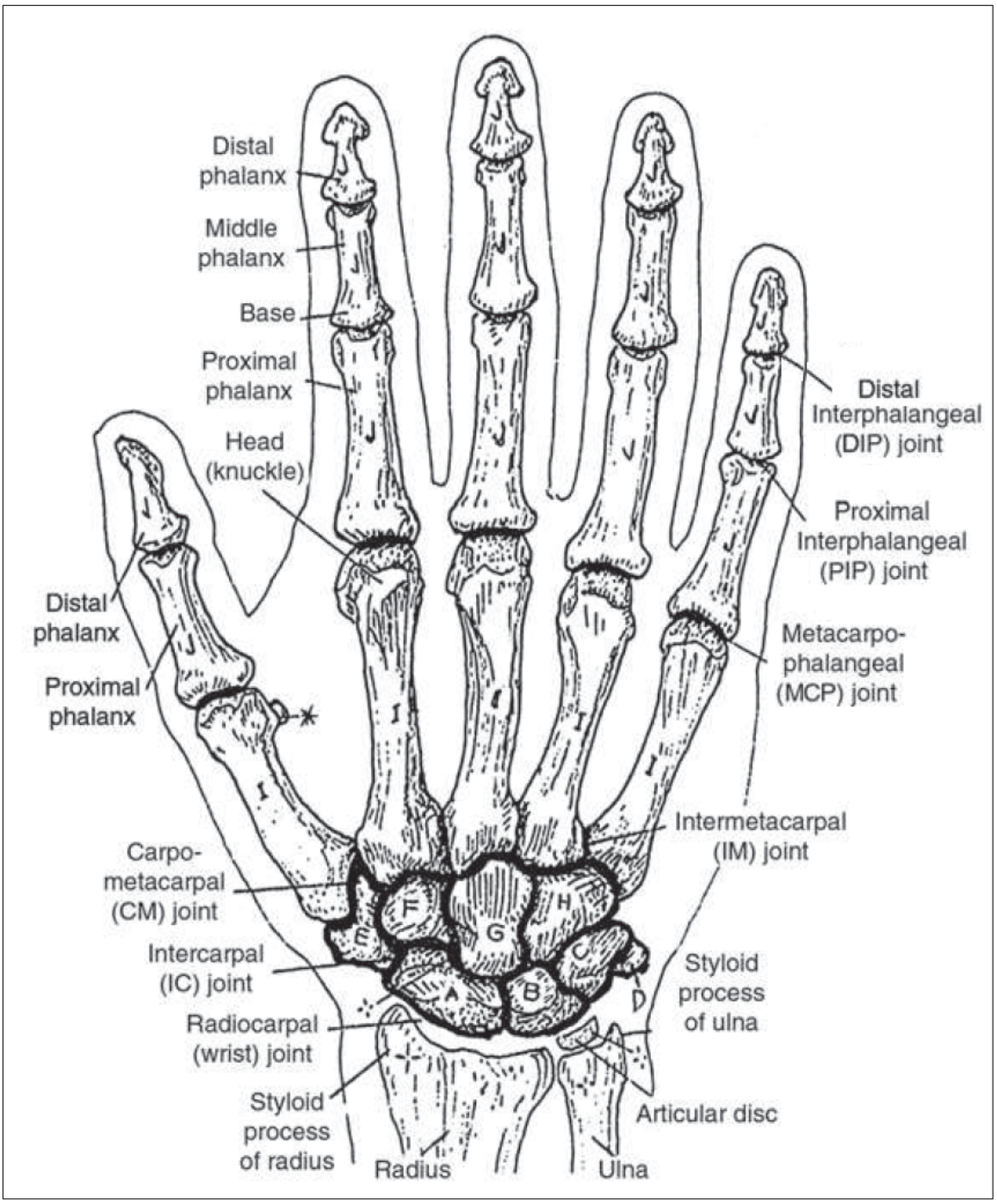

Figure 2.8 The anatomy of the human hand. Modified from [6].

anatomical location is more variable [6, 83].

2.3 Machine Learning

Learning is a process of adaptation in order to improve the results fit for the purpose. Machine learning (ML) is a field of study that deals with the question of how to obtain predictions for the future and produce solutions to possible problems by examining the past and incoming data. ML provides the development of algorithms and techniques that will enable the learning process of the computer. In other words, it is the modeling of the systems to make predictions by making inferences from the data with mathematical and statistical methods [84]. Today, there are many methodologies and algorithms for ML. It is a collection of algorithms that allows software programs to be more accurate in predicting results without explicit programming. The primary basis of machine learning is to create algorithms that can take input data and use statistical analysis to estimate an output while updating the outputs as new data emerges [7]. There are various studies in the field of health. For instance, scientists have devised models to develop systems that can detect cancer by examining cell images [85, 86, 87]. If only humans were to undertake this task, it would take much time, but thanks to machine learning, systems can accurately detect the chances of getting cancer or not without delay. All required to do this process is a machine with high computational capacity and a model with suitable algorithms. Several data mining and machine learning techniques such as classification, regression, and clustering are used to diagnose and treat diseases. Studies on hypertension, chest radiographs, prediction of blood urea concentration, scoliosis, diabetes, cardiovascular diseases, mammographic images, coronary artery disease can be seen in the literature [88, 89, 90, 91, 92, 93, 94].

In the literature, several attempts were made to control prostheses with machine learning methods. Edwards et al. [95] used machine learning to control the robotic arm in real-time. They tried the system in both amputee and non-amputee persons and they used a modified box-and-blocks task to assess the performance of the study. They used an adaptive switching method to continually reorder the joints by predicting which joint will be use next. They concluded that this method reduces process and cognitive load of the amputees in complex tasks. Swami et al. [96] produced a method to control prosthetic wrist with multi degrees of freedom by using random forest classification and regression methods. They asked ten healthy participants to

perform wrist radial/ ulnar deviation or pronation/ supination while they collect the data. Then they used this data to train the algorithm. This method allowed to control prosthetic wrist in multi degrees of freedom with high classification accuracy. Gibson et al. [97] used DT to classify EMG signals in real time into five movements (grasp, extension, flexion, pronation and pointing the index finger) to control the prosthesis. Their aim was to control the prosthesis in real time without the need of training. They placed six EMG electrodes on the arms of ten healthy participants. and they asked the participants to perform the movements for ten seconds. The results showed that 79% overall accuracy can be achieved by using this method. Suchodolski et al. [98] used DT to classify eleven movements from six electrodes placed on the skin and DT based on Neural Network Tree solution resulted in 89% accuracy and DT resulted in 85% accuracy. Wolczowski and Kurzynski [99] used two lever classifier for EMG and mechanomyography (MMG) signals to control the prosthesis. They also added sensor feedback signal from prosthesis to the classifier. They tried six types of grips and used decision tree in the classification. They compared EMG and MMG combined with only EMG or MMG. They showed that combined signals gave the highest accuracy and feedback mechanism increased the classification accuracy.

Parker et al. [100] proposed to use machine learning to provide vibrotactile feedback to the prosthetic users. They used a custom made robotic arm designed to use with non amputee arms. Five healthy participants conducted the experiments. Participants controlled the arm with a joystick. The actuators were placed with a sleeve arm. Blind folded and sound isolated participants were asked to navigate the arm inside a box from wall to wall. Reactive feedback, predictive feedback and no feedback were compared. Reactive feedback which applied vibrotactile feedback when the servo of the shoulder reached defined current load threshold. Predictive feedback applied vibrotactile feedback by predicting the electrical load on the servo motor using machine learning. They showed that feedback improved the task performance and predictive learning decreased the average load on the servo motors. Mazilu et al. [101] used machine learning to provide vibrotactile feedback when freezing of gait(FoG) is detected. The system detected the FoG using machine learning and applied vibrotactile feedback or auditory cue to warn the patient and resume walking. They used several ensemble methods and other machine learning algorithms such as kNN or naive Bayes. They

showed that the proposed system could detect the FoG with high accuracy and low latency. Huang et al. [102] proposed a model that can apply vibrotactile feedback to the amputee using the sensors on the hand glove. They used support vector machine and feedforward neural network to map the sensory information of the sensors. They created phantom map to apply feedback accordingly. In that study, SVM and FNN performed very good. However, they stated that rest of the system is under development and was not tested.

According to the learning method, ML is divided into three groups: Supervised, Unsupervised, and Reinforcement Learning. In Supervised Learning, a function is created between matching input values (labeled data) and desired output values. Training data consists of both inputs and outputs. This function can be determined by classification or regression algorithms. If the outputs in the data set are categorical, classification algorithms are used, and if outputs are numerical, regression algorithms are used. Logistic regression, multinomial logistic regression, decision tree algorithms are examples of this learning method [7].

Logistic regression is a statistical method used to analyze a dataset with one or more independent variables to determine a result. The result is measured by a binary variable; therefore, the dependent variable contains data coded as binary, i.e., only 1 (TRUE, success, etc.) or 0 (FALSE, error, etc.). The purpose of logistic regression is to find the most suitable (yet biologically plausible) model to describe the relationship between a two-way characteristic and a set of independent variables. Logistic regression generates the coefficients, standard errors and significance levels of a formula to estimate the probability of the classes. Rather than choosing parameters that minimize the sum of square root errors, estimation in logistic regression chooses parameters that maximize the probability of observing sample values [103, 104].

Multinomial logistic regression examines a nominal dependent variable with more than two categories and many independent variables. In this analysis, while the dependent variable is a multi-category (at least 3) nominal variables, independent variables can be continuous, sequential, or categorical. The multinomial logistic regression aims to estimate the probability that a particular person belongs to any category of the dependent variable. In the multinomial logistic regression, the category of the dependent variable to be referenced is selected first. MATLAB's mnrfit chooses the last

category as the reference category [105]. Several binary regressions are then performed to compare the remaining categories with the reference category [103]. For a dataset with k categories, k-1 equations must be calculated, which is for each category relative to the reference category. Therefore, Eq. 1.1 can be used to get the predicted log odds.

$$ln\big(\frac{\pi_j}{\pi_r}\big) = \beta_{j0} + \beta_{j1}x_{j1} + \beta_{j2}x_{j2} + \ldots + \beta_{jn}x_{jn} = Z_j \qquad j = 1, ...k-1, \tag{2.1}$$

Where r is the reference category, n is the number of predictor variables, π is the categorical probability. The coefficients are represented as β, which are calculated using maximum likelihood estimation. Hence, for each category, there will be one log odd, which is relative to the reference category, making k-1 log odds in total. In order to calculate probabilities for each category, Eq. 1.2 is used. This equation is also called softmax function. MLR uses softmax function as a classifier. After calculating the probabilities for each class, the class with highest probability is chosen as classification result [105, 106, 107].

$$pi_j = \frac{exp\big(Z_j\big)}{1 + \sum_{h=1}^{k-1} exp\big(Z_h\big)} \qquad j = 1, ...k-1, \tag{2.2}$$

The mnrfit command in MATLAB is used for training the MLR model. After training the mnrfit function gives coefficients of the trained model. To classify the data either mnrval function or Eq. 1.2 can be used [105].

The decision tree is either classification or regression method that creates a model in the form of a tree structure consisting of decision nodes and leaf nodes. It is used to divide a data set containing large datasets into smaller sets by applying a set of decision rules. The decision tree approach approximates the goal functions and shows the learning function in a tree structure. A decision tree is a descriptive and predictive model. This model helps the decision-maker consider the factors while making a decision and determine how each factor relates to the different outcomes of the decision in the past [108, 36]. Tree-based learning algorithms are considered to be one of the most used supervised learning methods. They are easy to interpret, they can handle nominal and numeric attributes or datasets with missing values or errors. Unlike linear models, they can also match nonlinear models quite well. However, sometimes small changes in the input data can cause large changes in the tree structure [109, 110]. Clas-

sification or regression can be adapted in solving any problem obtained. Decision trees, random forest, gradient boosting are widely used in all kinds of data science problems. Some methods combine various decision trees to predict better from a single decision tree. These are called tree ensemble algorithms. The primary purpose of ensemble algorithms is to get many weak learners to form a strong learner [7, 111, 110, 109].

There are several techniques for implementing ensemble algorithms. The first method is called bagging. Bagging is used when the variance of a decision tree is needed to be reduced. An example of this method is the random forest algorithm. Boosting is another method used to build a collection of insights. In this technique, learners are based on learning sequentially by matching simple models of early learners with data, then analyzing data for errors. Gradient boosting can be given as an example of this technique [112, 113].

A decision node can contain one or more branches. Non-split nodes are called Leaf or Terminal nodes that represent a decision or final classification. The first decision node in a tree corresponds to the best determinant called the root node. A node divided into sub-nodes is called the parent node, and the parent node's sub-nodes are called the child nodes. A decision tree can consist of both categorical and numerical data. It has decision nodes and terminal leaves to classify the data. The basic idea is that the input data is divided repeatedly into groups with the help of a clustering algorithm. The clustering process continues in depth until all nodes are pure. Learned tree models can be represented as if-then rule sets to increase human readability [7, 111].

Gini or entropy is used to measure the impurity of a node. Entropy, known as the uncertainty measure of a random variable, is the expected value of the information contained by all instances for a process [114]. The higher the entropy, the more information is obtained by the system. The entropy is calculated with Eq. 1.3 below.

$$E = -\sum_{i=1}^{n} (\log_2 \frac{ns(i)}{N}) * \frac{ns(i)}{N} \tag{2.3}$$

In this equation, n represents the number of classes, ns (i) the number of samples for each class, and N represents the total number of samples. It shows the representation value of the information gain data set after the division. The entropy value is expected to be high [7, 114].

Gini index or Gini Impurity was developed by the Italian statistician Corrado Gini in 1912 [115]. Gini is widely used in statistics. The coefficient is in the range of 0 to 1; 0 means perfect equality and 1 means perfect inequality. The Gini index is a metric used to measure how often a randomly selected item is detected incorrectly. A feature with a low Gini index should be preferred [7, 115]. The Gini index runs successfully or fails for the categorical target variable. The high Gini index increases the homogeneity. The advantage of the Gini index over entropy is, if there are more than two categories in the output variable, they perform better [110]. To calculate the Gini index, the sum of squares of each classes' probabilities is subtracted from 1, as shown in Eq. 1.4. Where c is the number of classes and p is the probabilities of each class [7, 115].The Gini index is minimum if each terminal nodes contain data from just one class and it is maximum when all classes are distributed equally in each node [116].

$$Gini = 1 - \sum_{i=1}^{c} (p_i)^2 \tag{2.4}$$

A simple decision tree structure is shown in Figure 2.9. There are applications such as classification tree or regression tree in the literature, which can be accepted as sub-methods of decision tree learning [117, 118, 119, 120, 121, 122, 123].

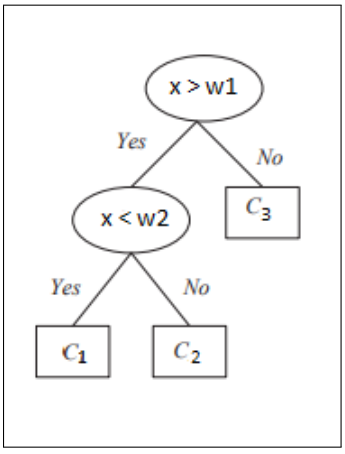

Figure 2.9 Schematic of decision tree structure. Modified from [7].

To find the best split node decision tree first finds the best Gini Impurity of all features. For the numerical data, algorithm first sorts the rows of each feature separately in ascending order. Then, the algorithm calculates the average of all adjacent values which are called candidate thresholds. After that, it calculates the Gini Impu-

rities by using Eq. 1.4 for each Leaf and then calculates the weighted average of Gini Impurities for each candidate threshold by splitting the data from that value. Lastly, the lowest value is chosen as the best split threshold for that feature. This process is repeated for each feature separately. After finding the Gini values of all features, the feature with lowest Gini index is chosen for the split [124, 116]. For the features with categorical data however, the algorithm splits the data for either one child node per class (multiway splits) or only two child nodes (binary split). CART [125] is an example of the binary split and FACT [126] algorithm is an example for the multiway split. CART (Classification and Regression Trees) uses the Gini method to create binary panes. For binary split, if the predictor is made of two classes, one split is possible. However, if the predictor has three or more classes, the algorithm splits the data by taking one class for the first child node and the rest of the classes for the other child node, which is called one-vs-rest algorithm [127]. The split with lowest Gini Impurity is chosen for that feature. Then the algorithm calculates the weighted Gini Impurities for each candidate and follows the same steps as numerical data.

2.4 FPGA Board

FPGA (Field-programmable gate array) is a programmable integrated device with hardware in which the internal structure can be changed according to the operation desired by the user [128]. There are logic gates and memory elements in the FPGAs, but it is entirely up to the user to decide which of them to connect to get the results. The user expresses the design he needs with the help of a hardware definition language (HDL) and can design the elements within the capacity of the FPGA. If the circuit does not work after completing the design or if the user wants to include a new feature or change a feature, the design can be changed and implemented again to the FPGA. Due to this advantage, many ASIC manufacturers in the market first try their designs on FPGA, and if necessary, they make changes to it and produce ASIC from the final form of the design and put it on the market. Apart from rapid prototyping, FPGA is used in many other applications such as; aerospace & defense, automotive, consumer electronics, data center, industry, medical applications, video

and image processing, machine learning applications, wired or wireless communications [8, 90, 117, 119, 121, 129, 130, 131, 132, 133, 134, 135].

FPGA ICs consist of 3 main parts: logic blocks, interconnections, and input-

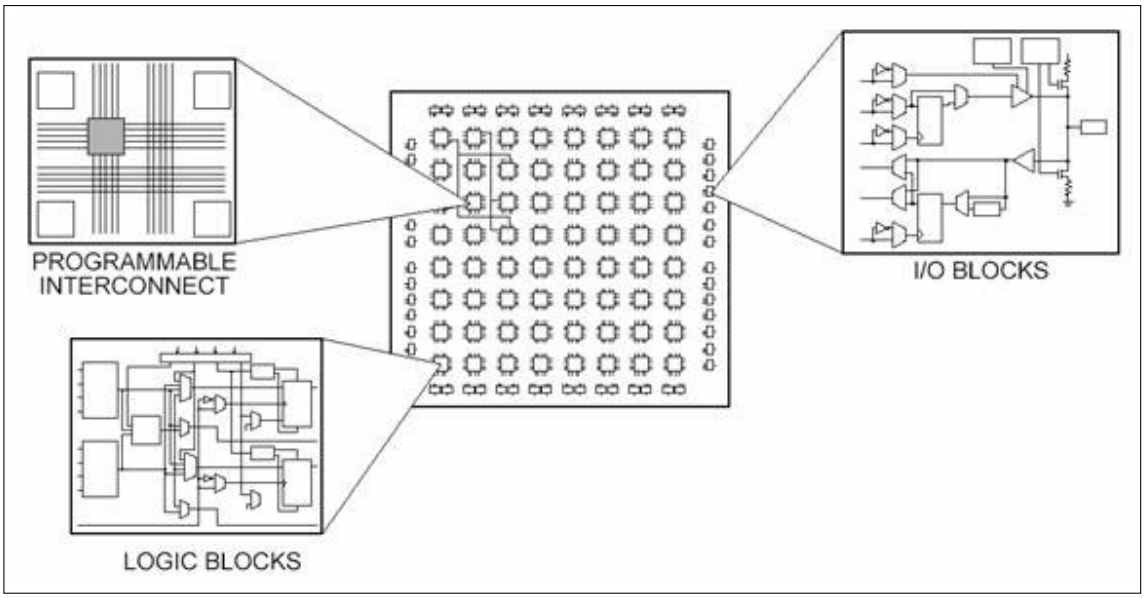

Figure 2.10 Parts of the FPGA [8].

output blocks, as shown in Figure 2.10. L.U.T.s (Look-up Table) perform logic operations in logic blocks and memory elements (Flip-Flops) to store their results. The interconnections provide the connections between the logic blocks according to the developer's needs and enable the desired function to work. Input-output ports, on the other hand, allow FPGA to transfer data by communicating with external devices such as sensors, screens, computers, etc. [136]. The FPGA has a parallel programming advantage to the other I.C.s. Also, it is reprogrammable and reusable, which makes it better than ASICs. It can be adapted to the need of the customer and the project. However, it is slightly expensive, and it consumes more power than microcontrollers [136, 137, 134]. The FPGA consist logic gates which are AND, OR, NOT, XOR and NAND gates. Gates are the fundamental blocks in FPGA. OR gate gets two input and if either of the inputs are 1 then the output will be one. The AND gate outputs one if both inputs are one, otherwise zero. The NOT gate outputs the opposite of the input. It only accepts one input. The XOR gate has two inputs and if just one of the inputs are one it outputs one, otherwise it outputs zero. The NAND gate is the exact opposite of the AND gate. XOR gates are used in binary adders. When two bits are

added, if the addition is equal or more than two the extra carry bit is shifted to the next column [136].

The languages used to determine the internal configuration of FPGA are called HDL (Hardware Description Language). The traditional languages are VHDL (Very High-Speed Integrated Circuit Hardware Description Language) and Verilog. While Verilog has a structure similar to C, VHDL is like a lower-level language. However, these languages are completely different from traditional programming languages, and they express the structure of the hardware, that is, the connection of the elements and the internal configuration of the integrated, rather than flow and operation. There are also High-Level Synthesis (HLS) Design Tools such as LabVIEW. LabVIEW is a graphical programming language to make FPGA coding easier than traditional languages, and it is more effective. Because, unlike most of the HLS code generators, rewriting the code in VHDL for satisfying the timing or resource constraints is not needed [138].

Figure 2.11 shows the FPGA board (NI 7845R) that was used in this project. This FPGA board contains cost efficient, high power Kintex-7 70T FPGA chip. Also, to communicate with the environment it includes, 8 analog input (AI) ports, 8 analog output (AO) ports and 48 digital input and output (DIO) ports. Lastly, there is a NI ASIC card to communicate with the computer by USB cable. The block diagram of the FPGA is shown in Figure 2.12 [9]. The analog inputs are connected to the analog to digital converters (ADC), which converts the analog signals to the digital signals. They are mostly used in data transmission,information processing, computing and control systems. On the other hand the output ports are connected to the digital to analog converters (DAC). DACs convert the stored or transmitted data to analog signal. They are used to display the data or to control other devices such as motors [139].

In the literature FPGA is widely used in machine learning real-time applications. Krips et al. [140] used resilient backpropagation algorithm for hand tracking in real-time. They aimed to develop a video tracking device in works to increase safety. They showed such applications can be done in real-time. Page et al. [141] used FPGA to detect seizures using several machine learning methods and compared them in terms of latency, accuracies, memory, etc. However, they used a high memory FPGA model and used algorithms that require high memory such as kNN, which is not suitable for

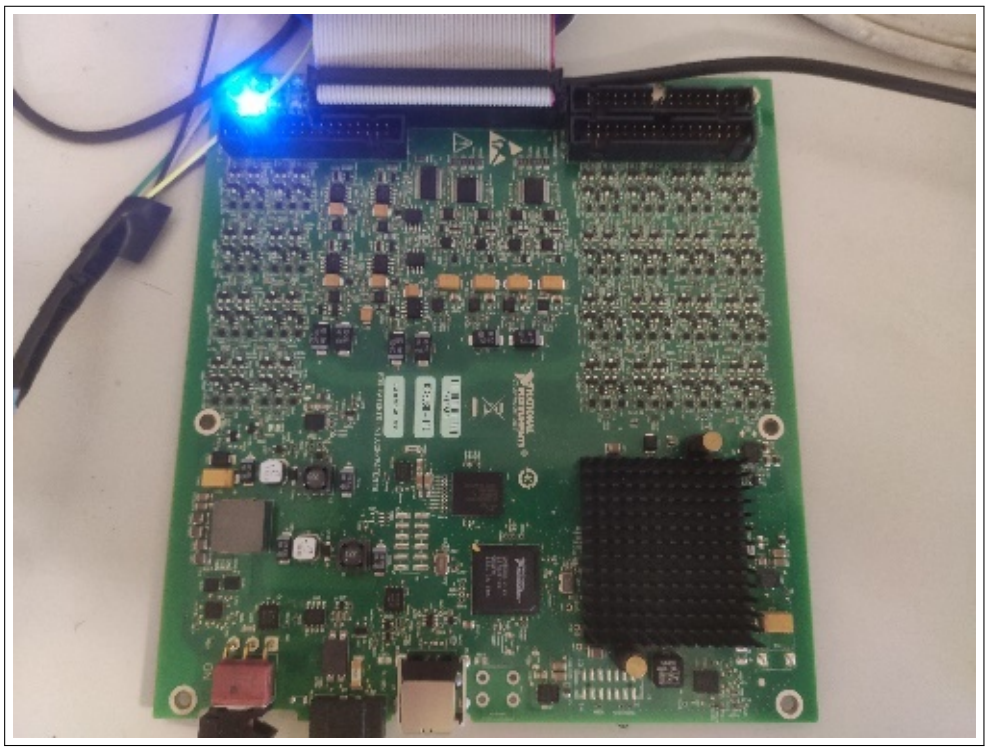

Figure 2.11 Photograph of the FPGA card (NI 7845R) that was used in this project.

