Fused Mechanomyography and Inertial Measurement for Human-Robot Interface

by

Samuel Charles Wilson

Department of Mechanical Engineering

Supervised by

Doctor Ravi Vaidyanathan and Professor Alison McGregor

A dissertation submitted to Imperial College London in accordance with the requirements for award of the degree *Doctor of Philosophy*.

February 2020

Abstract

Human-Machine Interfaces (HMI) are the technology through which we interact with the ever-increasing quantity of smart devices surrounding us. The fundamental goal of an HMI is to facilitate robot control through uniting a human operator as the supervisor with a machine as the task executor. Sensors, actuators, and onboard intelligence have not reached the point where robotic manipulators may function with complete autonomy and therefore some form of HMI is still necessary in unstructured environments. These may include environments where direct human action is undesirable or infeasible, and situations where a robot must assist and/or interface with people. Contemporary literature has introduced concepts such as body-worn mechanical devices, instrumented gloves, inertial or electromagnetic motion tracking sensors on the arms, head, or legs, electroencephalographic (EEG) brain activity sensors, electromyographic (EMG) muscular activity sensors and camera-based (vision) interfaces to recognize hand gestures and/or track arm motions for assessment of operator intent and generation of robotic control signals. While these developments offer a wealth of future potential their utility has been largely restricted to laboratory demonstrations in controlled environments due to issues such as lack of portability and robustness and an inability to extract operator intent for both arm and hand motion.

Wearable physiological sensors hold particular promise for capture of human intent/command. EMG-based gesture recognition systems in particular have received significant attention in recent literature. As wearable pervasive devices, they offer benefits over camera or physical input systems in that they neither inhibit the user physically nor constrain the user to a location where the sensors are deployed. Despite these benefits, EMG alone has yet to demonstrate the capacity to recognize both gross movement (e.g. arm motion) and finer grasping (e.g. hand movement). As such, many researchers have proposed fusing muscle activity (EMG) and motion tracking e.g. (inertial measurement) to combine arm motion and grasp intent as HMI input for manipulator control. However, such work has arguably reached a plateau since EMG suffers from interference from environmental factors which cause signal degradation over time, demands an electrical connection with the skin, and has not demonstrated the capacity to function out of controlled environments for long periods of time.

This thesis proposes a new form of gesture-based interface utilising a novel combination of inertial measurement units (IMUs) and mechanomyography sensors (MMGs). The modular system permits numerous configurations of IMU to derive body kinematics in real-time and uses this to convert arm movements into control signals. Additionally, bands containing six mechanomyography sensors were used to observe muscular contractions in the forearm which are generated using specific hand motions. This combination of continuous and discrete control signals allows a large variety of smart devices to be controlled.

Several methods of pattern recognition were implemented to provide accurate decoding of the mechanomyographic information, including Linear Discriminant Analysis and Support Vector Machines. Based on these techniques, accuracies of 94.5% and 94.6% respectively were achieved for 12 gesture classification. In real-time tests, accuracies of 95.6% were achieved in 5 gesture classification.

It has previously been noted that MMG sensors are susceptible to motion induced interference. The thesis also established that arm pose also changes the measured signal. This thesis introduces a new method of fusing of IMU and MMG to provide a classification that is robust to both of these sources of interference.

Additionally, an improvement in orientation estimation, and a new orientation estimation algorithm are proposed. These improvements to the robustness of the system provide the first solution that is able to reliably track both motion and muscle activity for extended periods of time for HMI outside a clinical environment.

Application in robot teleoperation in both real-world and virtual environments were explored. With multiple degrees of freedom, robot teleoperation provides an ideal test platform for HMI devices, since it requires a combination of continuous and discrete control signals. The field of prosthetics also represents a unique challenge for HMI applications. In an ideal situation, the sensor suite should be capable of detecting the muscular activity in the residual limb which is naturally indicative of intent to perform a specific hand pose and trigger this post in the prosthetic device. Dynamic environmental conditions within a socket such as skin impedance have delayed the translation of gesture control systems into prosthetic devices, however mechanomyography sensors are unaffected by such issues.

There is huge potential for a system like this to be utilised as a controller as ubiquitous computing systems become more prevalent, and as the desire for a simple, universal interface increases. Such systems have the potential to impact significantly on the quality of life of prosthetic users and others.

Acknowledgements

I would like to express my thanks to Dr. Ravi Vaidyanathan and Professor Alison McGregor for giving me the opportunity to pursue this work and the advice and guidance to complete it.

I would also like to extend my thanks to Alex Lewis for providing the direction and encouragement to take this work further than I ever envisioned.

Also to the members of the Biomechatronics Lab and its alumni: Chris, Enrico, James, Lewis, Matt, Marcel, Marcus and Paolo, as well as all those who joined us for their assistance and for creating an environment in which it was a pleasure to work.