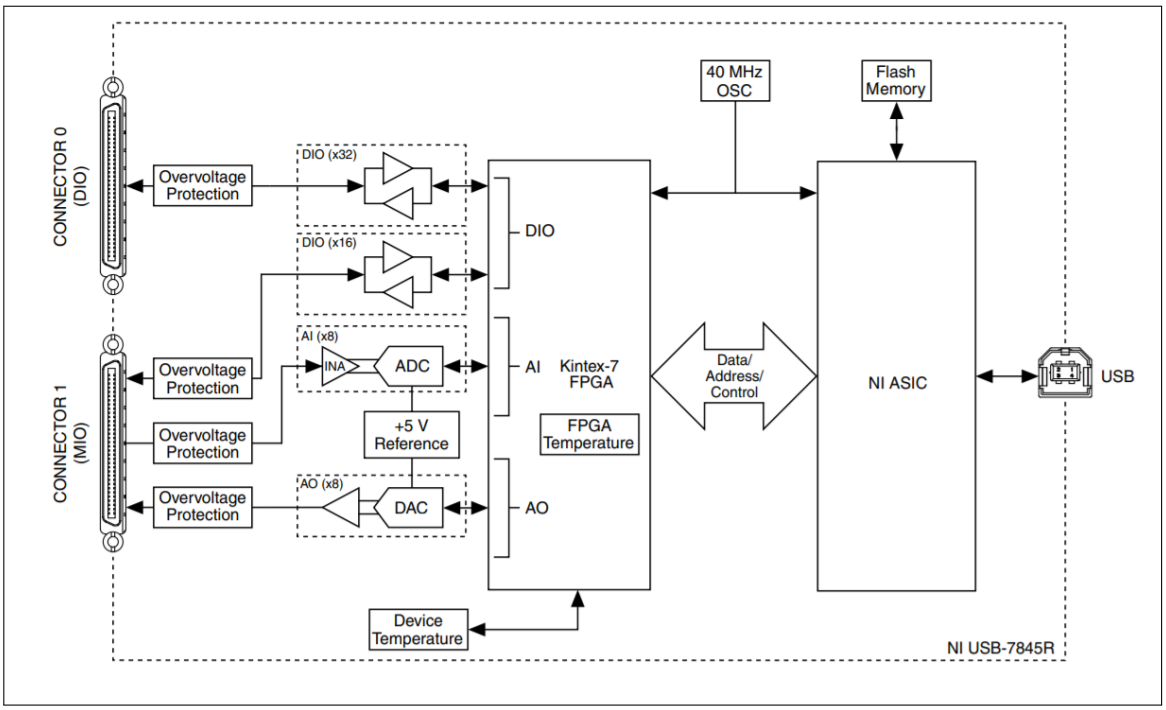

Figure 2.12 Block diagram of the FPGA card (NI 7845R) [9].

low memory applications. Boschmann et al. [142] used FPGA to improve signal processing time thus reduce latency multi-channel EMG in myoelectric prosthesis. They controlled the prosthetic hand with this classified data. They were successful to reducing latency and controlling the prosthetic hand. Fejer et al. [143] used FPGA to control prosthetic hand in real-time by analyzing the visual environment captured from a video camera. Their results show the FPGA consumes less power than processors and they are faster. Alfaro-Ponce et al. [90] used FPGA to detect electrocardiographic arrhythmias by using continous neural networks. They achieved high accuracy and sensitivity in real-time on EKG signals.

3. MATERIALS AND METHODS

3.1 Experimental Setup

We used MATLAB to control and communicate with the NI data acquisition card (DAQ) (model number NI6259). The data acquisition card was used to send the data to FPGA card (model number NI7845R). The features of both FPGA card and data acquisition card is shown in TABLE 2.1. The data acquisition card has four analog outputs and they were connected to the input ports of the FPGA card. After calibrating and classifying the data, vibrotactile feedback stimuli were generated and sent to the power amplifier accordingly. The classification results were sent back to the computer from an analog output port of the FGPA through the data acquisition card and saved for comparison. The stimuli were amplified and sent to the actuators. The actuators were placed on both upper arms of the participants. The block diagram of the setup is shown in Figure 2.1. The first actuator was placed in the left arm and used to give object-type (soft object, hard object, and no object) and force-related (low force and high force) feedbacks. The second one was placed in the right arm and used to give feedback for the movement-type events (flexion in air, extension in air, stationary in air, flexion in object, extension in object, and stationary in object).

Table 3.1
Specifications of FPGA card (NI 7845R) and DAQ card (NI 6259).

Product	Analog Inputs (16-bit)	Max Sampling Rate per Channel	Analog Outputs (16-bit)	Max Update Rate per Channel (MS/s)	Digital I/O
NI 7845R	8	500 kS/s	8	1	48
NI 6259	32	1.25 MS/s	4	1.25	48

The participants were sitting on a comfortable chair in front of a computer screen. The experiments were conducted using MATLAB. Earphones were used to block the sounds from the actuators and play white noise while conducting the ex-

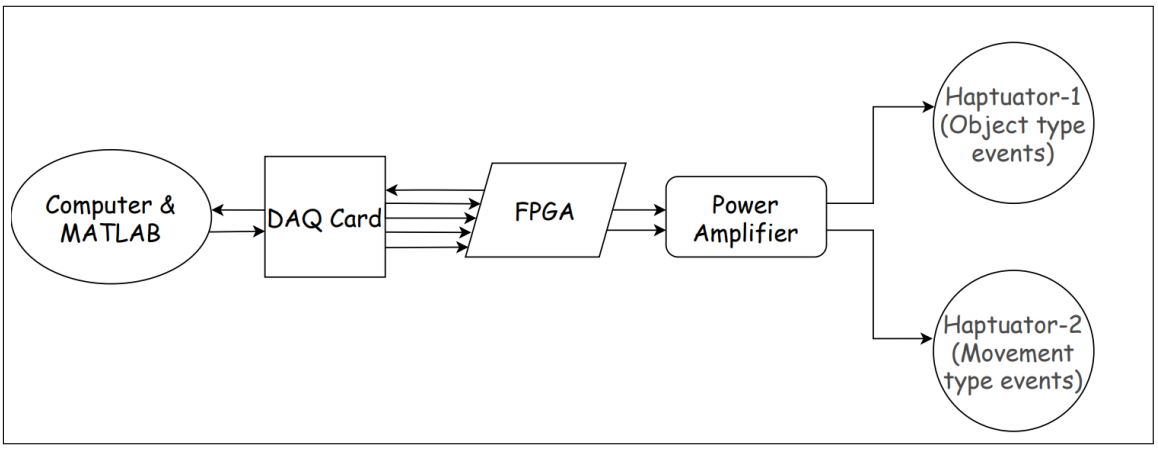

Figure 3.1 MATLAB is used for conducting the experiments and controlling the DAQ card to send the raw data to FPGA card. The FPGA card receives the raw data and after calibration and classification, it produces the vibrotactile stimuli and sends the stimuli to the power amplifier by two output channels. One output is for the object-type events, and it drives the Haptuator-1. The other output is for movement-type events and it drives the Haptuator-2. The power amplifier amplifies the signals and sends them to the actuators accordingly.

periments. A keyboard and a mouse were used to get participant's responses. An illustration of part of the experimental setup and placement of the actuators is shown in Figure 2.2. The power amplifier was located between the computer screen and the keyboard, along with its power supply. FPGA card and the data acquisition card were not shown in this illustration.

3.2 Data Calibration and Classification

In this study, we used the sensor dataset from Karakuş et al. [25, 13]. The sensor data were taken from the robotic hand in the Tactile Research Laboratory. Fourteen bend sensors and eleven force sensors were mounted on it. The bend (flex) sensor (Figure 2.3) is a sensing circuit element that changes its resistance by bending amount. Resistance varies in direct proportion to bent. Higher the bend, the greater the resistance value [10]. Bend sensors were placed on the dorsal side of metacarpophalangeal (MCP), proximal interphalangeal (PIP), and distal interphalangeal (DIP) single-precision floating-point (SGL)joints. Piezoresistive Force Sensor (Figure 2.4) detects the applied pressure and converts it into an analog signal [11]. Force sensors were mounted on the ventral side. The sensors were named by their positions on the hand.

Figure 3.2 The actuators are electromagnetic recoil-based, and a foam rubber piece was used to hold in place using adjustable straps. To block the sound coming from the actuators, participants were listening to white noise with headphones. They were sitting in front of a computer for the experiment.

The sensor outputs were converted from resistive changes to voltage changes via multi-channel op-amp circuits [13]. The hand executed simple grasping tasks with a soft object, a hard object, and without any object, and the collected data of the sensors were stored and sent to the input ports of the FPGA card by DAQ card. We used FPGA to filter and convert the data to the sensor's correct units (gram force for force sensors and degrees for bend sensors).

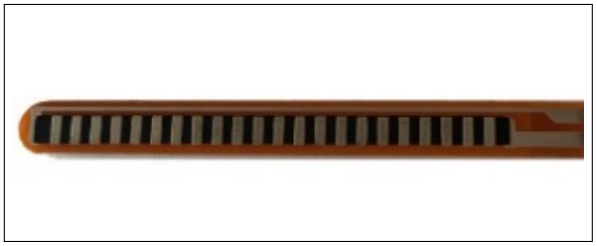

Figure 3.3 Bend sensor (FlexSensor, SpectraSymbol) [10].

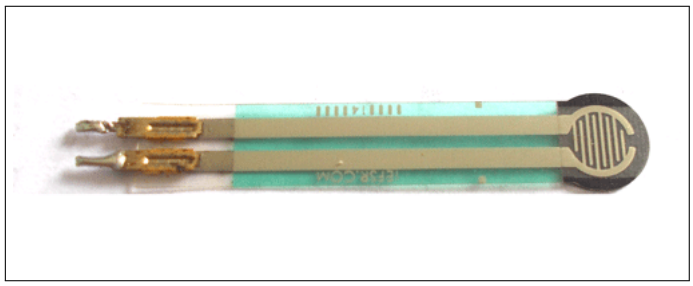

Figure 3.4 Force sensor (FSR400-Short, Interlink Electronics) [11].

3.2.1 Numerical Representation Format in FPGA

The sensor data consist of 560000 floating-point numbers. To handle floating-point numbers in FGPA, the data were converted to the single-precision floating-point (SGL) type. In order to facilitate the use and calculation of floating-point numbers and to save memory on FPGA card significantly, we used LabVIEW FPGA Floating-Point Library by National Instruments [144].

3.2.2 Data Calibration

Pre-processing the data is a crucial step of machine learning because the learning ability of the model is directly affected by the quality of the data and the information that can be derived from it [7]. Therefore, before feeding it to the decision tree classifier, the data were filtered and calibrated in the FPGA card. First, the input data were filtered with a Butterworth filter (second-order low-pass filter). The filter module in LabVIEW only accepts the numbers as fixed-point types or 32-bit integers. Hence, the data were converted to the integers, and in order to avoid the loss of the decimal numbers, the data were multiplied by 108 before filtering. Then the output of the filter was converted back to the SGL type and divided by 108.

$$y = ax^b + c \tag{3.1}$$

$$y = ax^b \tag{3.2}$$

The data were filtered and converted to correct units (gram force for force sensors and degrees for bend sensors. In Eq. 2.1 and Eq. 2.2, a, b and c are the calibration parameters shown in Table 2.2. X represents the calibrated output of the data and y represents the input which is the raw sensor data in our case.

FPGA is unable to compute exponential floating-point numbers other than the square root. In Eq. 2.1 and Eq. 2.2, the power of x with the floating-point b values from Table 2.2 needed to be calculated.

Following approximation in Eq. 2.3 was used to calculate the intended powers using only the square and square root.

For $|\alpha|<|\beta|$,

$$\frac{\alpha}{\beta} = \sum_{k=1}^{n} c_k \left(\frac{1}{2}\right)^k + R \quad c_{\mathrm{k}} = 0 \; or \; 1 \tag{3.3}$$

$|\mathrm{R}| < 0.005$ was selected because when powers are fitted there are 0.01 precision so that exponent α/β has two significant digits. Using Eq. 2.3, the intended number can be approximated. For example, to calculate the 0.70, Eq. 2.4 can be used.

$$0.70 = (\frac{1}{2}) + (\frac{1}{2})^3 + (\frac{1}{2})^4 + (\frac{1}{2})^6 - 0.003125 \tag{3.4}$$

Therefore, to calculate $\mathrm{x}^{0.70}$, Eq. 2.5 can be used.

$$x^{0.90} = \sqrt{x}.\sqrt{\sqrt{\sqrt{x}}.\sqrt{\sqrt{\sqrt{\sqrt{x}}}.\sqrt{\sqrt{\sqrt{\sqrt{\sqrt{x}}}} - 0.003125}}} \tag{3.5}$$

Then the derivatives of the calibrated data were taken. The raw and calibrated data of all sensors are shown in the Appendix (Figure a.7-a.11). The top graph shows the calibration in MATLAB for comparison with the FPGA card calibration output which is shown in the middle graph. The bottom graph shows the raw data input to the FPGA card. The derivatives of the calibrated data for each sensor are shown in the Appendix (Figure a.12 and Figure a.13). The graphs show the each sensor's derivative data. For each sensor, there were two outputs, which formed ten features in total for the multinomial logistic regression. However, for the decision tree classification,

Table 3.2
Table of calibration parameters. Modified from [13].

Sensor Type	Sensor Name	a	b	c
Bend Sensors	Ring-MCP	0.32	0.69	
	Ring-PIP	0.66	0.48	
	Ring-DIP	0.15	0.90	
Force Sensors	Ring-distal	-601.36	-1.11	8.08
	Ring-palmar	-55.34	-0.57	9.11

the MCP was eliminated. The reason for eliminating the MCP is because the data acquisition card only has four analog output ports to send the raw data. Therefore there were eight features in total and the features used for training and classification.

For training, %70 of the output data were initially transferred to MATLAB to train the data. Then the parameters of multinomial logistic regression or the decision tree structure were implemented in FPGA card for classification.

There were two different classification techniques. In object-type classification, there were three classes, no-object, soft object, and hard object. In movement-type classification, there were five classes, stationary, flexion, contact, release, and extension. Figure 2.5 shows the calibration code on FGPA.

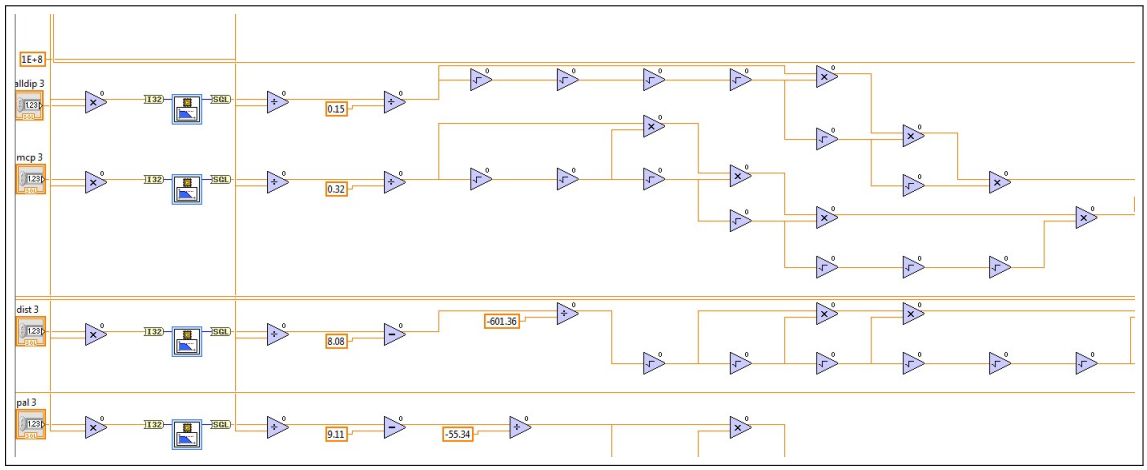

Figure 3.5 The calibration part of the code.

3.2.3 Multinomial Logistic Regression

The FPGA card has low memory capacity, and the data could not be stored on it. Therefore, the data was exported to MATLAB for training. The data contained ten features (five sensor outputs and five derivatives of each sensor and one output class for the supervised learning). The data were trained using the MATLAB function mnrfit [105].

$$\ln(\frac{\pi_j}{\pi_r}) = B_{j0} + B_{j1}X_{j1} + B_{j2}X_{j2} + \cdots + B_{jp}X_{jp} \qquad j = 1, \ldots, k = 1 \qquad (3.6)$$

To implement the model in the FPGA card, Eq. 2.6 was used. Where π_r is the reference class, which π stands for a categorical probability, the number of all classes is k, p is the number of predictor variables [145]. For example, there are three classes in object type classification, and the last class, soft object class, is the reference class. Furthermore, applying the Eq. 2.6 for all classes, using the coefficients taken from MATLAB after training the data, we had two probabilities for three classes. From these probabilities, class probabilities had to be calculated. To accomplish that, a simple decision algorithm was used to get the maximum probability, which was the result of the classification.

First, the division of the categorical probabilities (for object type classification,

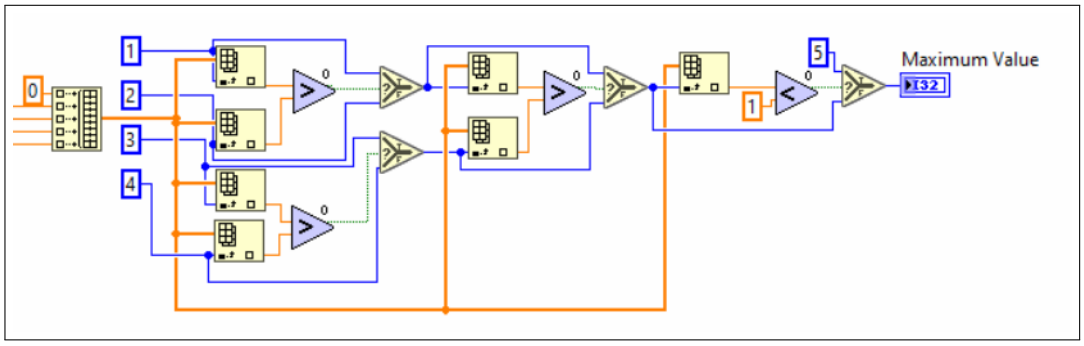

Figure 3.6 The decision steps to find the classification result of MLR for movement-type classification.

$\ln(\frac{\pi_1}{\pi_3})$ and $\ln(\frac{\pi_2}{\pi_3})$ were taken. If the reference class were higher, the result of the division would be less than one, and the reference class probability was the maximum probability. However, if the ratio were the higher one, the result would be higher than

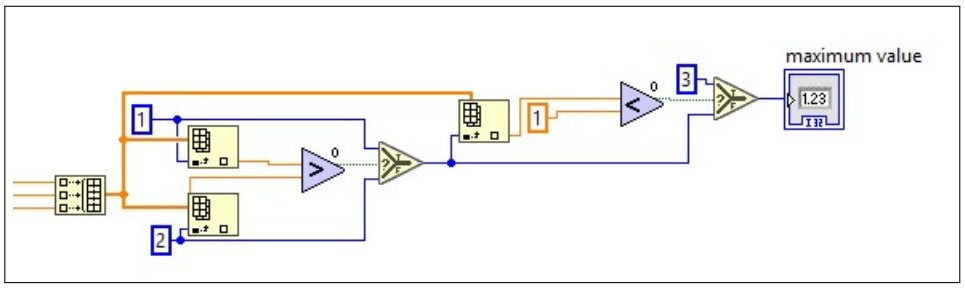

Figure 3.7 The decision steps to find the classification result of MLR for object type classification.

one. In this case, we had to compare both divisions and choose the higher one, and that would give the class which had maximum probability. Figure 2.6 and Figure 2.7 show the decision steps for movement and object type classifications, respectively.

3.2.4 Decision Tree Classification

Decision trees are simple to understand yet very effective. Decision tree models do not consume much memory, which is a plus in low memory FPGA applications. Besides, data preparation for decision tree classification is more straightforward than other machine learning methods, which often require data normalization, handling null values and some other techniques [104].

Python and the sci-kit-learn library were used for training the decision tree model [146]. After data calibration, the features were saved to the computer from the FPGA card. The data were split into the training set and testing set by 70% and 30%, respectively. For object-type events, three classes were defined as a hard object, no object and soft object. Five classes were defined as stationary, flexion, contact, release, and extension for the movement type. A split criterion was defined to stop the growth of the tree at the intended number of leaf nodes. The Gini cost function was used to measure the quality of a split, which indicates the pureness of the node.

Next, the decision tree model was visualized and the visual tree structure was used to implement the model to the FPGA card. We used the if-else structure to implement the model. Starting from the root node, we carefully coded the model. Each node has an input variable and a split point that divides the data. The split point is

chosen from other possible split points by calculating the lowest cost while training. According to that split point, the subsequent two leaf nodes are constructed from the parent node by following each branch with the following calculated splits until the algorithm reaches its final prediction.

Input variable and corresponding splitting point were used to construct the

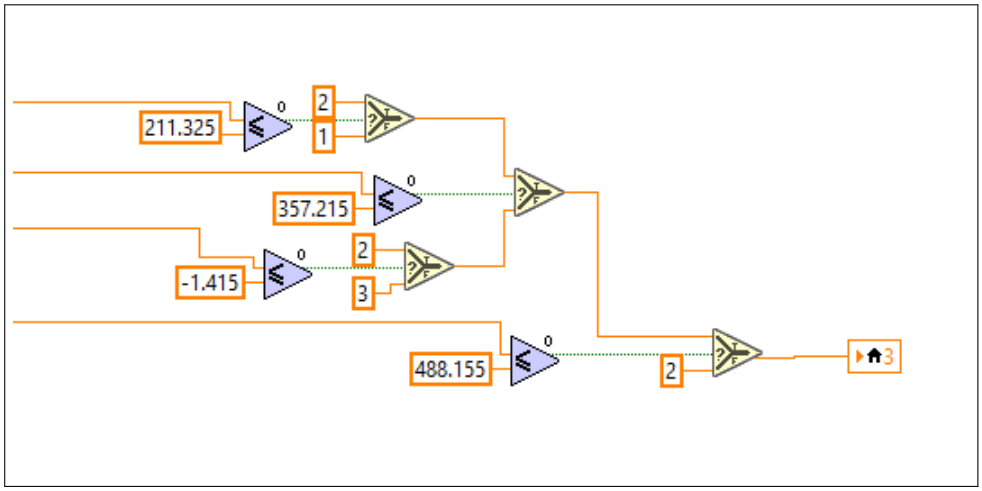

Figure 3.8 The decision steps to find the classification result for object type classification. If-else was used to implement the decision tree.

if condition. The following two if-else structures are created within the true or false statements using the subsequent nodes (child node) and their input variable and corresponding splitting point. Figure 2.8 shows the tree structure that was used to implement the object-type classification model. For instance, in Figure 2.8, the root node is located at the leftmost part. If the PIP value gets higher than 488.155, the classification reaches the terminal node and the result is flexion. However, if the PIP value gets lower than 488.155, the child node is selected. This process continues until all branches reach a terminal node. This method facilitates the implementation of decision trees to the FPGA and defining the split amount helps to control memory usage on FPGA card.

3.3 Signal Generation

The vibrotactile stimuli were generated in the FPGA card after classifying the data. We created an algorithm that checks the transitions between states to generate the stimuli.

The algorithm checks the changes in the classification results in real-time. The algorithm creates a window by collecting the following sixty incoming results in real-time and chooses the most frequent class among the classes in that window for object and movement-type classification results separately, whenever there is a change the algorithm generated the stimuli. However, if the classes do not change, the algorithm waits for a change. After choosing a class, the algorithm resets the window and sends the results to the next stage.

In the next stage, the algorithm uses the results from the window to define the DESC event and generates the signal accordingly. Then FPGA card sends the generated signal to the power amplifier to drive the actuators. If it is an object-type event, the FPGA card actuates Haptuator 1 and if it is a movement-type event, FPGA card actuates Haptuator 2. The block diagram of the algorithm is illustrated in Figure 2.9. The signal generation part of the code is shown in the Appendix (Figure a.14)

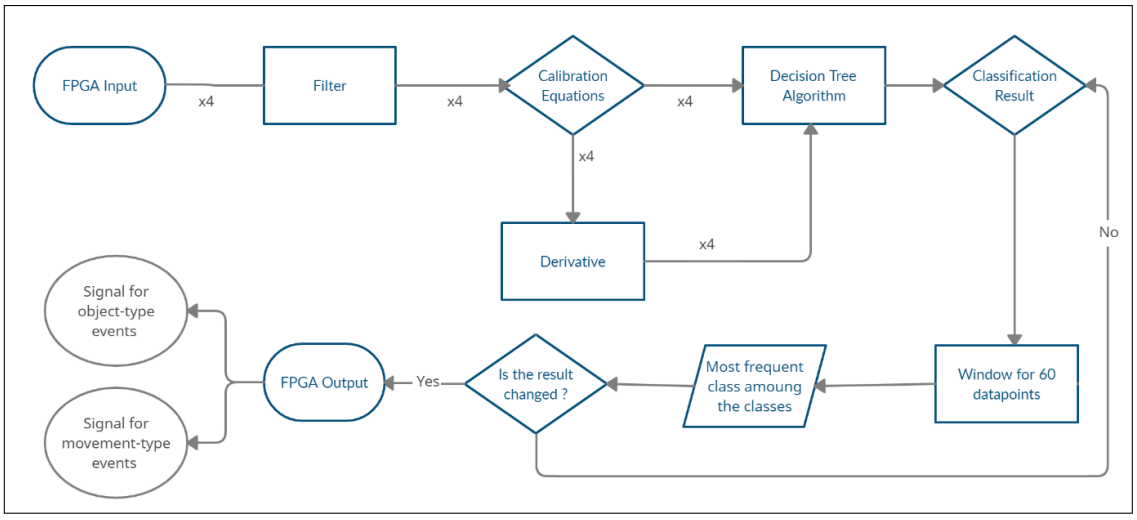

Figure 3.9 Multiple data paths are shown in the figure with a single arrow. To adapt to the DESC policy, the program waits until there is a change in the classification result either in the object type or the movement type classes. The window resets after reaching the 60 data points. The window is used to eliminate false classification results. FPGA card outputs are connected to the power amplifier to amplify the signal before driving the Haptuators (Figure 1.3).

3.3.1 Representation of The DESC Events

We mapped the object and movement type classes from decision tree classification to the frequency and the magnitude values from discrete prosthesis events, as shown in Table 2.

For object-type events and force-related feedback, Haptuator 1 was used and placed on the left arm. For movement-type events, Haptuator 2 was used and placed on the right arm. The classification results of the movement and object type events were used to define the events. For instance, to define the low force events, the algorithm checks the object-type classification result, whether it is the hard object or the soft object and checks the movement-type classification result whether it is contact or release. To define the high force events, the algorithm checks the object-type results, whether it is the hard object or the soft object and checks the movement-type classification result if it is flexion.

Two magnitude values (M1 and M2) were explicitly calculated for each participant, and two frequencies (F1 and F2, 80/180 Hz respectively) were used. Manipulating a soft object with low force was assigned to the frequency F1 and magnitude M1 and manipulating a soft object with the high force was assigned to the frequency F1 and magnitude M2. The frequency F2 was used to represent the hard object manipulation with the low and high force, where magnitudes stayed the same. These object-type events were assigned to the Haptuator 1. However, changes to no object did not actuate to the Haptuator 1.

Flexion without any object, in other words, flexion in the air, was represented using the frequency F1 and magnitude M1 and flexion in the hard or soft object was represented using the frequency F1 and magnitude M2. Extension in the air was represented using the frequency F1 and magnitude M1 and extension in the hard or soft object was represented using the frequency F1 and magnitude M2. These movement-type events were assigned to the Haptuator 2. When the movement changed to stationary, Haptuator 2 was not actuated. All of the events with their given codes are shown in Table 2.3.

Table 3.3
Table of events and their frequency and magnitude values. Modified from [12].

Haptuators	Events (codes)	Frequency	Magnitude
	Soft object/low force (L1)	F_1	M_1
Haptuator 1	Soft object/high force (L2)	F_1	M_2
(Object-type events)	Hard object/low force (L3)	F_2	M_1
	Hard object/high force (L4)	F_2	M_2
	Flexion/no object (R1)	F_1	M_1
Haptuator 2	Flexion/in object (R2)	F_1	M_2
(Movement-type events)	Extension/no object (R3)	F_2	M_1
	Extension/in object (R4)	F_2	M_2

3.3.2 Vibrotactile Stimuli

The stimuli were generated in the FPGA by using one-dimensional Look-Up Tables (LUTs). To create the LUTs, MATLAB and LabVIEW were used. First, using MATLAB, two sinusoidal signals with frequencies 80Hz (F1) and 180Hz (F2) were created separately. These signals were sinusoidal mechanical displacements presented in bursts with 50 ms cosine squared rise and fall times. The signal duration was 600 ms, including 50 ms rise and 50 ms fall time. Note that magnitude was calculated in the FPGA by multiplying the signal with stimulus magnitudes (M1 and M2) which were defined individually for each participant. Then, the created signal was exported to LabVIEW. By using LabVIEW, LUTs were created using the signal data. Finally, the LUTs were saved and used in the FPGA. The settings and the representation of the signal in the LUT's are shown in the Appendix (Figure a.15 and Figure a.16).

The delay between each signal was defined as 300 ms and the features of each signal were decided in the FPGA by using classification results and the DESC events from Table 2. Figure 2.10 shows an examples of the signal output of the FPGA card. Signals that are sent to the Haptuator 1 were shown in the upper graphs, and Haptuator 2 outputs were shown in the lower graph of the example. In Figure 2.10, the first signal represents the flexion in the air (R1) event. Therefore, the frequency is F1 and the magnitude is M1. Because the algorithm looks for the transitions, the

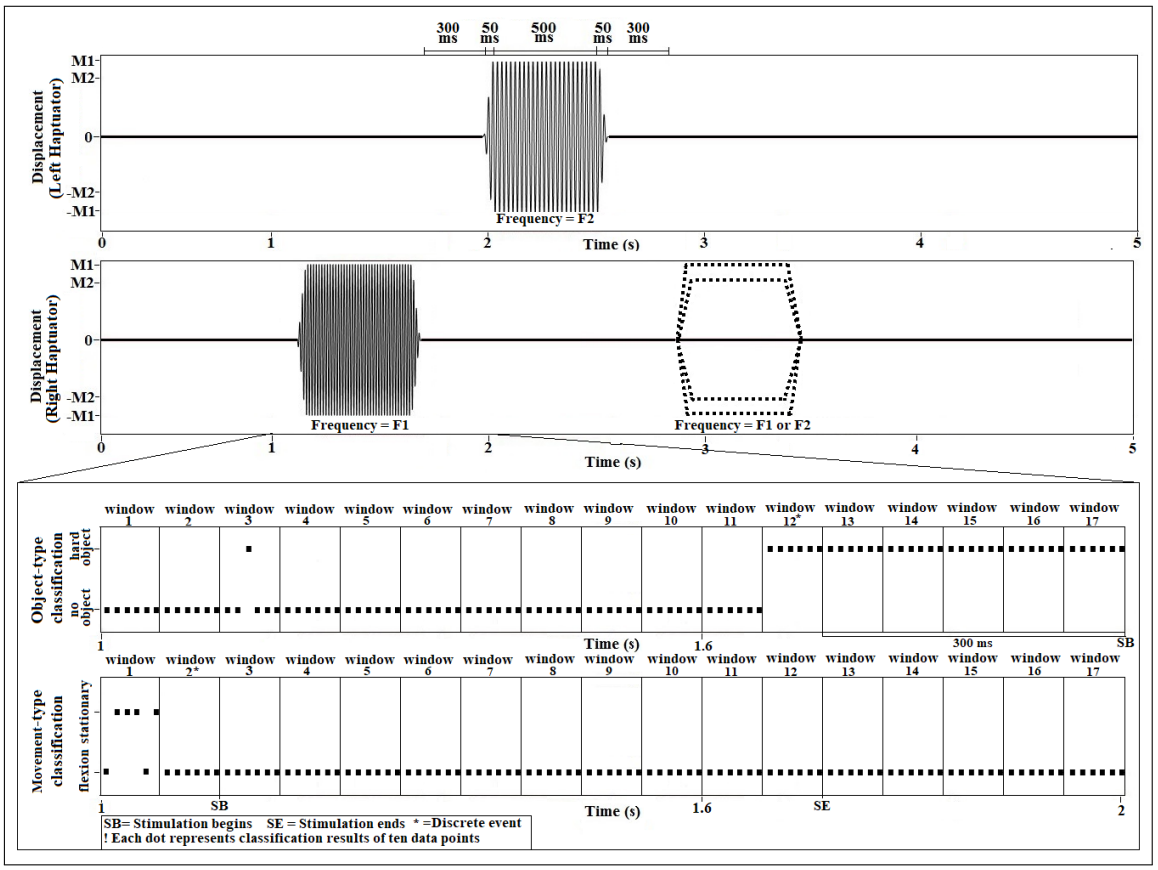

Figure 3.10 Example of the stimuli that have different frequencies for the DESC experiments. This example shows R1-L3 movements from Table 1 and the third signal represents the possible stimuli (in this case R3, R2 or none) according to the input data. Signal durations are 600ms with 50ms increasing time and 50ms decreasing time. The waiting time between the signals is 300ms. The number of signals, amplitudes and frequencies change according to the input data. Also the windows and the classification results of the data points are shown in below graph. Below graph only shows the time between one and two seconds of the above graph.

previous state of the first signal might be stationary or extension in air. The second signal represents the low force on a hard object (L3) with frequency F2 and magnitude M1. The previous state was flexion in the air and there is a 300 ms interstimulus interval between the signals. According to future events, the third signal might have the frequency F1 or F2 and the magnitude M1 or M2. Also, there might not be any signal if no transition occurs to a new state. The lower graph shows the a portion of the classification results processed in real time. Windows are generated with sixty data points to take most frequent classes among the classes each time. In window 1 and window 3 the classification mistakes can be seen however they do not affect the algorithm because of the window. In window 2 of the movement type classification graph, the frequent classes change from stationary to flexion. The algorithm generates

vibrotactile stimuli on Haptuator 2 after this discrete event transition. Than the stimuli lasts for 600 ms and then interstimulus interval for 300 ms passes. By the end of the stimuli the events change in object type classification as shown in window 12 and after the interval the algorithm generates the stimuli on Haptuator 1.

3.4 Participants

Four healthy female and two male participants, aged from 24 to 33, were included in the experiments. These participants have also joined the previous study conducted in our laboratory [12]. In the beginning, all participants were informed about the experiments and signed an informed consent. Also, participants stated that they did not have any dermatological, neurological, or psychiatric disorders that could affect experimental results. The experiments approved by the Institutional Review Board for Research with Human Subjects of Bogazici University.

3.5 Psychophysical Experiments

The experiments were completed in one session. First, absolute detection threshold values were measured for each participant. Then the DESC experiments were conducted.

3.5.1 Absolute Thresholds

Absolute detection thresholds were measured for each actuator placed on the left and right arm. For each actuator, two threshold values were measured for each vibrotactile frequencies (F1 = 80 Hz and F2 = 180 Hz). Adaptive tracking method was used in a two-interval forced-choice task [147, 53].

The participants were sitting in front of a computer screen and the experiments were done using MATLAB. The stimulus was generated in the computer and sent to the

data acquisition card and then to the FPGA card. In absolute threshold experiments, the algorithm was bypassed to send the stimuli directly to the power amplifier to drive the actuators. The experiment started with user entry and red, green and yellow squares appeared in sequence. During red or green intervals, stimuli were presented randomly with equal probability. Interval durations were 2s. The yellow interval was used for user feedback using two predefined keys to decide which interval the stimulus was presented. If the selection was correct, the yellow square blinked and if it was wrong, the yellow square did not blink. In the beginning, the stimulus was easily detectable and it was changed according to the correct or incorrect responses using three-down one-up rule [148, 12]. Whenever the last 20 trials were between ± 1 dB, the experiment was ended and the mean score of this range was used as the threshold value.

3.5.2 DESC Experiments

Discrete event-driven sensory feedback control experiments were conducted after measuring the absolute threshold values. Since the participants were the same as the previous study [12], the psychophysical models were reused in this study. The new measured absolute thresholds and the psychophysical models were used to define the magnitudes. MATLAB was used for interaction with the participants. The raw data was generated from MATLAB and sent to FPGA card through the data acquisition card. The FPGA card generated the signal automatically in real-time using Table 2.3. The raw data were split according to the transitions in Figure 2.12 and used in the experiments. All possible events are shown in a block diagram in Figure 2.12. The arrows indicate transitions from one state to another state and double-sided arrows indicate that transitions between those states can be either way.

Before starting the experiment, participants were trained to distinguish between events using MATLAB. During the training session, vibrotactile stimuli were created for all transitions in Figure 2.12. While giving the feedback, the pictures of the corresponding transitions were shown (Figure 2.11). In Figure 2.11, arrows indicate the direction of the motion. Participants were trained as long as they needed. For the

DESC experiments, fourteen predefined sequences were used [12]. These sequences are made of two or three consecutive discrete events. The pictures from Figure 2.11 were combined to represent events in those sequences. The initial steps were shown at the beginning of each sequence.

At the beginning of the experiments, participants were asked to press a button to start the experiment. Then, vibrotactile stimuli were given according to the randomly selected sequences. A red square was displayed on the screen during the vibrotactile feedback. Pictures of all sequences were then presented in a window for selection. Participants were asked to identify the sequence of vibrotactile stimuli presented and they were asked to select the correct sequence by pressing on the corresponding picture. Actual sequences and participants' responses were recorded on the computer. The experiments took about two hours and ended after 140 trials of ten randomly allocated repeats for each sequence.

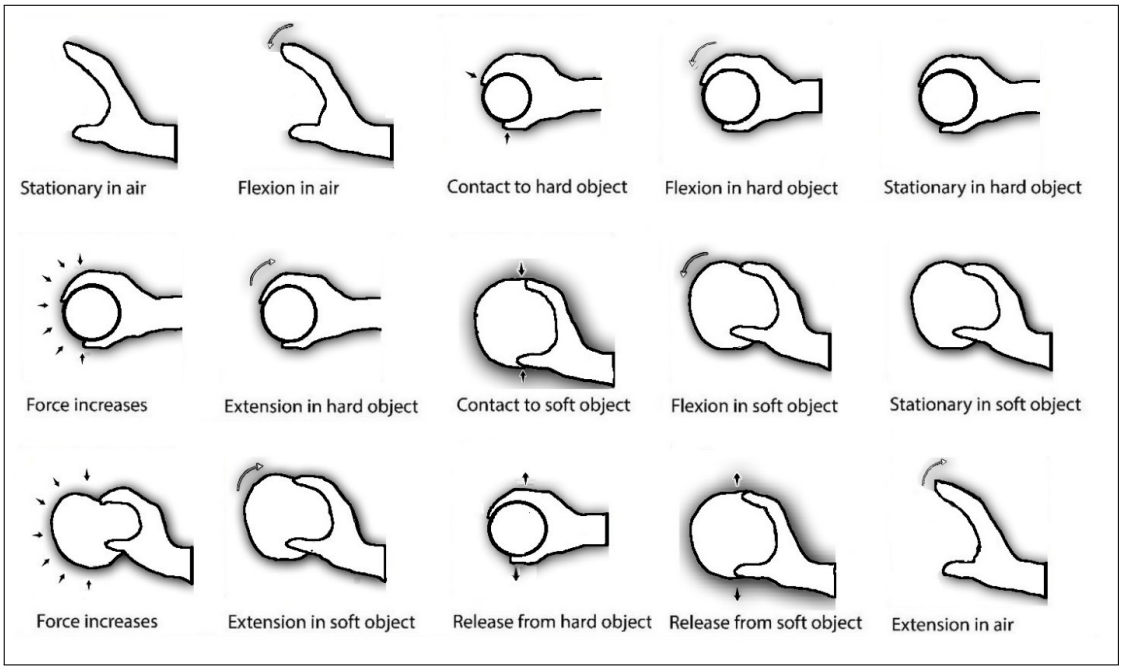

Figure 3.11 The pictures were used in DESC experiments to show fourteen sequences that were created by combining two or three consecutive discrete events. Modified from [12].

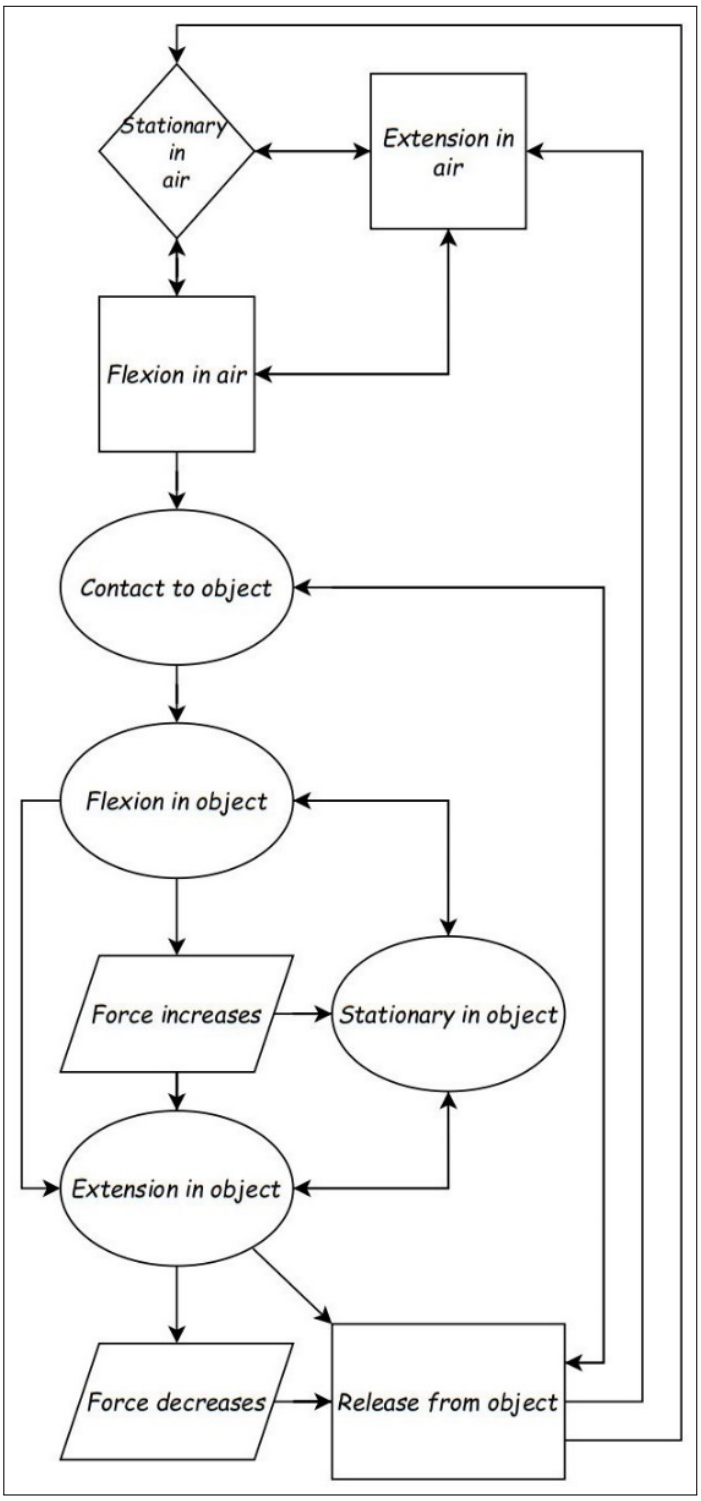

Figure 3.12 States/transitions that are shown in this block diagram were observed in a typical cylindrical grasp task. Modified from [12].

3.6 Statistical Analyses

Statistical analyses were performed in MATLAB 2019b and SPSS [149]. Two-way non-parametric ANOVA was used to analyze the differences in threshold values

converted from dB to micrometer units. Stimulus frequencies (80 and 180 Hz) and stimulus site (right and left arm) were the parameters for the analysis. Because the number of participants were low, thus the sample size was low, non-parametric ANOVA were had to be used. Therefore ARTool [150] was used to convert the data using align and rank transform for analyzing multi-factor at a time using two-way ANOVA. For the DESC experiments, one proportion z-test was used to compare the proportions of the correct responses in the confusion matrix to the chance level. A chi-squared test was used to assess the proportion of predicted classes' homogeneity. Two proportion z-test with Bonferroni correction was used to compare the proportions of the correct responses in the confusion matrix of this study and offline results [12]. Class averaged recall, precision and F1 scores were calculated. Offline performance scores from [12] were compared to calculated performance scores using non-parametric Wilcoxon rank-sum test.

4. RESULTS

4.1 Classification Results

The classes for object-type events are; no object, hard object and soft object. The movement-type classes are; stationary, flexion, contact, release and extension. The algorithm looks for changes in both events. Figure 3.1 shows a part of the data which is one cycle representing grasping a hard object. The data were taken from the ring finger, which has Ring-MCP, Ring-DIP, Ring-PIP, Ring-Distal, Ring-Palmar sensors mounted on it. The hand performed 80 cycles cylindrical grasp, where 35 cycles were for the hard object, 35 cycles for the soft object, and ten cycles for without any object. There were 560000 data points and each data point took 1 ms in the data. For one cycle there were 7000 data points. The windows lasted sixty ms. The data starts from stationary in the air (no object) and continues with flexion in the air. After contacting the object, the object-type class changes and flexion continues. One cycle of the data were split into classes, as shown in Figure 3.1 [25]. The calibration took place in the FPGA. Figure 3.1(a) shows an example of the bend sensor data and Figure 3.1(b) shows an example of the force sensor data. To train the MLR model, the data were exported to MATLAB and to train the decision tree model; the data were exported to Python. Later, the trained models were implemented on the FPGA card.

For the MLR the data were sent to the FPGA by using the LabVIEW interface. The data of five sensors were sent the FPGA for calibration and classification. However, for the DT, the data of four sensors, (MCP was excluded), were sent from the analog outputs of the data acquisition card to the inputs of FPGA card. That was because, the daq card had only four outputs. First, the data were filtered (low pass Butterworth filter) to get the contact forces and joint angles. Then the data were calibrated using the Eq. 1.1 and Eq. 1.2 in Section 3.

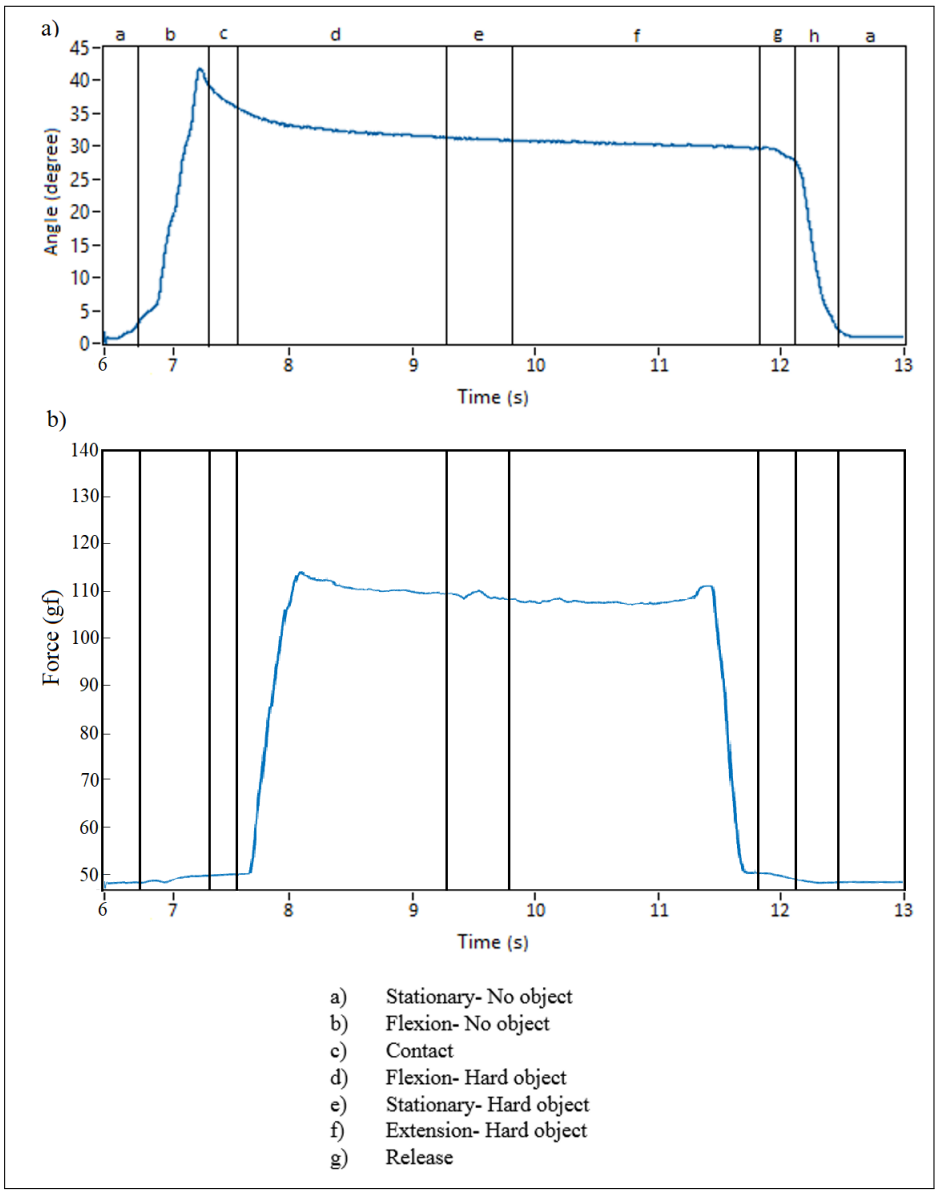

Figure 4.1 The labels which were extracted from the example data. The data represents one cycle of grasping a hard object. (a) The bend sensor data example. The cycle lasts seven seconds and shows the grasping a hard object. (b) The force sensor data example. The cycle lasts seven seconds and shows the grasping a hard object. Modified from [13].

4.1.1 Multinomial Logistic Regression Results

The resources on FPGA are defined by the number of slices where a slice is composed of look-up tables (LUTs) and flip flops. Default numeric controls (add, multiply, etc.) consumed too many LUTs, and the memory was not enough for those operations. To overcome this issue, LabVIEW FPGA Floating-Point Library by NI. [144] was used, and it decreased memory and LUT usage significantly. With the default numeric con-

trols 76427 LUT's were used, however with the floatin-point library only 19200 LUT's were used for the same code.

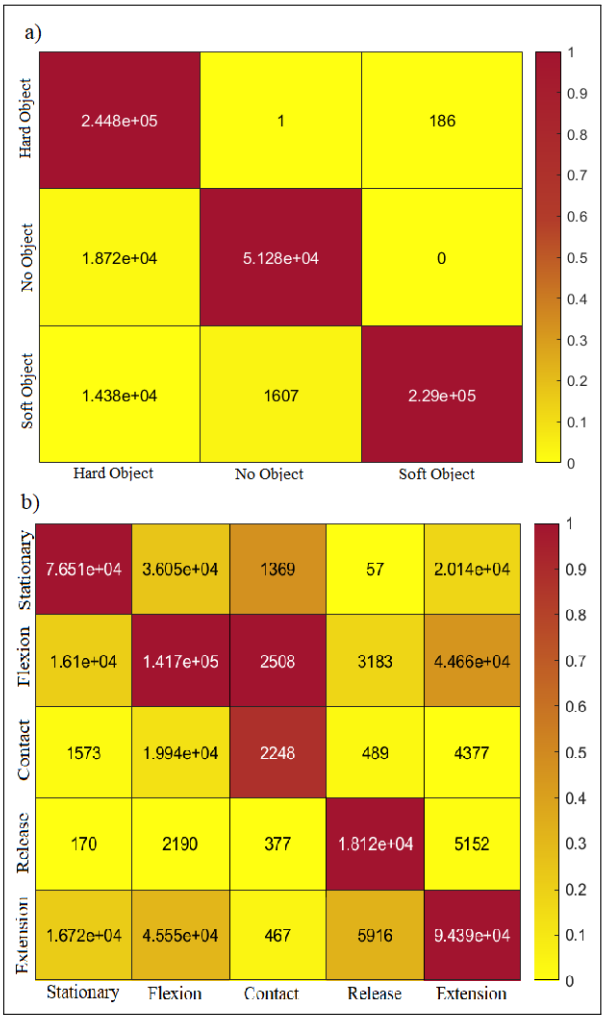

Figure 4.2 Confusion matrices of the MLR classifications. (a)Confusion matrix for object type classification. The data with 560000 data points were tested in FPGA card. The model was very accurate at predicting the classes with 93% accuracy (b) Confusion matrix for movement type classification. The data with 560000 data points were tested in the FPGA card. The accuracy of the model was low (59%) hence, MLR was not a best choice for movement type classification.

The data were randomly shuffled and the five-fold cross-validation method was used for training. Simply cross-validation divides the data into five equal parts, allowing each part to be used for both training and testing. Thus bias and errors caused by dispersion and fragmentation are minimized. Cross-Validation allowed us to see whether the model's high performance was random or not [7].

Table 3.1 shows the classification results done in both FPGA card and MATLAB. Real-time classification yielded lower results than the offline classification [12].

Table 4.1
Comparison between offline [12] and real-time classification of MLR.

Device	Classification	Classification Accuracy
Offline classification with MATLAB	Object type	0.98
	Movement type	0.72
Real-time classification with FPGA	Object type	0.94
	Movement type	0.59

That is because, in offline classification, feature normalization was done to the data after calibration, which is not applicable in FPGA. We can see that object type classification gives an excellent result. The confusion matrix of the object type results are shown in Figure 3.2(a). However, for movement type, it is clear that we need to use different algorithms to get better accuracies. The performance scores of the real-time MLR classification is shown in Table 3.2. We can see that for object type classification

Table 4.2
Classification results of real-time MLR classifier.

Classification Type	Classification Labels	Recall	Precision	F1 Score
Object type	No object	0.73	0.97	0.83
	Soft object	0.93	0.99	0.96
	Hard object	0.99	0.88	0.94
Movement type	Stationary	0.57	0.68	0.62
	Flexion	0.68	0.57	0.62
	Contact	0.08	0.32	0.13
	Release	0.70	0.65	0.67
	Extension	0.58	0.56	0.57

the results are high. However, for the movement type the recall, precision and F1 scores are low and in Figure 3.2(b), this indicates that, the model was best at classifying the

flexion and worst at classifying the contact. The model often misclassified the contact with flexion and extension. That might be because contact comes after the flexion and before the extension and there might be some similarities between contact, flexion and extension. So that, the model might not be able to discriminate them from flexion and extension.

4.1.2 Decision Tree Classification Results

The data were randomly shuffled and the 10-fold cross-validation method was used for training. The decision tree structure of the object-type classification is shown in Figure 3.3. The first line at each node shows where the data split for that feature. For instance, the model checks if feature 8, which is the data of derivative of the DIST, is whether greater than 488.155 for the root node. The Gini indexes indicate the success rate of the split. Zero means the best split. It can be seen that the terminal nodes are very close to zero, which indicates that the classification for the object-type events will be very successful for the sensor data. The sample amount in that split is shown in the third line. For instance, after the root node, the data is split into two, and the tree reaches the terminal node if the condition is met. After reaching the terminal node, it can be seen that 138112 samples from training data met the condition of the root node. The value in the fourth line shows the number of samples that belong to each class for that node. For instance, in the terminal node after the root node, there are 138112 samples and 593 of them belong to the first class, 134944 samples belong to the second class and 2575 of them belong to the third class. The last line shows the decision class for the terminal node. So after the root node, if the data of derivative of the DIST gets lower than 488.155, the model concludes that it is the second class.

The number of splits was limited by defining the amount of the leaf nodes. This helped us to implement the model into the low memory FPGA efficiently. We defined and tested several split criteria and checked how the accuracies change accordingly. Graphs in Figure 3.4 show the number of leaf nodes and their training accuracies for both object and movement-type classifications. Note that these results were calculated by taking averages of the cross-validation results. Figure 3.4(a) shows the object-type

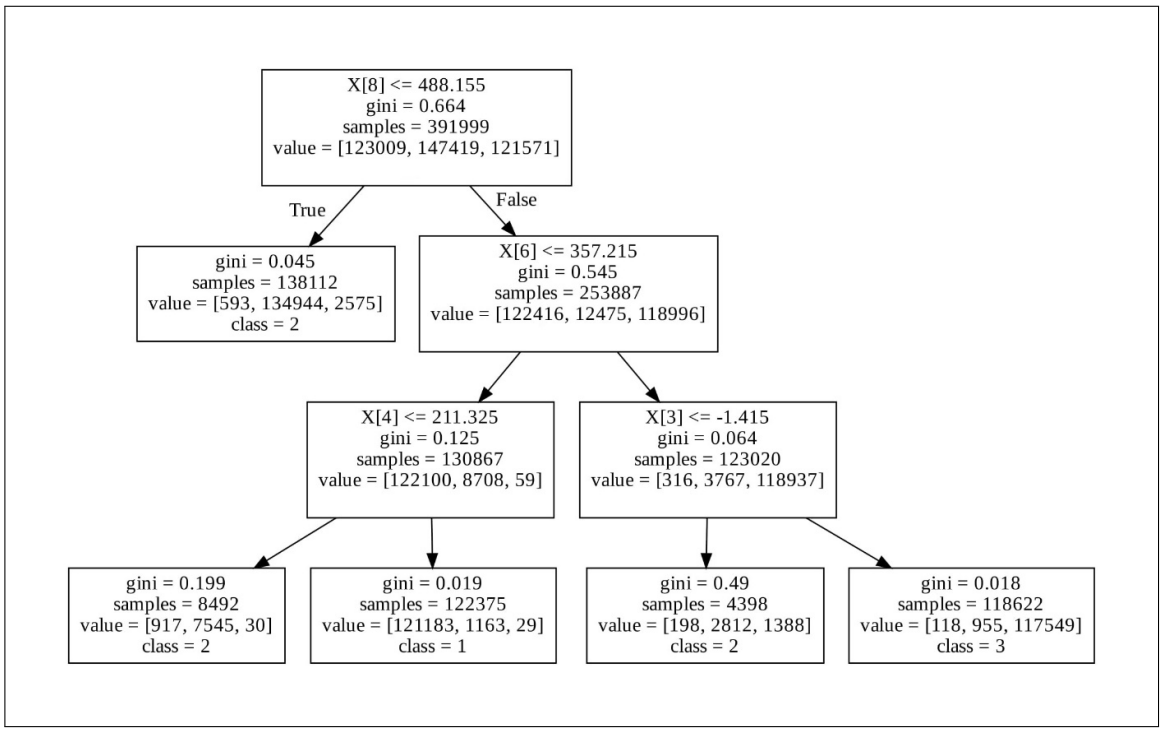

Figure 4.3 The Gini indexes show how well the split was done. The Gini value gets a result between 0 and 1, and the closer the result is to 0, the better the discrimination is. The number of samples was shown in each node. The value array represents the discrimination of the samples that were used in the node. There are three classes in object type events; therefore, samples were divided into three groups in the value array. The decision tree algorithm takes the maximum number to decide the class at that node. Five leaf nodes were used for split criteria.

classification results. The classes of the object-type classification are easily distinguishable from one another. That is why the accuracy converges rather quickly. We only plotted the numbers up to 30 in this graph to show the detailed information. After careful consideration, split criteria with five nodes were selected to achieve the optimum results with the minimum leaf nodes possible. The test data accuracy was % 97.

Figure 3.4(b) shows the training accuracies for movement-type classification. Since it is more challenging to distinguish classes than object-type classes, accuracy increases slowly. The classes and their places on the example data can be seen in Figure 3.1. Table 3.3 shows the detailed results of the movement-type classification. Three different accuracies were tested and their memory usages were observed. The slice LUT's are basically the LUTs that are a set of logic gates. For every combination of inputs possible, LUTs store the predefined list of outputs and provide a quick way to get the output of a logic operation.

Table 4.3
The results of decision tree algorithm for movement type events.

Nodes of Decision Tree	Offline Training Accuracy	Memory Components	Used	Total	Percentage	Accuracy of Test data
		Total Slices	5465	10250	53.3%	
19	0.72 ± 0.02	Slice Registers	16764	82000	20.4%	0.69
		Slice LUT's	16324	41000	39.8%	
		Total Slices	7068	10250	69%	
80	0.81 ± 0.03	Slice Registers	22071	82000	26.9%	0.78
		Slice LUT's	22164	41000	54.1%	
		Total Slices	9147	10250	89.2%	
290	0.90 ± 0.02	Slice Registers	28152	82000	34.3%	0.88
		Slice LUT's	29797	41000	72.7%	

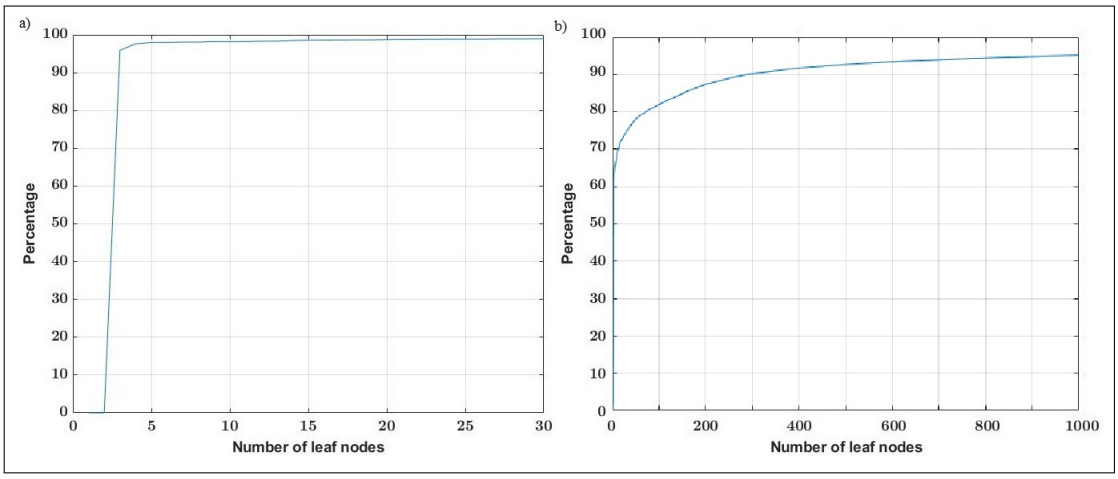

Figure 4.4 (a) The graph for classification of object type events. The classes are easy to discriminate and the percentage converges to 100 quickly (b) The graph for classification of movement type events. The classification is more complicated than object-type events. It requires 290 leaf nodes to reach 90%.

Slice registers are made of flip-flops. It stores a bit pattern that holds the states between iterations, synchronizes input and output and communicates with the host VI that LabVIEW offers [151]. The model reaches 72% accuracy, which is the average

of 10-fold cross-validation using 19 leaf nodes. Implementing the model on the FPGA uses nearly half of the memory. Note that these memory results contain the calibration step and the decision tree model. Since there is a lot of memory left unused, we tried to implement a model with 80 leaf nodes that resulted in 81% accuracy. This model uses 69% of the memory. Lastly, to reach 90% accuracy, 290 leaf nodes were used and implemented on the FPGA card. The reason we choose 290 was to achieve the optimum results with the minimum leaf nodes possible to use memory in the most efficient way. With this model, % 89 of the memory was used. After implementing the decision tree models of both object and movement types, real-time performances of the models were tested with the data.

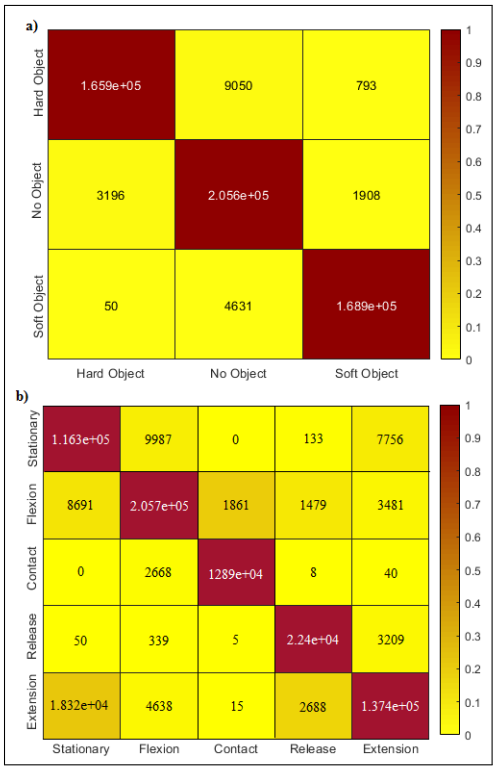

Figure 4.5 (a) Confusion matrix for object type classification. The data with 560000 data points were tested in FPGA card. Five leaf nodes were chosen for split criteria. The decision tree classification performed exceptionally well with %97 accuracy. (b) Confusion matrix for movement type classification. The data with 560000 data points were tested in the FPGA card. The decision tree classification using FPGA card performed considerably well with %89 accuracy.

The accuracy of the object-type classification was %97 and the accuracy of the movement-type classification was %89. The recall, precision and F1 scores of the DT classification are shown in Table 3.4. The score are high for both the movement and object type classifications. We can say that DT can successfully discriminate the

classes in these data types. Figure 3.5(a) shows the confusion matrix of the decision tree model for object-type classification. The model was very successful in predicting the classes. The highest error was predicting no object where it was actually a hard object. The model learned well to separate hard and soft objects, which gave the lowest error. Figure 3.5(b) shows the confusion matrix for the movement-type classification performance of the model. The model was successful however had some difficulties discriminating extension from the stationary. Also, the model predicted flexion while it was actually an extension. The reason might be because some of the angle values are the same for the stationary, flexion and extension. A small percentage of the contact was misclassified with flexion and vice versa, probably due to contact occurs after the flexion. Furthermore, because release comes after extension in some situations, the model was confused about classifying some of the releases with the extension. So some similarities may cause errors in a small percentage of the data.

Table 4.4
Classification results of real-time DT classifier.

Classification Type	Classification Labels	Recall	Precision	F1 Score
Object type	No object	0.98	0.94	0.96
	Soft object	0.97	0.98	0.98
	Hard object	0.94	0.98	0.96
Movement type	Stationary	0.87	0.81	0.84
	Flexion	0.93	0.92	0.93
	Contact	0.83	0.87	0.85
	Release	0.86	0.84	0.85
	Extension	0.84	0.90	0.87

4.2 Psychophysical Results

Psychophysical experiments were conducted in one session for each participant. Absolute detection thresholds were measured for two vibrotactile frequencies (80 Hz

and 180 Hz) and two stimulation sites (right and left arm). Absolute detection thresholds for each participant are shown in Figure 3.6. Each participant is shown in different colors and the threshold scores are divided into four groups using the frequency and site that were used to measure the thresholds. To compare with the literature, the threshold scores in the y-axis are plotted in dB referenced to 1 µm. Note that thresholds were measured only once in this study. The experiment started by placing the Haptuator 1 on the left arm and the thresholds were measured for 80 Hz and 180 Hz, respectively. Then Haptuator 2 was placed on the right arm and the same measurements were taken. Threshold values were consistent with the literature [12, 4]. Threshold differences were analyzed using two-way ANOVA. There were no interactions found between frequency and site or their significant effects on thresholds. Threshold values were used in FPGA to adjust the stimulation amplitude for each participant using the psychophysical models that were established in the previous study [12].

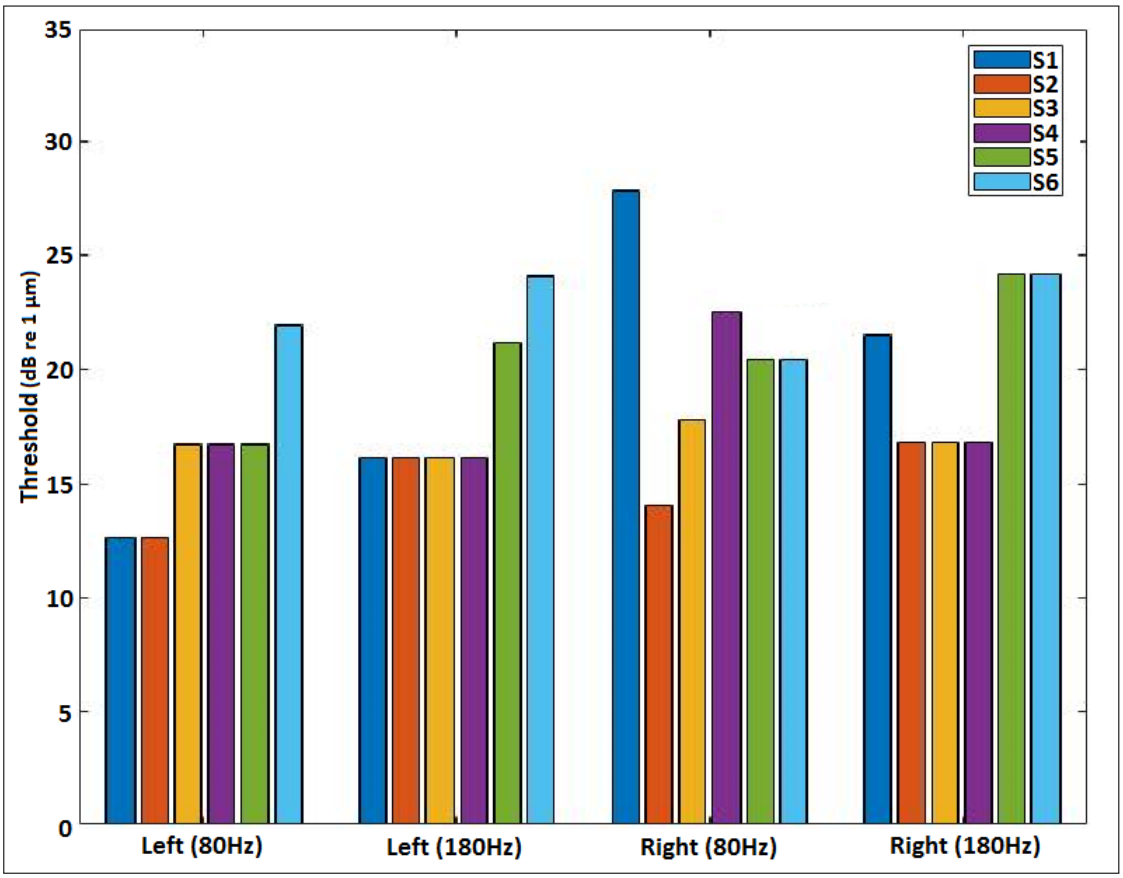

Figure 4.6 Absolute detection thresholds for each participant at different frequencies and locations. The thresholds were measured only once by the adaptive tracking method and the results are shown as bars.

DESC experiments were conducted right after the absolute threshold experiments. Participants were trained to learn and differentiate the discrete event-driven vibrotactile patterns, as shown in Table 2.3, before the DESC experiments. The length of the training session was dependent on the participants. For the actual experiments, fourteen sequences were created with two or three discrete events from Table 2.3. To be able to compare both performances, these sequences were selected similar to the previous study [12]. The sequences and their event codes are shown in Figure 3.7. These sequences were defined in MATLAB by combining the raw data that represents the events. Raw data were sent to the data acquisition card and eventually to the FPGA card. FPGA card generated the stimuli in real-time using Table 2.3 and the algorithm that was defined in section 2.3. The stimuli duration was 600 ms with 50 ms rise and fall time and FPGA waited 300 ms before generating the next stimulus.

Figure 3.7(a) shows the confusion matrix, which was created from pooled data of the participants' performances in the DESC experiments. The diagonal shows the correct responses for each class. The performance of the participants on choosing the correct response was moderately acceptable. Participants had the worst performance on identifying sequence ten (contact to hard object, flexion in hard object, force increases). This sequence was mostly mistaken with sequence nine, which were contact to soft object, flexion in soft object, force increases. However, their best performance was identifying sequence four (stationary in air, flexion in air, contact to hard object). This sequence was confused with sequence three (stationary in air, flexion in air, contact to soft object). Participants were commonly mistaken in identifying sequences that are similar to the correct sequences. Especially, discriminating the hard object from the soft object and vice versa was a common mistake. Statistical analysis comparing the correct responses to the chance level showed that the correct responses were significantly above the chance level ($p<0.002$). In addition, the chi-square test result was $p<0.05$, which indicated that target and predicted sequences were significantly dependent. The confusion matrices of each participant's responses and their performance score are shown in the Appendix (Figure a.1-a.6 and Table a.1-a.6).

We compared our results with the previous study [12]. Proportion differences are shown in Figure 3.7(b). Positive percentages indicate that participants performed better in our study than in the previous study at that sequence. Negative percentages

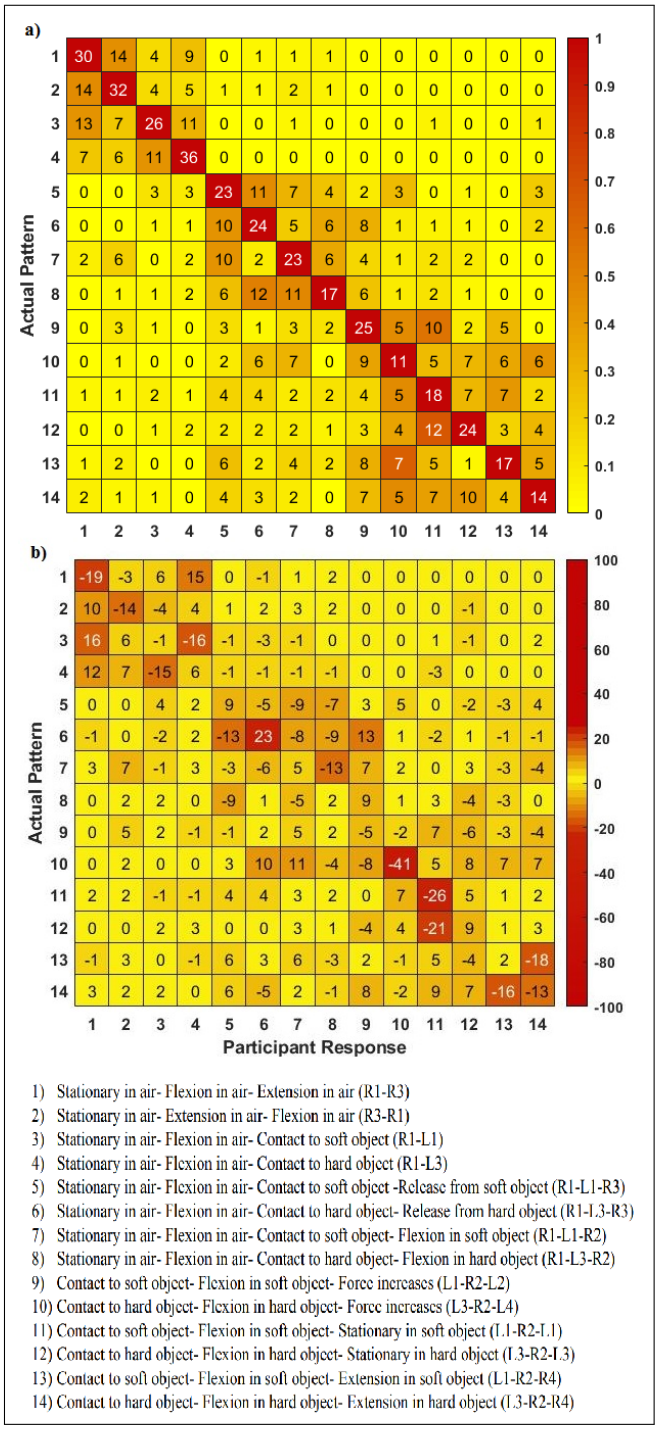

Figure 4.7 (a) The confusion matrix of pooled results from the DESC experiments. The actual number of responses is shown in the matrix cells. The numbers that represent the class labels are the sequences of the events. (b) The matrix of differences between offline [12] and real-time pooled results from DESC experiments. The differences are in percentage and negative percentages indicate that offline percentages are bigger than the real-time percentages. The numbers that represent the class labels are the sequences of the events.

indicate vice versa. Note that six participants completed the DESC experiments in this study; however, seven participants were able to complete the DESC experiments

in the previous study [12]. Therefore, the lack of one participant might be the reason for some differences. The most significant difference was in sequence ten which was negative. Sequence ten was the worst performance in this study, so that difference is understandable. The most significant difference among the positive percentage was sequence six. Although our study lack of one participant, participants were better at identifying some sequences than offline study. The differences between the correct responses of both experiments (diagonal of Figure 3.7(b)) were analyzed using Bonferroni corrected two-proportion z-test. The null hypothesis means there should be no difference between the diagonals of the two studies. All p values were greater than 0.004, which fails to reject the null hypothesis and implies that there is no significant difference between the two studies.

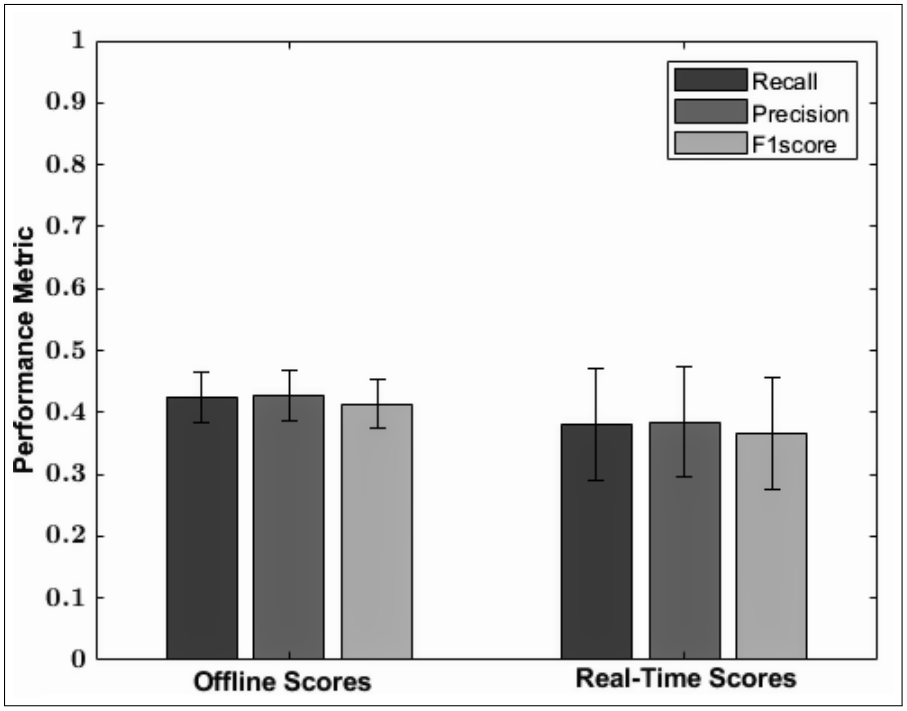

Figure 4.8 Overall mean performance scores of the DESC experiments compared with the offline scores from [12].

Figure 3.8 shows the class-averaged recall, precision and F1 scores from each participant for real-time scores of this study and the offline scores [12]. No significant differences were found between the two experiments (all p's > 0.05). Even though the real-time recall (0.38 ± 0.08), precision (0.38 ± 0.09) and F1 score (0.37 ± 0.09) are slightly lower than the offline scores.

5. DISCUSSION

5.1 Previous Studies

FPGA is used in various applications and scopes for decision tree applications. Narayanan et al.[120] implemented the decision tree model for training the model using a Gini score calculation method. They used HDL to implement the algorithm to the FPGA. The main focus was to accelerate the training process. Their study was the first attempt to implement decision tree models in FPGA. However, we trained the model offline and implemented it in FPGA for classification due to memory limitations. Alhammami et al. [152], Pittman et al. [153] and J. Oberg, et al. [121] implemented random decision tree models on FPGA for body part recognition using Microsoft Kinect. However, these designs cannot be used in low-power FPGAs due to memory constraints. Taiga Ikeda, et al. [118] proposed the threshold compaction method to merge similar thresholds and reduce the size of the gradient-boosted decision tree models, which simply combine decision trees in series to achieve a strong learner from many sequentially connected weak learners. Rafal Kulaga and Marek Gorgon [119] designed a decision tree classifier to recognize letters and digits using FPGA. They wrote the decision tree model in a high-level language, such as C, and then used Vivado to convert it to a hardware description language. LabVIEW was used as a hardware description language in our study because it is easier to use, more efficient and effective, which is the first study to our knowledge that implements the decision tree algorithm using LabVIEW to the FPGA.

G. Di Patrizio Stanchieri, et al. [135] proposed an electro-tactile sensory feedback system using optical fiber data communication link. However, they were focused on an optical fiber data communication link between two FPGAs, not the sensory feedback. They found improvement in power consumption, transmission data rate and the robustness to electromagnetic disturbances.

Vibrotactile feedback is widely used to give sensory feedback to the patients in the literature [23, 59, 154, 155, 156, 12, 63, 64]. Patterson and Katz used both vibro-

tactile and mechanotactile feedback in 1992 and they used a myoelectric hand to give grasp force information. They found out that the feedback reduced the grasping errors. [63]. After that study, the myoelectric hand was used to get grab force information and only continuous vibrotactile feedback was applied in some studies, which improved prosthetic use performance slightly better [59, 64, 58]. However, continuous feedback increases the processing load and forces the user to focus on the stimuli for changes continuously to identify the events. To solve that problem, the DESC policy was used in non-invasive sensory feedback for the first time by Christian Cipriani, et al. [22]. Participants of that study controlled the robotic hand by their thumb and index fingers. The robotic hand was attached to the participants' forearm and they grasped and lifted off an object during the experiments. More recently, they used the virtual egg test by using a DESC glove, which was composed of vibrating motors and sensorized digit thimbles. The result was promising in terms of performance improvement [21]. They also showed that discrete feedback reduces slip in another study [23]. However, in these studies, the stimulus duration was short and vibration intensity, frequency and duration were adjusted manually.

A recent study [12] was successfully done in our laboratory, where they used two actuators (electromagnetic vibrotactile) to understand psychophysical principles behind the DESC theory by using the patterns or stimuli changing in amplitude, location, and frequency, after placing them on the upper arms of 10 human subjects. They controlled the actuators by using audio power amplifiers, which were custom-made, and they generated the stimulus waveform from the data acquisition card controlled by a MATLAB script. To equalize the sensation magnitudes, they measured absolute thresholds, psychometric functions, and magnitude estimates. The subjects had to complete several tasks (same- different, pattern recognition, and DESC experiments). For the DESC experiments, they used 300 ms interstimulus intervals, which were sinusoidal signals with the frequencies 80 Hz and 180 Hz for the vibrotactile events. The magnitudes were calculated individually for the subjects to represent movement in air and movement in object. The actuator placed in the left arm was used to signal object-type (soft object, hard object, and no object) and force-related (low force and high force) discrete events. The other actuator was used to signal movement type (flexion in air, extension in air, stationary in air, flexion in object, extension in object,

and stationary in object) discrete events. These methods were used in this study in real-time. Enzo Mastinu, et al. [157] examined the effects on motor coordination while providing tactile perception. They compared the performances of no feedback, continuous feedback, discrete event-driven feedback and hybrid feedback which combined the continuous and discrete event feedbacks. The experimental procedure included routine grasping and grasping an object under uncertainty. They found that hybrid and DESC feedback yielded similar performance, which is better than no feedback and continuous feedback. However, they used the invasive sensory feedback method in their study. To the best of our knowledge, no attempt was made to apply real-time vibrotactile feedback using FPGA.

5.2 Technical Limitations and Other Issues

The FPGA that was used in this study (NI USB-7845R) offers low memory; therefore, some machine learning models, such as KNN, can not be implemented. This situation led us to use the MLR and DT, which are very efficient and do not require high memory usage. Another issue with the low memory was that training could not be done in FPGA. Therefore the model was trained offline and the trained model was implemented to FPGA. Also, usage of the floating-point numbers is limited and even with the SGL type, the precision is limited. The SGL type may occasionally throw a rounding error if significant digits get more than six [158]. This might result in an insignificant classification error for the model. Therefore, this error percentage can be neglected.

There were certain limitations on placing the actuators to the upper arm for the experiments. The actuators were mounted on a rubber foam and straps were attached to the foam to secure it to the arm. The actuators were mounted perpendicularly. During the experiments detaching the straps caused the experiment to start over from the beginning. Also, the displacement of the actuator might be affected by the skin's reaction in the opposite direction and resulting in a decrease in the performance of the participants. Although the actuators were placed nearly on the same site using the placement measurements from Karakuş and Güçlü [12] on the arm, there was

variation between the thresholds from the that study because of the indentation shifts and small location shifts. As stated in Section 1.2, change in the indentation effects the thresholds. These issues can be solved with the actuators attached directly to the prosthesis. Nevertheless, measured threshold values were consistent with the literature as can be seen in Figure 2.5 [12, 4]. The contactor area was 0.03 cm^2 in our study. It was previously shown that the spatial summation effect of the Pacinian channel could be neglected using small contactor areas [74, 159] and the thresholds follow a flat trend. Therefore it was expected to have no difference between 80 Hz and 180 Hz threshold results, which was the case in our study.

One another issue was with the Haptuator devices, although they have good linearity over a wide frequency range, they were not designed specifically to work with prostheses. They were generating vibrations on the casing which could generate complex stimuli on the skin. Therefore, we had to make some adjustments to the Haptuator by mounting small plastic probes as done in previous study [13]. However, sometimes during the experiments, the probes were dismounting and we had to remount them on the Haptuator. So that the experiments had to be retaken by the participants after recalculation of the absolute thresholds. This issue can be solved with using a Haptuator that comes with a probe designed by the manufacturer.

In our system, the FPGA decides on the events using the classification results. Although we used window to prevent misclassification errors, FPGA made mistake in 32 out of 840 sequences. To prevent these errors and improve the results, a simple decision algorithm can be added before applying the feedback. In Figure 4.7 we can see that, some events are not possible to come after a specific event. For example there is no possibility of force increase, when the event is stationary in air or there is no possibility of flexion in soft object when the current event is stationary in hard object etc. So that, the algorithm can eliminate the irrelevant events and wait for the relevant events that might come after the current event. For instance, if it is flexion to soft object the algorithm will wait for the extension in soft object, force increases event or stationary in soft object. When the FPGA makes a mistake and defines the event as stationary in air the algorithm will eliminate that event and might take the classification of the next window or wait for the next event. However, this algorithm might increase the memory usage of the FPGA.

5.3 Future Work

Future work may address using an available real prosthetic arm such as IH2 Azzurra hand (Prensilia SRL, Italy), Michelangelo hand prosthesis (Otto Bock Healthcare, Austria), etc. For the implementation of the system to the real prosthetic arm some improvements can be made. For example the Azzurra hand already has force sensors mounted on it. With the addition of the bend sensors, the hand can be used to provide feedback as presented in this study. With the hand's wireless communication feature, the sensor data can be sent to the FPGA wirelessly. Because of the FPGA card that was used in this study, is big and inefficient for the real prosthetic hand, one FPGA chip (such as Kintex-7 70T) with A/D converter can be added to a custom made circuit. The Haptuators can be placed on either each upper arm or stumps of each arm by designing a socket on the hand to secure them in their places. Also, a small custom made board with a small amplifier and one port D/A converter can be placed near each Haptuator. Wireless communication between the FPGA and the Haptuators can be accomplished by adding wireless communication cards to the boards. The proposed algorithm code can easily be implemented to the FPGA chip by getting the bit files that LabVIEW can create. The presented system can also become beneficial in surgery robots, such as da Vinci surgical system (Intuitive Surgical Inc., Sunnyvale, CA, USA), by allowing the surgeons to have the improved control in the surgery. It was shown that lack of haptic feedback is a limiting factor in robotic surgeries [160] and haptic feedback improves the control of the robot [161]. Our system can be implemented to the surgery robots and it can help to shorten the training time and decrease the surgeons' cognitive burden. On the other hand, artificial skin [162, 131, 132] may be implemented on a prosthetic hand using FPGA. That would improve the quality of the data and therefore, the model would perform better. Also, the training may be implemented in the FPGA to prevent errors that the sensors might cause after some time due to abrasion; and the model can be trained by the user during the use of prosthesis. Besides, to measure the absolute threshold values after reattaching the actuators, custom software on a computer or smartphone can help the user to measure using the adaptive tracking method and adjust the magnitude levels, so that the user would not have to go to a clinic or a laboratory for a simple calibration. The

performance of the presented system was tested on six healthy participants in a short time. The performance may increase with a long term study on prosthetic users and the results of the study can be compared with this study.

In our study we used force and bend sensors and their derivatives, as features of the machine learning algorithms. The force sensor represents exteroceptive information and bend sensors represents proprioceptive information in humans [163]. In the literature, derivative of the force sensor is shown to increase the accuracy while detecting object slippage [164, 165, 166] and discriminating the object types [13]. There are also other techniques such as; Fast Fourier Transform, mean absolute value, root mean square, waveform length, mean frequency, standard deviation, maximum, mean value etc. [167, 168, 169]. The frequency analysis could be relevant to distinguish the contributions of different mechanoreceptors such as Pacinian corpuscles and Meissner corpuscles [169]. In the future, these techniques can be tested and the technique that gives most informative features can be implemented. Thus the machine learning algorithm would perform better and accuracy can be increased. However, because we are using low memory FPGA, number of features has to be chosen as low possible.

This presented system can be applied invasively using cuff electrodes, TIME or LIFE electrodes for peripheral nerve stimulation (PNS) [170]. Two electrodes can be implemented in median and ulnar nerves separately. Tactile and proprioceptive sensations can be evoked by choosing different parameter set of rectangular biphasic current pulses [171, 172]. Fixed 50 Hz frequency was commonly used in PNS studies, which can be also used in this study [173, 45, 174]. The amplitude can be adjusted using staircase method to define thresholds and upper-limit pulse amplitudes. Pulse widths can be defined between 20 and 250 μs as done in literature [157, 173, 45, 174]. However, with our algorithm, the parameters of invasive pulses can be changed in every 60 ms.

In the future, activity recognition can be adapted to our system. Roggen et al. [175] showed that, by using accelerometer, gyroscope and magnetometer some daily activities such as opening and closing door,fridge,dishwasher or drawer, drinking from cup, preparing coffee, cleaning table, preparing and eating sandwich can be recognized. By using such method our system can recognize the activity and give the feedback accordingly. That would be more accurate and effective. Such system can be tested and compared to our study.

5.4 Conclusion

In this study, we proposed a method for real-time vibrotactile feedback using the DESC policy. Most of the highly sophisticated prosthetic devices are rejected by the users because of the substantial cognitive load [14, 176]. Therefore it is not very feasible at this time to convey all the physical information effectively. Some simplification is required to get effective results. In this study, we used only two levels of frequencies and two levels of magnitudes. However, even with this simplicity, the performance is not perfect. As we apply this feedback in sequence in daily use, some events might be missed. In the three pattern sequence of DESC stimulation, participants' performances were acceptable but not very good for recognizing all three events. However, at least two out of three were recognized accurately. When the user has this in a daily task, some will be missed because of task continuity. The user might get that information a second later or in a different way. It may not be feasible and efficient to introduce a lot of continuous types of information which will decrease the user performance. That is why the DESC is needed in these types of applications.

We used FPGA card to process the data and apply appropriate feedback according to the classification result. FPGA is known for its efficient power consumption that reduces costs for large-scale operations. This makes them a great choice as accelerators for battery-powered devices [177]. Thus it is a good choice for a prosthetic device. However, low memory makes it rather challenging to implement complicated machine learning algorithms. Therefore efficient yet straightforward algorithms are needed to be chosen. First, the MLR was chosen. The accuracy of the object type classification was % 93, and it was a very promising result. However, the movement-type (%59) accuracy was not satisfying enough. To improve the results, DT was used. The decision tree model yielded very promising results for both object type and movement type. Object type classification accuracy was 97% and movement-type accuracy was 89%. Decision tree models were controlled by defining split criterion. This simply facilitated the achievement of maximum accuracy using less memory. The proposed model in this study can easily be used in other machine learning applications, and the accuracy can be compared to our study.

The data were sent to FPGA card using MATLAB and data points were sent

one by one to mimic real-time applications using sensors. Data acquisition card was simply the communication link between the computer and the FPGA card. However, the data acquisition card will not be needed in real-time applications using a real prosthesis because sensors will be connected directly to the input ports of the FPGA card. An algorithm was created in FPGA card to filter, calibrate and classify the raw data and generate vibrotactile stimuli using DESC policy. To apply the DESC policy, the algorithm checked the changes in classification results for both object and movement types. The algorithm created a window for sixty data points and chose the most frequent class among all the classes and whenever the change occurs in either type the DESC signals were generated. That prevented generating the wrong stimuli due to misclassification errors. The Haptuators were placed on each arm to increase ability to discriminate the vibrotactile feedback.

We found no significant difference between the performances of this study and previous study [12]. Our study is an excellent example of how psychophysical models can be established once (the models that we used were taken from the previous study [12]) and then reused easily by only measuring the thresholds using the adaptive tracking method for the same person. Thus in real-life applications, users can simply measure the thresholds whenever needed and continue to use the prosthesis without any problem. The previous study showed that vibrotactile feedback using discrete events are perceivable with reasonable accuracy by the user and our results confirm this study [12]. Remarkably, participants were able to identify correct sequences almost half of the time in both studies and they were mostly confused by similar sequences. This performance was quite good compared to the 7% chance level. Note that this results are for the full sequences. So that if the participant could not get all the sequences correctly, the answer is considered wrong. Because of this situation these scores might seem low, however if we calculate two discrete events out of three we reach 65% and if we calculate one event out of three we reach 80% accuracy. So that this performance scores are considered to be enough for the prosthesis use and in the literature it was shown that DESC based vibrotactile feedback improves the use of prosthesis better than no feedback or continuous feedback [23, 22, 21]. In addition, after some time, the performance of the user will improve for getting all the sequences correctly, because they will learn better after spending some considerable time using the prosthesis with

our system. This system can be tested on long term in the future and the results can be compared to our study.

When we imagine a real life situation of grasping and lifting a glass; the user would first needs place the hand around the glass. Then the user has to start closing the fingers (flexion in the air), touch the glass (contact to hard object) and in order to lift the glass ,the hand has to grip firmly, therefore, the fingers have to close little more after contacting the glass (flexion in hard object), this will help the user to easily grasp and lift the object without any slip. If the prosthesis keeps closing, the pressure on force sensor will increase (force increases), so that the user will know to stop the flexion before breaking the glass. Finally, the user has to lift the glass by applying right amount of force and the user has to keep that force while lifting. This is where another advantage of our system comes. If the prosthesis mistakenly opens or closes the hand, our system can detect that event right after it changes and give to the user an appropriate feedback. Although, there are several studies to prevent object slippage by using automatic grip controllers [164, 165, 166, 168, 178, 179], unintentionally generated EMG signals that gives flexion or extension commands, could result the object to drop or crush. With our system, the user will feel the movement and prevent unwanted situations. In our experiments, if one of these events were misclassified, the whole sequence was considered wrong. That is the reason the performance score for whole sequence is medium. However, the user can grab and lift the glass even with that classification error in real life. That is why performance score of our study is acceptable. When we apply feedback for these discrete events, the user will be in control of every step just as the real life and the prosthesis will finally seem to the user as a part of the body. Because, as mentioned in Section 1, without sensory feedback, the patients tend to abandon the prosthesis or use a cosmetic one. One reason is they can not own the prosthesis as their missing limb [14, 15].

In this thesis study, we successfully implemented two machine learning algorithms to a low memory FPGA card and applied vibrotactile feedback in real time. We successfully mapped the sensor data to discrete vibrotactile events. We used two frequencies and two magnitude values to map the events. The results showed that the performance of a prosthetic user can be improved with this system and reduce device abandonment. We present the detailed information and steps to apply DESC based

vibrotactile feedback by real time classification of sensor data and we showed some possible future applications that can be made on a real prosthetic hand. We hope that this system can be used in future applications and make prosthetic users' life easier.

APPENDIX A. ADDITIONAL FIGURES AND TABLES

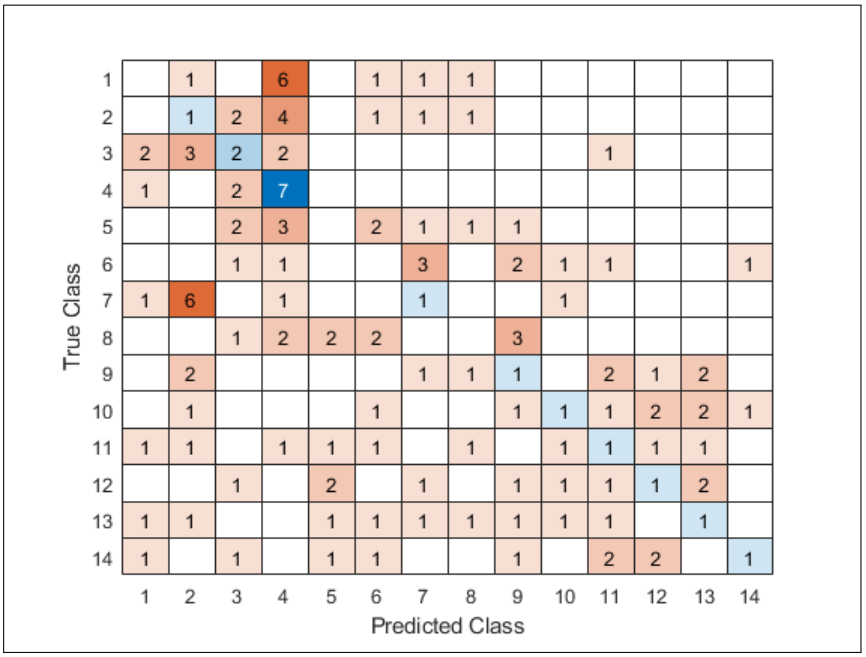

Figure A.1 Confusion matrix of the responses of S1 in DESC experiment.

Table A.1
Performance scores of S1 in DESC events.

Sequence	Precision	Recall	F1 score
1	0	0	0
2	0	7.14	8.33
3	10	16.67	18.18
4	20	22.22	34.28
5	75	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	11.11	9.09	10
10	11.11	16.67	13.33
11	20	20	20
12	9.09	14.29	11.11
13	9.09	12.50	10.52
14	10	33.33	15.38

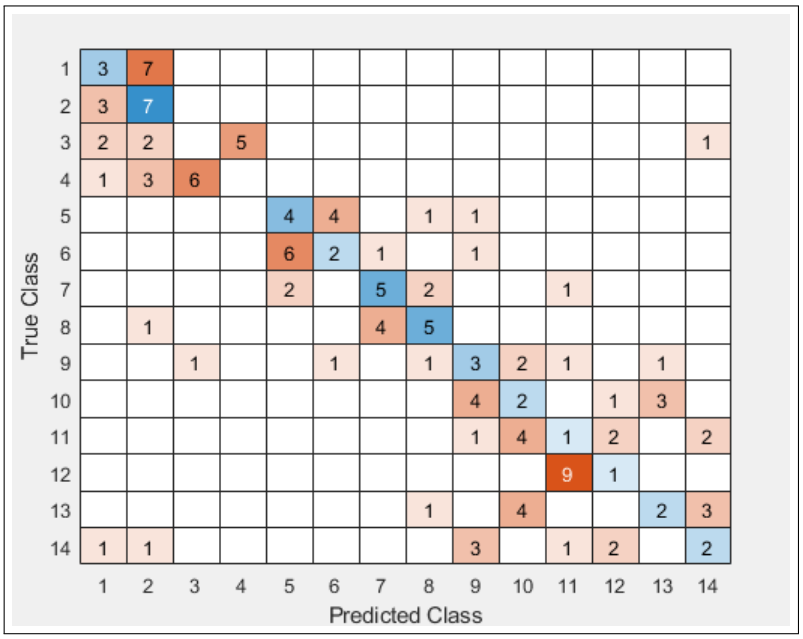

Figure A.2 Confusion matrix of the responses of S2 in DESC experiment.

Table A.2
Performance scores of S2 in DESC events.

Sequence	Precision	Recall	F1 score
1	33.33	40	36.36
2	70	33.33	45.13
3	0	0	0
4	0	0	0
5	50	50	50
6	20	28.57	23.53
7	62.5	50	55.56
8	50	50	50
9	30	23.08	26.09
10	20	16.67	18.18
11	10	7.7	8.70
12	20	16.67	12.50
13	20	33.30	24.99
14	20	25	22.22

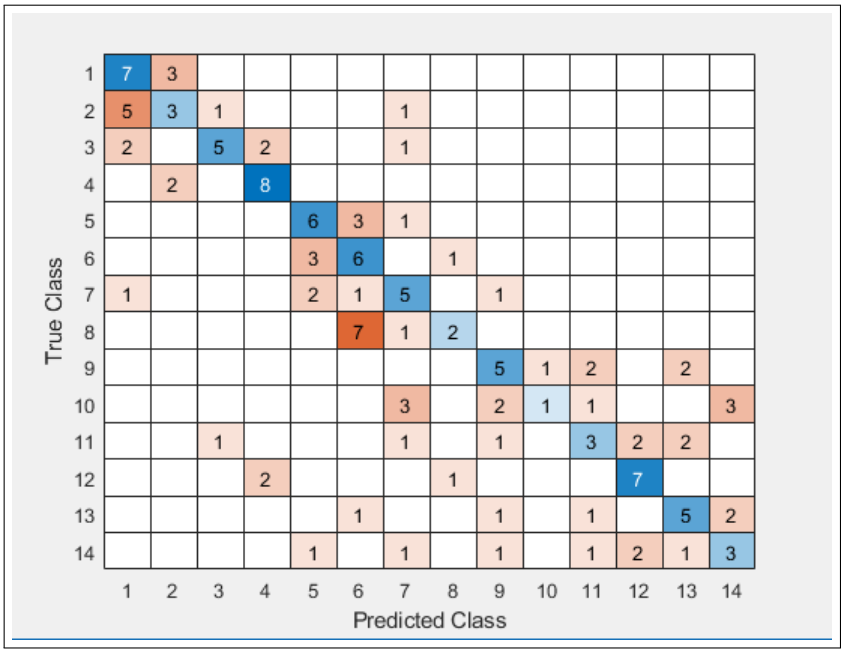

Figure A.3 Confusion matrix of the responses of S3 in DESC experiment.

Table A.3
Performance scores of S3 in DESC events.

Sequence	Precision	Recall	F1 score
1	70	46.67	56
2	30	37.50	33.33
3	50	71.43	58.82
4	80	66.67	72.73
5	60	50	54.55
6	60	33.3	42.83
7	50	35.71	41.66
8	20	50	28.57
9	50	45.46	47.62
10	10	50	16.67
11	33.3	37.5	35.28
12	70	63.64	66.67
13	56.54	60	58.22
14	30	37.5	33.33

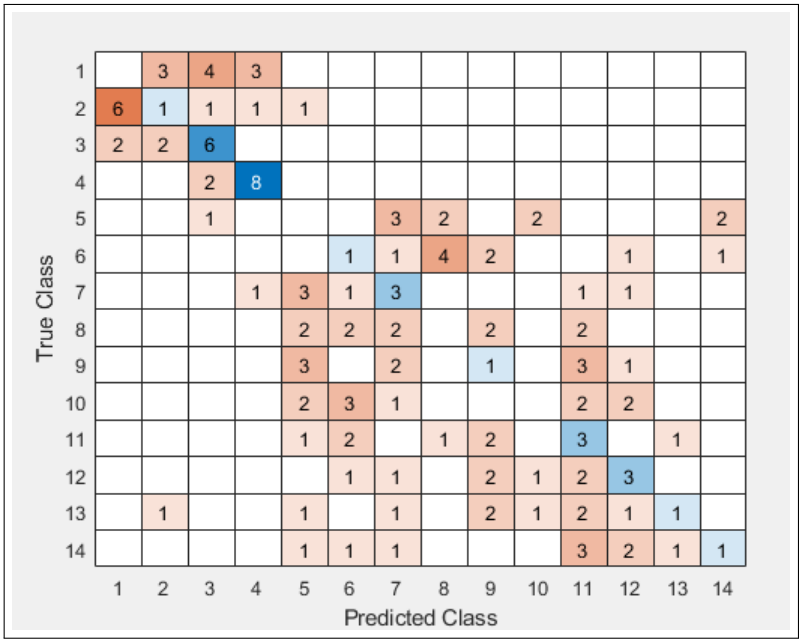

Figure A.4 Confusion matrix of the responses of S4 in DESC experiment.

Table A.4
Performance scores of S4 in DESC events.

Sequence	Precision	Recall	F1 score
1	0	0	0
2	10	14.29	11.77
3	60	42.86	50
4	77.78	53.85	63.64
5	0	0	0
6	8.33	9.10	8.70
7	33.33	20	25
8	0	0	0
9	10	9.10	9.53
10	0	0	0
11	30	16.67	21.43
12	30	27.27	28.57
13	10	33.30	15.38
14	9.09	25	13.33

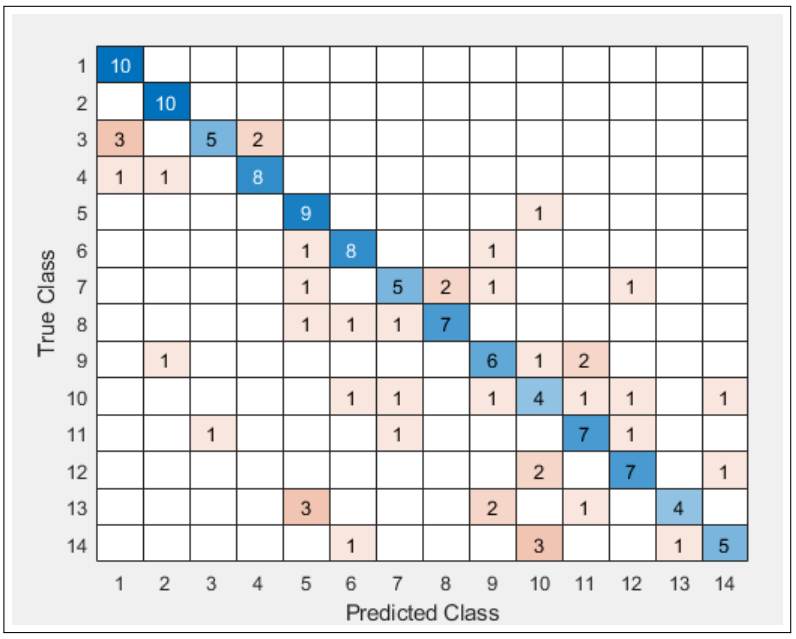

Figure A.5 Confusion matrix of the responses of S5 in DESC experiment.

Table A.5
Performance scores of S5 in DESC events.

Sequence	Precision	Recall	F1 score
1	100	71.43	83.33
2	100	83.33	90.91
3	50	83.33	62.50
4	80	80	80
5	100	60	75
6	80	72.73	76.19
7	50	62.50	55.56
8	70	77.78	73.69
9	60	54.55	57.15
10	40	40	40
11	70	63.64	66.67
12	70	70	70
13	40	80	53.33
14	50	71.43	58.82

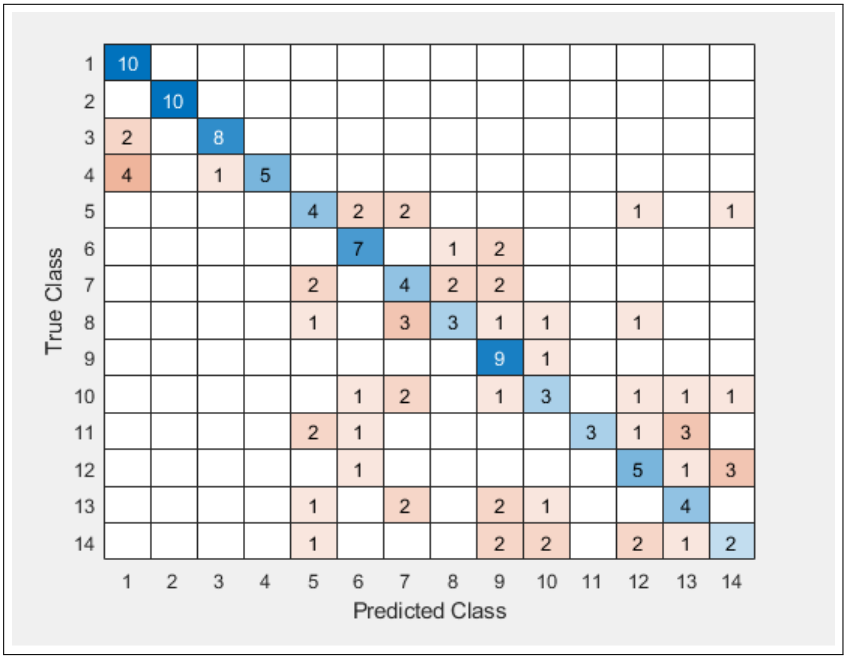

Figure A.6 Confusion matrix of the responses of S6 in DESC experiment.

Table A.6
Performance scores of S6 in DESC events.

Sequence	Precision	Recall	F1 score
1	100	62.5	76.92
2	100	100	100
3	80	88.89	84.21
4	50	100	66.67
5	40	36.36	38.09
6	70	58.33	63.63
7	40	30.77	34.78
8	30	50	37.50
9	90	47.37	62.07
10	30	37.5	33.33
11	30	100	46.15
12	50	45.46	47.62
13	40	40	40
14	20	28.57	23.53

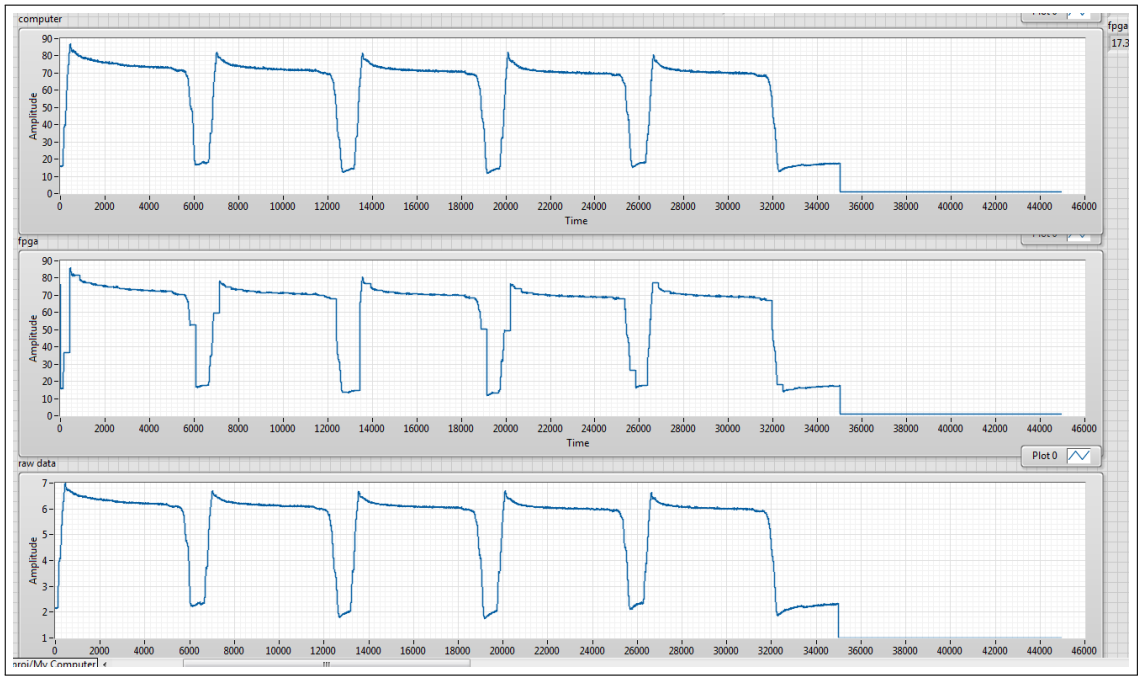

Figure A.7 Example of the calibrated MCP data. In the first graph, the data calibration was done in MATLAB for comparison with FPGA. In the second graph, the data calibration was done in FPGA. The raw data is shown in the third graph. The dataset were taken from [25].

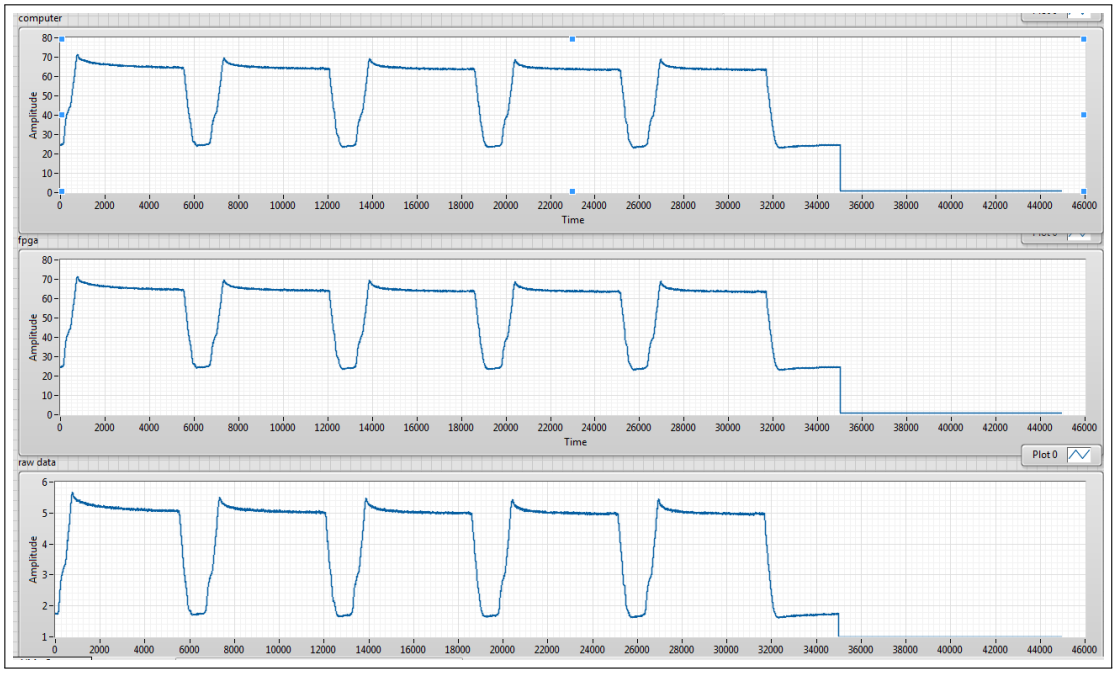

Figure A.8 Example of the calibrated DIP data. In the first graph, the data calibration was done in MATLAB for comparison with FPGA card. In the second graph, the data calibration was done in FPGA card. The raw data is shown in the third graph. The dataset were taken from [25].

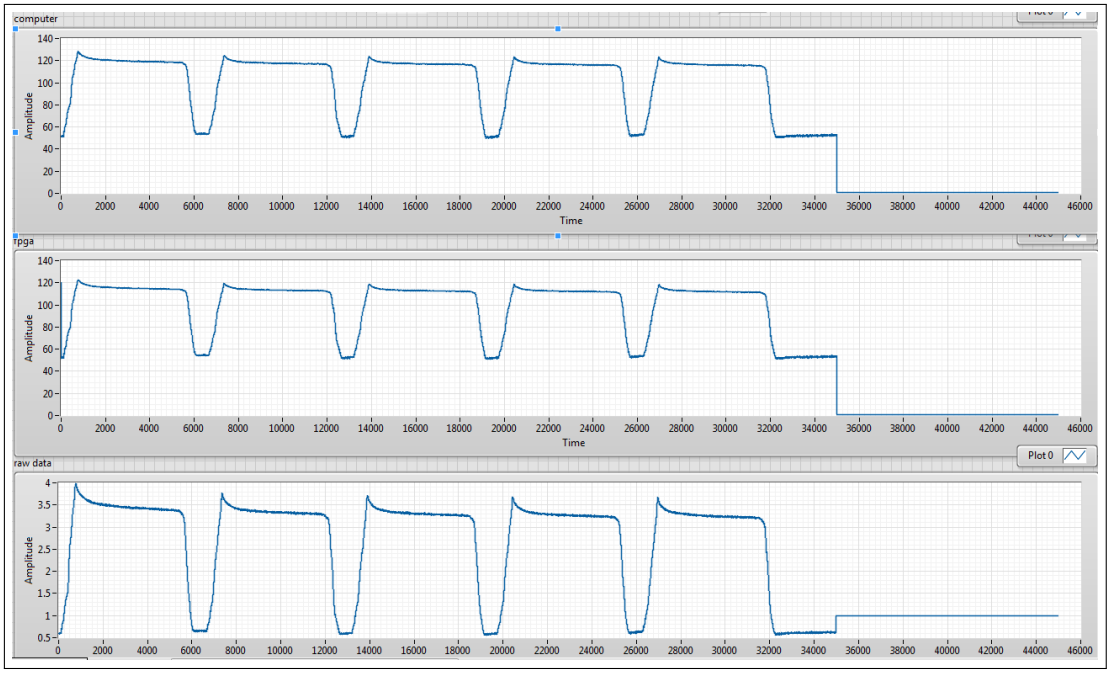

Figure A.9 Example of the calibrated PIP data. In the first graph, the data calibration was done in MATLAB for comparison with FPGA card. In the second graph, the data calibration was done in FPGA card. The raw data is shown in the third graph. The dataset were taken from [25].

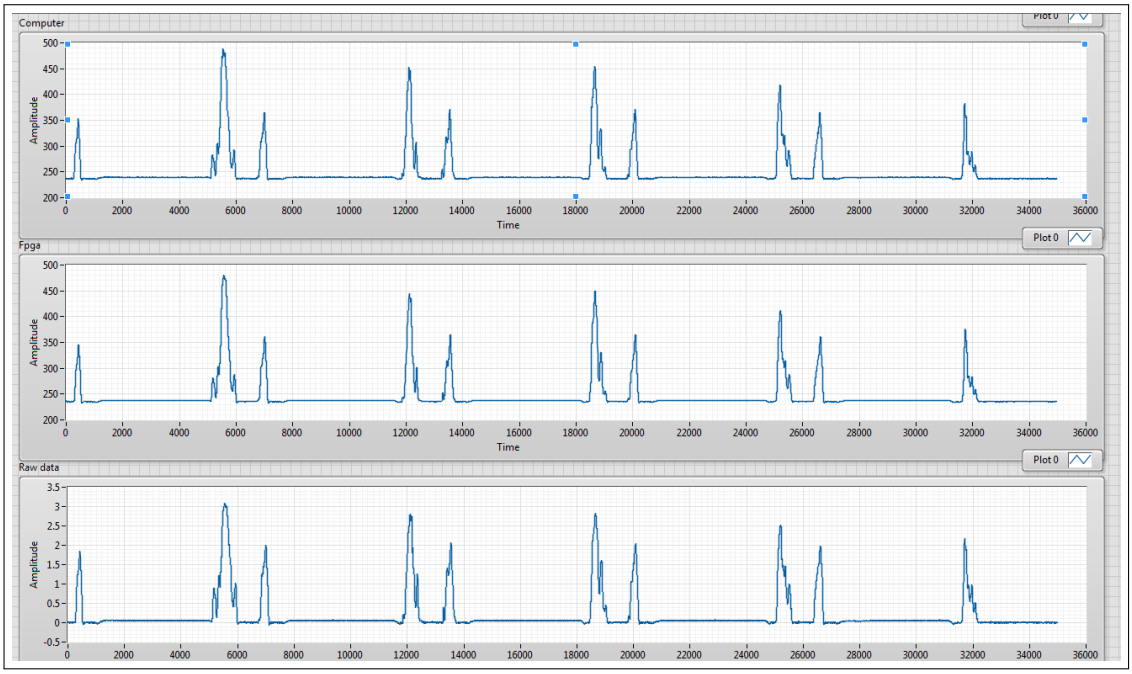

Figure A.10 Example of the calibrated PAL data. In the first graph, the data calibration was done in MATLAB for comparison with FPGA card. In the second graph, the data calibration was done in FPGA card. The raw data is shown in the third graph. The dataset were taken from [25].

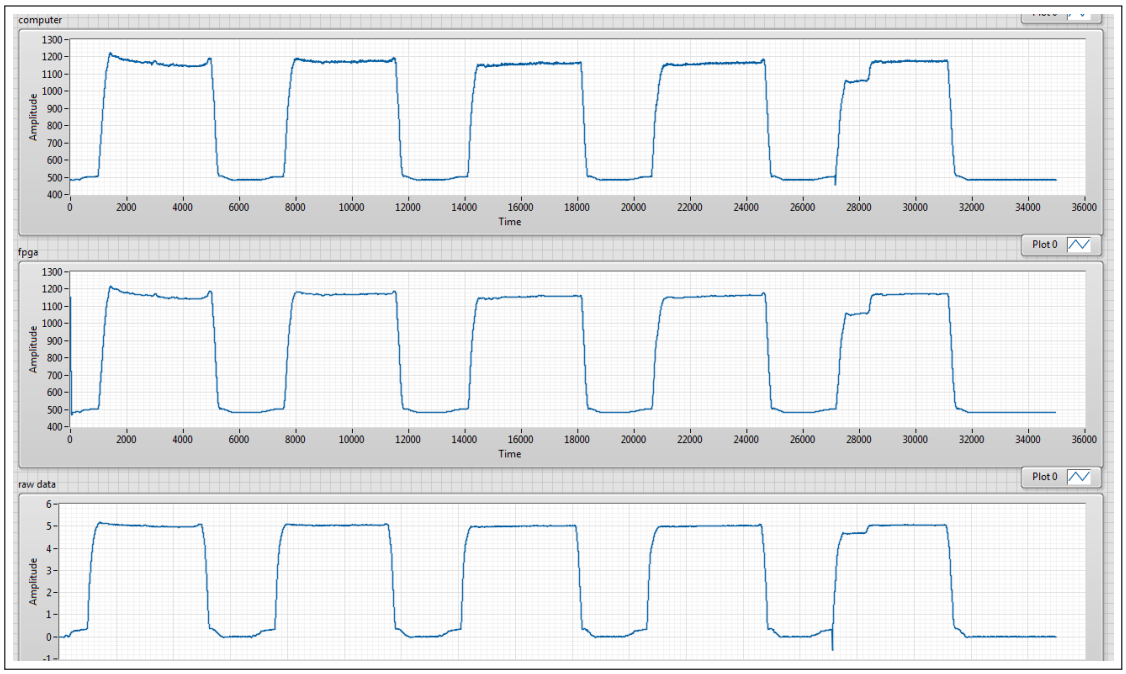

Figure A.11 Example of the calibrated DIST data. In the first graph, the data calibration was done in MATLAB for comparison with FPGA card. In the second graph, the data calibration was done in FPGA card. The raw data is shown in the third graph. The dataset were taken from [25].

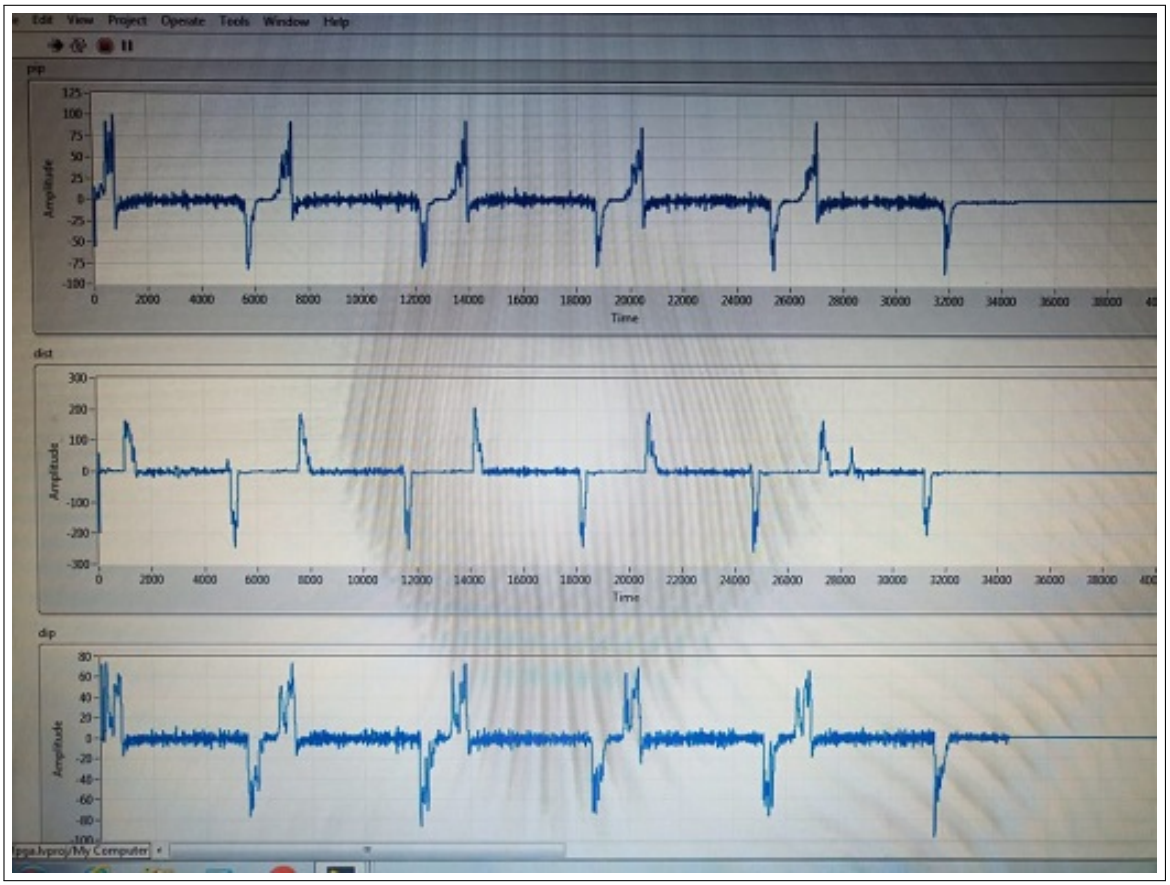

Figure A.12 The graphs of derivatives of the calibrated sensor data for PIP,DIST and DIP. The first graph shows the derivative of the PIP. The second graph shows the derivative of the DIST. The third graph shows the derivative of the DIP. The dataset were taken from [25].

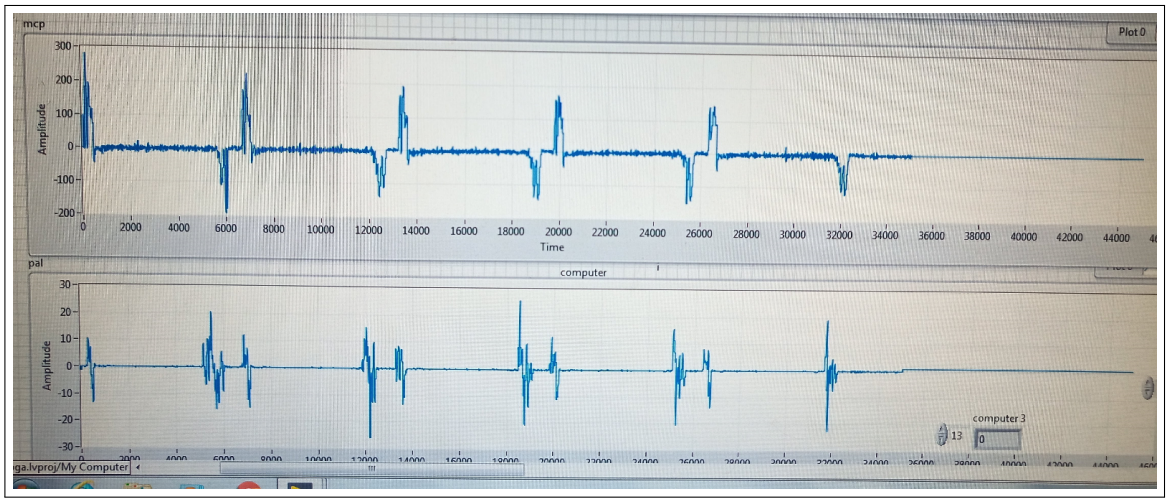

Figure A.13 The graphs of derivatives of the calibrated sensor data for MCP and PAL. The first graph shows the derivative of the MCP. The second graph shows the derivative of the PAL. The dataset were taken from [25].

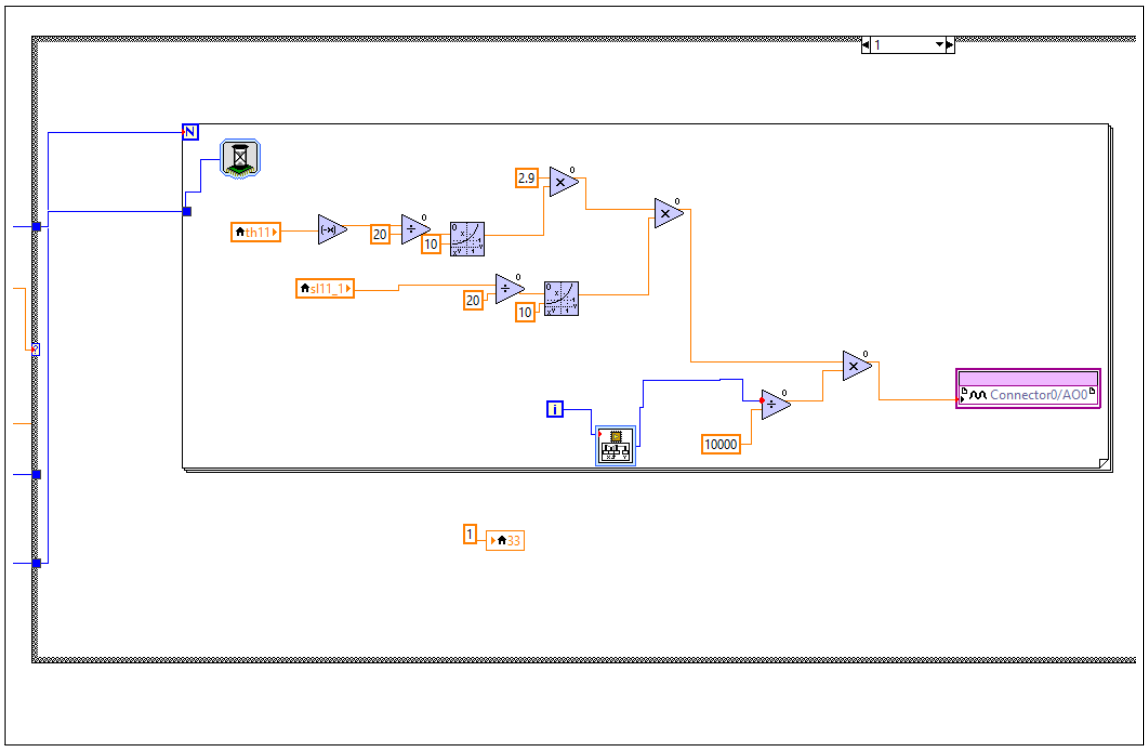

Figure A.14 Signal Generation part of the code. The code generates the signal by getting the classification results. The algorithm checks for discrete events and generates the signal accordingly. The amplitudes of the signals were calculated using the psychophysical models of each participant. The Look-up Table was used to generate the signal.

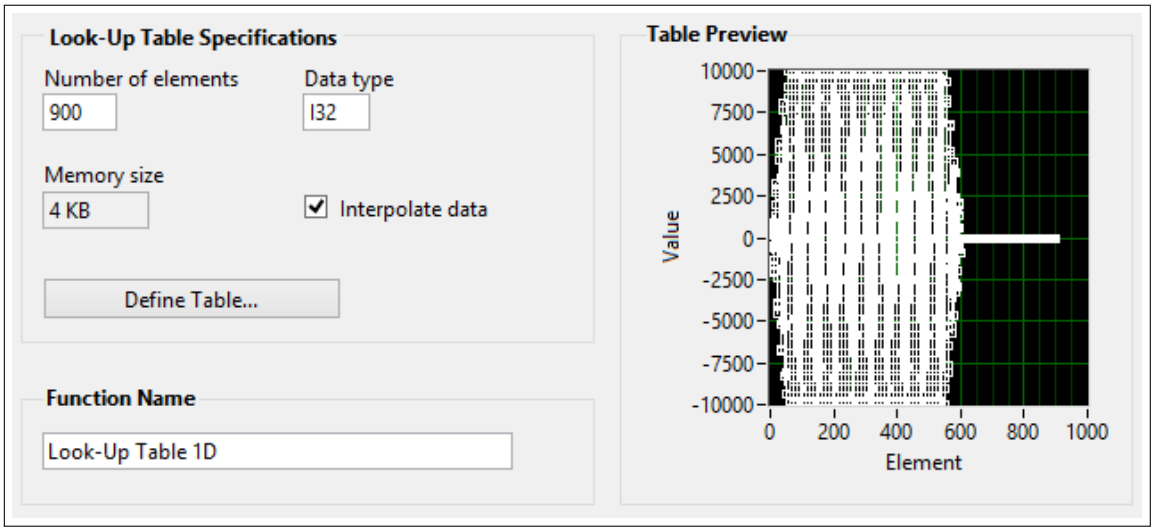

Figure A.15 LUT of the 180Hz signal. The signal was created in MATLAB and imported into the FPGA card using LUT's.

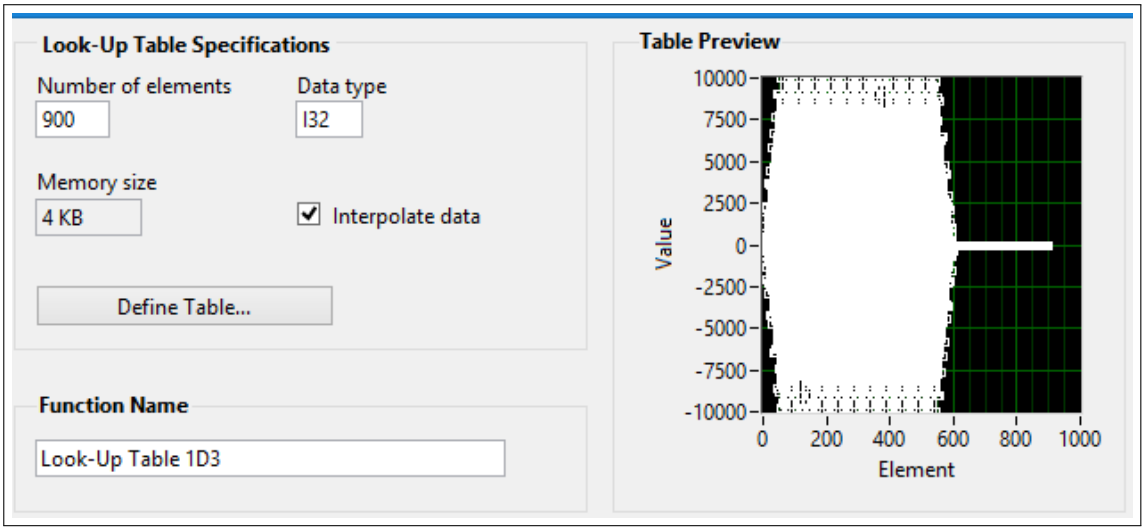

Figure A.16 LUT of the 80Hz signal. The signal was created in MATLAB and imported into the FPGA card using LUT's.

APPENDIX B. LIST OF PUBLICATIONS PRODUCED FROM THE THESIS

1. FPGA Implementation of Multinomial Logistic Regression For Vibrotactile Feedback In a Robotic Hand. Erbaş, İ., D.A. Vargas, and B. Güçlü. in 2020 International Conference on e-Health and Bioengineering (EHB). 2020. IEEE.

REFERENCES

1. Kandel, E. R., J. H. Schwartz, T. M. Jessell, *et al.*, *Principles of Neural Science*, Vol. 4, McGraw-hill New York, 2000.

2. Bear, M., B. Connors, and M. A. Paradiso, *Neuroscience: Exploring The Brain*, Jones & Bartlett Learning, LLC, 2020.

3. Gescheider, G. A., S. Bolanowski, K. Hall, *et al.*, "The effects of aging on information-processing channels in the sense of touch: I. absolute sensitivity," *Somatosensory & Motor Research*, Vol. 11, no. 4, pp. 345–357, 1994.

4. Verrillo, R. T., "Vibrotactile thresholds for hairy skin," *The Journal of Experimental Psychology*, Vol. 72, no. 1, pp. 47–50, 1966.

5. Makous, J. C., G. A. Gescheider, and S. J. Bolanowski, "The effects of static indentation on vibrotactile threshold," *The Journal of the Acoustical Society of America*, Vol. 99, no. 5, pp. 3149–3153, 1996.

6. Kapit, W., L. M. Elson, and L. M. Elson, *The Anatomy Coloring Book*, Harper & Row New York, 1977.

7. Alpaydin, E., *Introduction to Machine Learning*, MIT Press, 2020.

8. Aldair, D. A., "Fpga based modified fuzzy pid controller for pitch angle of bench-top helicopter," *Iraq Journal of Electrical and Electronic Engineering*, Vol. 81, p. 13, 2012.

9. Intruments, N., "User manual ni usb-7845r r series for usb multifunction rio with kintex-7 70t fpga, 2014."

10. Symbol, S., "Resistive flex sensors," *Datasheet*, 2014.

11. Electronics, I., "Fsr 400-short," *P/N: 94-00010 Rev. A Datasheet.*

12. Karakuş, p., and B. Güçlü, "Psychophysical principles of discrete event-driven vibrotactile feedback for prostheses," *Somatosensory & Motor Research*, Vol. 37, no. 3, pp. 186–203, 2020.

13. İpek Karakuş, *Psychophysical Evaluation Of A Sensory Feedback System For Proshetic Hands.* PhD thesis, Bogazici University, Istanbul, Turkey, 2020.

14. Biddiss, E., and T. Chau, "Upper-limb prosthetics: critical factors in device abandonment," *American Journal of Physical Medicine & Rehabilitation*, Vol. 86, no. 12, pp. 977–87, 2007.

15. Schofield, J. S., K. R. Evans, J. P. Carey, *et al.*, "Applications of sensory feedback in motorized upper extremity prosthesis: a review," *Expert Review of Medical Devices*, Vol. 11, no. 5, pp. 499–511, 2014.

16. Lake, C., and R. Dodson, "Progressive upper limb prosthetics," *Physical Medicine And Rehabilitation Clinics of North America*, Vol. 17, no. 1, pp. 49–72, 2006.

17. Bouwsema, H., C. K. van der Sluis, and R. M. Bongers, "Effect of feedback during virtual training of grip force control with a myoelectric prosthesis," *PLoS One*, Vol. 9, no. 5, p. e98301, 2014.

18. Botvinick, M., and J. Cohen, "Rubber hands 'feel' touch that eyes see," *Nature*, Vol. 391, no. 6669, pp. 756–756, 1998.

19. Lundborg, G., and B. Rosen, "Sensory substitution in prosthetics," *Hand Clinics*, Vol. 17, no. 3, pp. 481–488, 2001.

20. Dietrich, C., K. Walter-Walsh, and S. o. Preißler, "Sensory feedback prosthesis reduces phantom limb pain: proof of a principle," *Neuroscience Letters*, Vol. 507, no. 2, pp. 97–100, 2012.

21. Clemente, F., M. D. Alonzo, and M. a. o. Controzzi, "Non-invasive, temporally discrete feedback of object contact and release improves grasp control of closed-loop myoelectric transradial prostheses," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, Vol. 24, no. 12, pp. 1314–1322, 2016.

22. Cipriani, C., J. L. Segil, F. Clemente, *et al.*, "Humans can integrate feedback of discrete events in their sensorimotor control of a robotic hand," *Experimental Brain Research*, Vol. 232, no. 11, pp. 3421–3429, 2014.

23. Aboseria, M., F. Clemente, L. F. Engels, *et al.*, "Discrete vibro-tactile feedback prevents object slippage in hand prostheses more intuitively than other modalities," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, Vol. 26, no. 8, pp. 1577–1584, 2018.

24. Crossley, J., *Functional Exercise and Rehabilitation: The Neuroscience of Movement, Pain and Performance*, Routledge, 2021.

25. Karakuş, p., H. Şahin, A. Atasoy, *et al.*, "Evaluation of sensory feedback from a robotic hand: A preliminary study," in *Haptics: Science, Technology, and Applications*, pp. 452–463, Springer International Publishing.

26. Johnson, R., J. van der Linden, and Y. Rogers, "Musicjacket: the efficacy of real-time vibrotactile feedback for learning to play the violin," *Extended Abstracts on Human Factors in Computing Systems*, pp. 3475–3480, 04 2010.

27. Gopalai, A. A., and S. A. A. Senanayake, "A wearable real-time intelligent posture corrective system using vibrotactile feedback," *IEEE/ASME Transactions on Mechatronics*, Vol. 16, no. 5, pp. 827–834, 2011.

28. Van Der Linden, J., R. Johnson, J. Bird, and otherss, "Buzzing to play: lessons learned from an in the wild study of real-time vibrotactile feedback," in *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*, pp. 533–542.

29. Agur, A. M., and A. F. Daley, *Grant's Atlas of Anatomy*, 2009.

30. Vallbo, A., and R. Johansson, "Properties of cutaneous mechanoreceptors in the human hand related to touch sensation," *Human Neurobiology*, Vol. 3, pp. 3–14, 02 1984.

31. Dudkiewicz, I., R. Gabrielov, I. Seiv-Ner, *et al.*, "Evaluation of prosthetic usage in upper limb amputees," *Disability And Rehabilitation*, Vol. 26, no. 1, pp. 60–3, 2004.

32. Maat, B., G. Smit, D. Plettenburg, and P. Breedveld, "Passive prosthetic hands and tools: A literature review," *Prosthetics and Orthotics International*, Vol. 42, no. 1, pp. 66–74, 2018.

33. Vujaklija, I., D. Farina, and O. C. Aszmann, "New developments in prosthetic arm systems," *Orthopedic Research and Reviews*, Vol. 8, p. 31, 2016.

34. Cupo, M. E., and S. J. Sheredos, "Clinical evaluation of a new, above-elbow, body-powered prosthetic arm: a final report," *Journal of Rehabilitation Research And Development*, Vol. 35, no. 4, pp. 431–460, 1998.

35. Smit, G., D. H. Plettenburg, and F. C. van der Helm, "Design and evaluation of two different finger concepts for body-powered prosthetic hand," *Journal of Rehabilitation Research & Development*, Vol. 50, no. 9, pp. 1253–1266, 2013.

36. McGovern, A., R. Lagerquist, and D. J. o. Gagne, "Making the black box more transparent: Understanding the physical implications of machine learning," *Bulletin of the American Meteorological Society*, Vol. 100, no. 11, pp. 2175–2199, 2019.

37. Dalley, S. A., H. A. Varol, and M. Goldfarb, "A method for the control of multigrasp myoelectric prosthetic hands," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, Vol. 20, no. 1, pp. 58–67, 2011.

38. Geethanjali, P., "Myoelectric control of prosthetic hands: state-of-the-art review," *Medical Devices (Auckland, NZ)*, Vol. 9, p. 247, 2016.

39. Webster, J., *Medical Instrumentation Application and Design, 4th Edition*, John Wiley and Sons, Incorporated, 2009.

40. Merrill, D. R., J. Lockhart, P. R. Troyk, *et al.*, "Development of an implantable myoelectric sensor for advanced prosthesis control," *Artif Organs*, Vol. 35, no. 3, pp. 249–52, 2011.

41. Pasquina, P. F., M. Evangelista, A. J. Carvalho, *et al.*, "First-in-man demonstration of a fully implanted myoelectric sensors system to control an advanced electromechanical prosthetic hand," *The Journal of Neuroscience Methods*, Vol. 244, pp. 85–93, 2015.

42. Carey, S. L., P. M. Stevens, and M. J. Highsmith, "Differences in myoelectric and body-powered upper-limb prostheses: systematic literature review update 2013–2016," *JPO: Journal of Prosthetics and Orthotics*, Vol. 29, no. 4S, pp. P17–P20, 2017.

43. Dhillon, G. S., and K. W. Horch, "Direct neural sensory feedback and control of a prosthetic arm," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, Vol. 13, no. 4, pp. 468–72, 2005.

44. Micera, S., J. Carpaneto, and S. Raspopovic, "Control of hand prostheses using peripheral information," *IEEE Reviews in Biomedical Engineering*, Vol. 3, pp. 48–68, 2010.

45. Raspopovic, S., M. Capogrosso, F. M. Petrini, *et al.*, "Restoring natural sensory feedback in real-time bidirectional hand prostheses," *Science Translational Medicine*, Vol. 6, no. 222, pp. 222ra19–222ra19, 2014.

46. Ghafoor, U., S. Kim, and K.-S. Hong, "Selectivity and longevity of peripheral-nerve and machine interfaces: a review," *Frontiers in Neurorobotics*, Vol. 11, p. 59, 2017.

47. Kaczmarek, K. A., M. E. Tyler, A. J. Brisben, *et al.*, "The afferent neural response to electrotactile stimuli: preliminary results," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, Vol. 8, no. 2, pp. 268–70, 2000.

48. Li, K., P. Boyd, Y. Zhou, *et al.*, "Electrotactile feedback in a virtual hand rehabilitation platform: Evaluation and implementation," *IEEE Transactions on Automation Science and Engineering*, Vol. 16, no. 4, pp. 1556–1565, 2018.

49. Dosen, S., M. C. Schaeffer, and D. Farina, "Time-division multiplexing for myoelectric closed-loop control using electrotactile feedback," *Journal of NeuroEngineering and Rehabilitation*, Vol. 11, p. 138, 2014.

50. Antfolk, C., M. D'Alonzo, B. Rosén, *et al.*, "Sensory feedback in upper limb prosthetics," *Expert Review of Medical Devices*, Vol. 10, no. 1, pp. 45–54, 2013.

51. Antfolk, C., A. Björkman, S. O. Frank, *et al.*, "Sensory feedback from a prosthetic hand based on air-mediated pressure from the hand to the forearm skin," *Journal of Rehabilitation Medicine*, Vol. 44, no. 8, pp. 702–7, 2012.

52. Kaczmarek, K. A., J. G. Webster, P. Bach-y Rita, *et al.*, "Electrotactile and vibrotactile displays for sensory substitution systems," *IEEE Transactions on Biomedical Engineering*, Vol. 38, no. 1, pp. 1–16, 1991.

53. Yildiz, M. Z., . Toker, F. B. Özkan, *et al.*, "Effects of passive and active movement on vibrotactile detection thresholds of the pacinian channel and forward masking," *Somatosensory and Motor Research*, Vol. 32, no. 4, pp. 262–72, 2015.

54. Cipriani, C., M. D'Alonzo, and M. C. Carrozza, "A miniature vibrotactile sensory substitution device for multifingered hand prosthetics," *IEEE Transactions on Biomedical Engineering*, Vol. 59, no. 2, pp. 400–408, 2012.

55. Svensson, P., U. Wijk, A. Björkman, *et al.*, "A review of invasive and non-invasive sensory feedback in upper limb prostheses," *Expert Review of Medical Devices*, Vol. 14, no. 6, pp. 439–447, 2017.

56. Stephens-Fripp, B., G. Alici, and R. Mutlu, "A review of non-invasive sensory feedback methods for transradial prosthetic hands," *IEEE Access*, Vol. 6, pp. 6878–6899, 2018.

57. Gonzelman, J., H. Ellis, and O. Clayton, "Prosthetic device sensory attachment," *US Patent Specification*, 1953.

58. Pylatiuk, C., A. Kargov, and S. Schulz, "Design and evaluation of a low-cost force feedback system for myoelectric prosthetic hands," *JPO: Journal of Prosthetics and Orthotics*, Vol. 18, no. 2, pp. 57–61, 2006.

59. Chatterjee, A., P. Chaubey, J. Martin, and N. Thakor, "Testing a prosthetic haptic feedback simulator with an interactive force matching task," *JPO: Journal of Prosthetics and Orthotics*, Vol. 20, no. 2, pp. 27–34, 2008.

60. Saunders, I., and S. Vijayakumar, "The role of feed-forward and feedback processes for closed-loop prosthesis control," *Journal of Neuroengineering and Rehabilitation*, Vol. 8, no. 1, pp. 1–12, 2011.

61. Witteveen, H. J., F. Luft, J. S. Rietman, *et al.*, "Stiffness feedback for myoelectric forearm prostheses using vibrotactile stimulation," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, Vol. 22, no. 1, pp. 53–61, 2014.

62. Mann, R. W., and S. D. Reimers, "Kinesthetic sensing for the emg controlled "boston arm"," *IEEE Transactions on Man-Machine Systems*, Vol. 11, no. 1, pp. 110–115, 1970.

63. Patterson, P. E., and J. A. Katz, "Design and evaluation of a sensory feedback system that provides grasping pressure in a myoelectric hand," *Journal of Rehabilitation Research & Development*, Vol. 29, no. 1, pp. 1–8, 1992.

64. Poveda, A. R., "Myoelectric prostheses with sensorial feedback," *University of New Brunswick's MyoElectric Controls/Powered Prosthetics Symposium*, pp. 73–80, 2002.

65. Johansson, R. S., and J. R. Flanagan, "Coding and use of tactile signals from the fingertips in object manipulation tasks," *Nature Reviews Neuroscience*, Vol. 10, no. 5, pp. 345–359, 2009.

66. Johansson, R. S., and B. B. Edin, "Predictive feed-forward sensory control during grasping and manipulation in man," *Biomedical Research*, Vol. 14, pp. 95–95, 1993.

67. Flanagan, J. R., M. C. Bowman, and R. S. Johansson, "Control strategies in object manipulation tasks," *Current Opinion in Neurobiology*, Vol. 16, no. 6, pp. 650–659, 2006.

68. Nascimento, A. I., F. M. Mar, *et al.*, "The intriguing nature of dorsal root ganglion neurons: Linking structure with polarity and function," *Progress in Neurobiology*, Vol. 168, pp. 86–103, 2018.

69. Bell, J., S. Bolanowski, and M. H. Holmes, "The structure and function of pacinian corpuscles: A review," *Progress in Neurobiology*, Vol. 42, no. 1, pp. 79–128, 1994.

70. Bolanowski Jr, S. J., G. A. Gescheider, R. T. Verrillo, *et al.*, "Four channels mediate the mechanical aspects of touch," *The Journal of The Acoustical Society of America*, Vol. 84, no. 5, pp. 1680–1694, 1988.

71. Gescheider, G. A., J. H. Wright, and R. T. Verrillo, *Information-Processing Channels in The Tactile Sensory System: A Psychophysical and Physiological Analysis*, Psychology Press, 2010.

72. Verrillo, R. T., and S. J. Bolanowski Jr, "The effects of skin temperature on the psychophysical responses to vibration on glabrous and hairy skin," *The Journal of the Acoustical Society of America*, Vol. 80, no. 2, pp. 528–532, 1986.

73. Gescheider, G. A., *Psychophysics: The Fundamentals*, Psychology Press, 2013.

74. Gescheider, G. A., B. Güçlü, J. L. Sexton, *et al.*, "Spatial summation in the tactile sensory system: probability summation and neural integration," *Somatosensory Motor Research*, Vol. 22, no. 4, pp. 255–68, 2005.

75. Bolanowski, S. J., G. A. Gescheider, and R. T. Verrillo, "Hairy skin: psychophysical channels and their physiological substrates," *Somatosensory & Motor Research*, Vol. 11, no. 3, pp. 279–290, 1994.

76. Verrillo, R. T., and G. A. Gescheider, "Enhancement and summation in the perception of two successive vibrotactile stimuli," *Perception & Psychophysics*, Vol. 18, no. 2, pp. 128–136, 1975.

77. Gescheider, G., and N. Migel, "Some temporal parameters in vibrotactile forward masking," *The Journal of the Acoustical Society of America*, Vol. 98, no. 6, pp. 3195–3199, 1995.

78. Cholewiak, R. W., and J. C. Craig, "Vibrotactile pattern recognition and discrimination at several body sites," *Perception & Psychophysics*, Vol. 35, no. 6, pp. 503–514, 1984.

79. Lowry, R., and G. Fechner, *Elements of Psychophysics.*, Wiley Online Library, 1967.

80. Prins, N., *et al.*, *Psychophysics: A Practical Introduction*, Academic Press, 2016.

81. Fechner, G., *Elements of Psychophysics.*, New York, 1966.

82. Zwislocki, J. J., and E. M. Relkin, "On a psychophysical transformed-rule up and down method converging on a 75% level of correct responses," *Proceedings of the National Academy of Sciences*, Vol. 98, no. 8, pp. 4811–4814, 2001.

83. Schwarz, R. J., and C. Taylor, "The anatomy and mechanics of the human hand," *Artificial Limbs*, Vol. 2, no. 2, pp. 22–35, 1955.

84. Bishop, C. M., *Pattern Recognition and Machine Learning*, Springer, 2006.

85. Fakoor, R., F. Ladhak, A. Nazi, and M. Huber, "Using deep learning to enhance cancer diagnosis and classification," *Proceedings of The International Conference on Machine Learning*, Vol. 28, pp. 3937–3949, 2013.

86. Mccarthy, J. F., K. A. Marx, P. E. Hoffman, *et al.*, "Applications of machine learning and high-dimensional visualization in cancer detection, diagnosis, and management," *Annals of the New York Academy of Sciences*, Vol. 1020, no. 1, pp. 239–262, 2004.

87. Zeng, Y., S. Xu, W. C. Chapman, *et al.*, "Real-time colorectal cancer diagnosis using pr-oct with deep learning," *Theranostics*, Vol. 10, no. 6, pp. 2587–2596, 2020.

88. Abdar, M., W. Książek, U. R. Acharya, *et al.*, "A new machine learning technique for an accurate diagnosis of coronary artery disease," *Computer Methods and Programs in Biomedicine*, Vol. 179, p. 104992, 2019.

89. Ajemba, P. O., L. Ramirez, and N. G. o. Durdle, "A support vectors classifier approach to predicting the risk of progression of adolescent idiopathic scoliosis," *IEEE Transactions on Information Technology in Biomedicine*, Vol. 9, no. 2, pp. 276–282, 2005.

90. Alfaro-Ponce, M., I. Chairez, and R. Etienne-Cummings, "Automatic detection of electrocardiographic arrhythmias by parallel continuous neural networks implemented in fpga," *Neural Computing and Applications*, Vol. 31, no. 2, pp. 363–375, 2019.

91. Azar, A. T., "A novel anfis application for prediction of post–dialysis blood urea concentration," *International Journal of Intelligent Systems Technologies and Applications*, Vol. 12, no. 2, pp. 87–110, 2013.

92. Dawes, T. J., A. de Marvao, W. Shi, *et al.*, "Machine learning of three-dimensional right ventricular motion enables outcome prediction in pulmonary hypertension: a cardiac mr imaging study," *Radiology*, Vol. 283, no. 2, pp. 381–390, 2017.

93. Kavakiotis, I., O. Tsave, A. Salifoglou, *et al.*, "Machine learning and data mining methods in diabetes research," *Computational and Structural Biotechnology Journal*, Vol. 15, pp. 104–116, 2017.

94. Rajpurkar, P., J. Irvin, R. L. Ball, *et al.*, "Deep learning for chest radiograph diagnosis: A retrospective comparison of the chexnext algorithm to practicing radiologists," *PLoS Medicine*, Vol. 15, no. 11, p. e1002686, 2018.

95. Edwards, A. L., M. R. Dawson, J. S. Hebert, *et al.*, "Application of real-time machine learning to myoelectric prosthesis control: A case series in adaptive switching," *Prosthetics and Orthotics International*, Vol. 40, no. 5, pp. 573–81, 2016.

96. Swami, C. P., N. Lenhard, and J. Kang, "A novel framework for designing a multi-dof prosthetic wrist control using machine learning," *Scientific Reports*, Vol. 11, no. 1, p. 15050, 2021.

97. Gibson, A. E., M. R. Ison, and P. Artemiadis, "User-independent hand motion classification with electromyography," in *Dynamic Systems and Control Conference*, Vol. 56130, p. V002T26A002, American Society of Mechanical Engineers, 2013.

98. Suchodolski, T., and A. Wolczowski, "Hand prosthesis control-software tool for emg signal analysis.," in *International Conference on Informatics in Control, Automation and Robotics*, pp. 321–326, 2010.

99. Wolczowski, A., and M. Kurzynski, "Control of hand prosthesis using fusion of biosignals and information from prosthesis sensors," in *Computational Intelligence and Efficiency in Engineering Systems*, pp. 259–273, Springer, 2015.

100. Parker, A. S., A. L. Edwards, and P. M. Pilarski, "Exploring the impact of machine-learned predictions on feedback from an artificial limb," in *2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR)*, pp. 1239–1246, IEEE, 2019.

101. Mazilu, S., M. Hardegger, Z. Zhu, *et al.*, "Online detection of freezing of gait with smartphones and machine learning techniques," in *2012 6th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*, pp. 123–130, IEEE, 2012.

102. Huang, H., C. Enz, M. Grambone, *et al.*, "Data fusion for a hand prosthesis tactile feedback system," in *2014 IEEE Biomedical Circuits and Systems Conference (BioCAS) Proceedings*, pp. 604–607, IEEE, 2014.

103. Hosmer Jr, D. W., S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, Vol. 398, John Wiley and Sons, 2013.

104. James, G., D. Witten, T. Hastie, *et al.*, *An Introduction To Statistical Learning*, Vol. 112, Springer, 2013.

105. MATLAB, "Multinomial logistic regression," Available: https://www.mathworks.com/help/stats/mnrfit.html.

106. Zhang, A., Z. C. Lipton, M. Li, *et al.*, *Dive Into Deep Learning*, 2021.

107. Williams, R., "Multinomial logit models–overview," *University of Notre Dame, Last revised February*, Vol. 13, p. 2017, 2017.

108. Bonaccorso, G., *Machine Learning Algorithms*, Packt Publishing Ltd, 2017.

109. Kingsford, C., and S. L. Salzberg, "What are decision trees?," *Nature Biotechnology*, Vol. 26, no. 9, pp. 1011–1013, 2008.

110. Rokach, L., and O. Maimon, "Decision trees," in *Data Mining and Knowledge Discovery Handbook*, pp. 165–192, Springer, 2005.

111. Russell, S., and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2002.

112. Brown, G., "Ensemble learning," *Encyclopedia of Machine Learning*, Vol. 312, pp. 15–19.

113. Zhou, Z.-H., "Ensemble learning," *Encyclopedia of Biometrics*, pp. 270–273, 01 2009.

114. Shannon, C. E., "A mathematical theory of communication," *The Bell System Technical Journal*, Vol. 27, no. 3, pp. 379–423, 1948.

115. Gini, C., *Variabilitá e Mutabilitá, Con-Tributo Allo Studio Delle Distribuzioni: Relazioni Statistische*, 1912.

116. Brodley, C. E., and P. E. Utgoff, "Multivariate decision trees," *Machine Learning*, Vol. 19, no. 1, pp. 45–77, 1995.

117. Chrysos, G., P. Dagritzikos, I. Papaefstathiou, and A. Dollas, "Hc-cart: A parallel system implementation of data mining classification and regression tree (cart) algorithm on a multi-fpga system," *ACM Transactions on Architecture and Code Optimization (TACO)*, Vol. 9, no. 4, pp. 1–25, 2013.

118. Ikeda, T., K. Sakurada, A. Nakamura, *et al.*, "Hardware/algorithm co-optimization for fully-parallelized compact decision tree ensembles on fpgas," in *Applied Reconfigurable Computing. Architectures, Tools, and Applications*, pp. 345–357, Springer International Publishing.

119. Kulaga, R., and M. Gorgon, "Fpga implementation of decision trees and tree ensembles for character recognition in vivado hls," *Image Processing & Communications*, Vol. 19, no. 2-3, p. 71, 2014.

120. Narayanan, R., D. Honbo, G. Memik, *et al.*, "An fpga implementation of decision tree classification," in *2007 Design, Automation & Test in Europe Conference & Exhibition*, pp. 1–6.

121. Oberg, J., K. Eguro, R. Bittner, *et al.*, "Random decision tree body part recognition using fpgas," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*, pp. 330–337.

122. Saadeh, W., F. H. Khan, and M. A. B. Altaf, "Design and implementation of a machine learning based eeg processor for accurate estimation of depth of anesthesia," *IEEE Transactions on Biomedical Circuits and Systems*, Vol. 13, no. 4, pp. 658–669, 2019.

123. Song, W., Q. Han, Z. Lin, *et al.*, "Design of a flexible wearable smart semg recorder integrated gradient boosting decision tree based hand gesture recognition," *IEEE Transactions on Biomedical Circuits and Systems*, Vol. 13, no. 6, pp. 1563–1574, 2019.

124. Peng, W., J. Chen, and H. Zhou, "An implementation of id3-decision tree learning algorithm," *School of Computer Science & Engineering, Sydney, NSW*, Vol. 13, pp. 1–20, 2009.

125. Breiman, L., J. H. Friedman, R. A. Olshen, *et al.*, *Classification and Regression Trees*, Routledge, 2017.

126. Loh, W.-Y., and N. Vanichsetakul, "Tree-structured classification via generalized discriminant analysis," *Journal of the American Statistical Association*, Vol. 83, no. 403, pp. 715–725, 1988.

127. Polat, K., and S. Güneş, "A novel hybrid intelligent method based on c4. 5 decision tree classifier and one-against-all approach for multi-class classification problems," *Expert Systems with Applications*, Vol. 36, no. 2, pp. 1587–1592, 2009.

128. Trimberger, S. M., *Field-Programmable Gate Array Technology*, Springer Science & Business Media, 2012.

129. Erbaş, I., D. A. Vargas, and B. Güçlü, "Fpga implementation of multinomial logistic regression for vibrotactile feedback in a robotic hand," in *2020 International Conference on e-Health and Bioengineering (EHB)*, pp. 1–4, IEEE.

130. Muthuramalingam, A., S. Himavathi, and E. Srinivasan, "Neural network implementation using fpga: issues and application," *International Journal of Information Technology*, Vol. 4, no. 2, pp. 86–92, 2008.

131. Óscar, O.-P., J. A. Hidalgo-López, *et al.*, "Fpga-based tactile sensor suite electronics for real-time embedded processing," *IEEE Transactions on Industrial Electronics*, Vol. 64, no. 12, pp. 9657–9665, 2017.

132. Óscar, O.-P., J. A. Hidalgo-López, and J. A. Sánchez-Durán, "Architecture of a tactile sensor suite for artificial hands based on fpgas," in *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pp. 112–117.

133. Rodríguez-Andina, J. J., M. D. Valdes-Pena, and M. J. Moure, "Advanced features and industrial applications of fpgas—a review," *IEEE Transactions on Industrial Informatics*, Vol. 11, no. 4, pp. 853–864, 2015.

134. Sm, S., "Field programmable gate arrays and their applications," *International Journal of Electrical and Electronic Engineering and Telecommunications*, Vol. 3, pp. 19–25, 01 2014.

135. Stanchieri, G. D. P., M. Saleh, M. Sciulli, *et al.*, "Fpga-based tactile sensory feedback system with optical fiber data communication link for prosthetic applications," in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 374–377.

136. Brown, S. D., R. J. Francis, J. Rose, *et al.*, *Field-Programmable Gate Arrays*, Vol. 180, Springer Science & Business Media, 1992.

137. Liu, J., and D. Liang, "A survey of fpga-based hardware implementation of anns," in *2005 International Conference on Neural Networks and Brain*, Vol. 2, pp. 915–918, IEEE.

138. Instruments, N., "Fpga fundementals," Available: https://www.ni.com/en-tr/innovations/white-papers/ 08/fpga-fundamentals.html.

139. Williston, K., *Digital Signal Processing: World Class Designs*, Newnes, 2009.

140. Krips, M., T. Lammert, and A. Kummert, "Fpga implementation of a neural network for a real-time hand tracking system," in *Proceedings First IEEE International Workshop on Electronic Design, Test and Applications '2002*, pp. 313–317.

141. Page, A., C. Sagedy, E. Smith, *et al.*, "A flexible multichannel eeg feature extractor and classifier for seizure detection," *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 62, no. 2, pp. 109–113, 2014.

142. Boschmann, A., A. Agne, L. Witschen, *et al.*, "Fpga-based acceleration of high density myoelectric signal processing," in *2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pp. 1–8.

143. Fejer, A., Z. Nagy, J. Benois-Pineau, *et al.*, "Fpga-based sift implementation for wearable computing," in *2019 IEEE 22nd International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, pp. 1–4, IEEE.

144. Instruments, N., "Labview fpga floating-point library," Available: http://sine.ni.com/nips/cds/ view/p/lang/en/nid/216788.

145. McCullagh, P., *Generalized Linear Models*, New York: Chapman & Hall, 1990.

146. Pedregosa, F., G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, Vol. 12, 01 2012.

147. Yildiz, M. Z., and B. Güçlü, "Relationship between vibrotactile detection threshold in the pacinian channel and complex mechanical modulus of the human glabrous skin," *Somatosensory and Motor Research*, Vol. 30, no. 1, pp. 37–47, 2013.

148. Güçlü, B., and S. J. Bolanowski, "Vibrotactile thresholds of the non-pacinian i channel: I. methodological issues," *Somatosensory Motor Research*, Vol. 22, no. 1-2, pp. 49–56, 2005.

149. IBM Corp. Released 2020. IBM SPSS Statistics for Windows, Version 27.0. Armonk, N. I. C.

150. Wobbrock, J. O., L. Findlater, D. Gergle, *et al.*, "The aligned rank transform for non-parametric factorial analyses using only anova procedures," *Proceedings of The SIGCHI Conference On Human Factors In Computing Systems*, pp. 143–146, 2011.

151. Instruments, N., "Labview fpga module help," Available: https://zone.ni.com/reference/en-XX/help/371599P-01.

152. Alhammami, M., O. C. Pun, and T. W. Haw, "Hardware/software co-design for accelerating human action recognition," in *2015 IEEE Conference on Sustainable Utilization And Development In Engineering and Technology (CSUDET)*, pp. 1–5, IEEE.

153. Pittman, N., A. Forin, A. Criminisi, *et al.*, "Image segmentation using hardware forest classifiers," in *2013 IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines*, pp. 73–80.

154. Engels, L. F., A. W. Shehata, E. J. Scheme, *et al.*, "When less is more–discrete tactile feedback dominates continuous audio biofeedback in the integrated percept while controlling a myoelectric prosthetic hand," *Frontiers In Neuroscience*, Vol. 13, p. 578, 2019.

155. Hanif, N. H. H. M., N. N. N. Hashim, P. H. Chappell, *et al.*, "Tactile to vibrotactile sensory feedback interface for prosthethic hand users," in *2016 IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, pp. 326–330.

156. Hasson, C. J., and J. Manczurowsky, "Effects of kinematic vibrotactile feedback on learning to control a virtual prosthetic arm," *Journal of NeuroEngineering and Rehabilitation*, Vol. 12, no. 1, p. 31, 2015.

157. Mastinu, E., L. F. Engels, F. Clemente, *et al.*, "Neural feedback strategies to improve grasping coordination in neuromusculoskeletal prostheses," *Scientific Reports*, Vol. 10, no. 1, p. 11793, 2020.

158. Instruments, N., "Using the single-precision floating-point data type," Available: https://zone.ni.com/reference/en-XX/help/371599P-01/.

159. Güçlü, B., G. A. Gescheider, S. J. Bolanowski, *et al.*, "Population-response model for vibrotactile spatial summation," *Somatosensory Motor Research*, Vol. 22, no. 4, pp. 239–53, 2005.

160. Okamura, A. M., "Haptic feedback in robot-assisted minimally invasive surgery," *Current Opinion in Urology*, Vol. 19, no. 1, p. 102, 2009.

161. Saracino, A., A. Deguet, F. Staderini, *et al.*, "Haptic feedback in the da vinci research kit (dvrk): a user study based on grasping, palpation, and incision tasks," *The International Journal of Medical Robotics and Computer Assisted Surgery*, Vol. 15, no. 4, p. e1999, 2019.

162. Ibrahim, A., L. Pinna, and M. Valle, "Interface circuits based on fpga for tactile sensor systems," in *2017 New Generation of CAS (NGCAS)*, pp. 37–40.

163. Siciliano, B., O. Khatib, and T. Kröger, *Springer Handbook of Robotics*, Vol. 200, Springer, 2008.

164. Pasluosta, C., H. Tims, A. Chiu, *et al.*, "Slippage sensory feedback and nonlinear force control system for a low-cost prosthetic hand," *American Journal of Biomedical Science and Research*, Vol. 1, no. 4, pp. 295–302, 2009.

165. Light, C., P. Chappell, B. Hudgins, *et al.*, "Intelligent multifunction myoelectric control of hand prostheses," *Journal of Medical Engineering & Technology*, Vol. 26, no. 4, pp. 139–146, 2002.

166. Gentile, C., F. Cordella, C. R. Rodrigues, *et al.*, "Touch-and-slippage detection algorithm for prosthetic hands," *Mechatronics*, Vol. 70, p. 102402, 2020.

167. Golz, S., C. Osendorfer, and S. Haddadin, "Using tactile sensation for learning contact knowledge: Discriminate collision from physical interaction," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3788–3794, 2015.

168. Veer, K., and T. Sharma, "A novel feature extraction for robust emg pattern recognition.," *Journal of Medical Engineering & Technology*, Vol. 40, no. 4, pp. 149–154, 2016.

169. Agriomallos, I., S. Doltsinis, I. Mitsioni, *et al.*, "Slippage detection generalizing to grasping of unknown objects using machine learning with novel features," *IEEE Robotics and Automation Letters*, Vol. 3, no. 2, pp. 942–948, 2018.

170. Boretius, T., J. Badia, A. Pascual-Font, *et al.*, "A transverse intrafascicular multichannel electrode (time) to interface with the peripheral nerve," *Biosensors and Bioelectronics*, Vol. 26, no. 1, pp. 62–69, 2010.

171. Dhillon, G. S., S. M. Lawrence, D. T. Hutchinson, *et al.*, "Residual function in peripheral nerve stumps of amputees: implications for neural control of artificial limbs," *The Journal of Hand Surgery*, Vol. 29, no. 4, pp. 605–615, 2004.

172. Dhillon, G., T. Kruger, J. Sandhu, *et al.*, "Effects of short-term training on sensory and motor function in severed nerves of long-term human amputees," *Journal of Neurophysiology*, Vol. 93, no. 5, pp. 2625–2633, 2005.

173. Horch, K., S. Meek, T. G. Taylor, *et al.*, "Object discrimination with an artificial hand using electrical stimulation of peripheral tactile and proprioceptive pathways with intrafascicular electrodes," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, Vol. 19, no. 5, pp. 483–489, 2011.

174. Valle, G., E. D'Anna, I. Strauss, *et al.*, "Hand control with invasive feedback is not impaired by increased cognitive load," *Frontiers in Bioengineering and Biotechnology*, Vol. 8, p. 287, 2020.

175. Roggen, D., A. Calatroni, M. Rossi, *et al.*, "Collecting complex activity datasets in highly rich networked sensor environments," in *2010 Seventh International Conference on Networked Sensing Systems (INSS)*, pp. 233–240, 2010.

176. Cordella, F., A. L. Ciancio, R. Sacchetti, *et al.*, "Literature review on needs of upper limb prosthesis users," *Frontiers in Neuroscience*, Vol. 10, p. 209, 2016.

177. Lacey, G., G. W. Taylor, and S. Areibi, *Deep Learning on Fpgas: Past, Present, and Future*, 2016.

178. Tomovic, R., and G. Boni, "An adaptive artificial hand," *IRE Transactions on Automatic Control*, Vol. 7, no. 3, pp. 3–10, 1962.

179. Kyberd, P. J., O. E. Holland, P. H. Chappell, *et al.*, "Marcus: A two degree of freedom hand prosthesis with hierarchical grip control," *IEEE Transactions on Rehabilitation Engineering*, Vol. 3, no. 1, pp. 70–76, 1995.