I extend my gratitude to the volunteers who participated in my studies, and whose feedback was always valuable in choosing a path forward.

Finally, I would like to express my deepest gratitude to my family, to Marcus, Catherine and to Diana.

Declaration of Originality

I, Samuel Charles Wilson, certify that all material in this dissertation which is not my own work has been duly acknowledged.

Copyright Declaration

©The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives license. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the license terms of this work.

Dedication

I couldn’t have got this far without you.

Thank you for your hard work and support.

You know who you are.

“The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.”

Mark Weiser

Contents

Abstract 2
Acknowledgements 4
Declaration of Originality 6
Copyright Declaration 6
Dedication 8
List of Tables 16
List of Figures 16
Nomenclature 18
Abbreviations 18
Symbols 19
Chapter 1 22
1.1 Project Background 23
1.2 Motivation 23
1.3 Aim 24
1.3.1 Smart Systems 24
1.3.2 Controlling Upper-Limb Prosthetic Devices 25
1.3.3 Robot Teleoperation 26
1.4 Contributions 26
1.5 Publications 27
1.5.1 Journal Papers 27
1.5.2 Conference Papers 27
1.6 Layout of Report 28
Chapter 2 30
2.1 Structure of the Literature Review 31
2.2 Overview of Intelligent Human-Machine Interfaces and Sensing Technology 32
2.2.1 Introduction to Terms 32
2.2.2 Traditional HMI 32
2.3 Amputees and their prosthetics 33
2.3.1 Causes of Amputation 33
2.3.2 Prosthetic Devices 34

2.3.3 Assistive HMIs 36
2.4 Sensing Modalities 37
2.4.1 Physical Input Systems 37
2.4.2 Vision-based systems 40
2.4.3 Inertial Measurement Units 41
2.4.4 Electromyography 42
2.4.5 Mechanomyography 45
2.4.6 Myokinemetric signals 46
2.4.7 Sonomyography 47
2.4.8 Other forms of interface 47
2.4.8.1 Gaze-based interface 47
2.4.8.2 Tongue-based interface 48
2.4.8.3 Brain Machine Interface 48
2.4.8.4 Voice 48
2.5 Guidelines for HMI Design 49
2.5.1 Functionality/Robustness 49
2.5.2 Usability/Comfort 51
2.5.3 Form/Pervasiveness 52
2.5.4 Accessibility/Cost 53
2.6 Overview of Orientation Estimation 56
2.6.1 Introduction to Quaternions 57
2.6.2 Gradient Descent Algorithm – In-depth Review 59
2.7 Chapter Summary 61
Chapter 3 64
3.1 Introduction to Hardware Development 65
3.1.1 Key User Requirements 65
3.2 NUIMU Development 66
3.2.1 Hardware Design 66
3.2.1.1 Microcontroller 67
3.2.1.2 Wireless Module 67
3.2.1.3 UART-USB Converter 67

3.2.1.4 Inertial Measurement Unit 67
3.2.1.5 Power Management 67
3.2.1.6 Auxiliary Ports 68
3.2.1.7 Manufacturing 68
3.2.1.8 Finalizing Hardware 68
3.2.2 Hardware Improvements 69
3.2.2.1 Improved Inertial Measurement Unit 69
3.3 Firmware 69
3.4 NUIMU Attributes 71
3.4.1 Other Applications 71
3.4.1.1 Secondary Inertial Measurement Unit 71
3.4.1.2 Prosthetic Hand Driver 73
3.4.1.3 Infrared Remote 73
3.5 Mechanomyography Sensor Design 74
3.6 Sensor Suite Evaluation 75
3.7 NUIMU Host Software 75
3.7.1 C# code for Windows (The NU Interface) 75
3.7.1.1 NUClass 76
3.7.1.2 UIClass 78
3.7.2 Android code for Mobiles (The NU App) 79
3.7.2.1 First Iteration (Android Studio) 80
3.7.2.2 Second Iteration (Unity) 80
3.8 Chapter Summary 80
Chapter 4 82
4.1 Initial Implementation of Orientation Estimation Algorithm 83
4.1.1 Sensor Bias Removal 83
4.1.2 Gyroscopic Calibration 83
4.1.3 Magnetometer Calibration 83
4.1.4 Comparison Against Known Orientation 85
4.2 Improving the Efficiency of the Algorithm 87
4.3 Gradient Descent Modifications 90

4.3.1 Motivation....90
4.3.2 Solution....90
4.3.3 Algorithm Convergence....91
4.3.4 Algorithm Assessment....92
4.3.5 Robustness to Gyroscopic Bias....96
4.4 New Algorithm Formulation....97
4.4.1 Motivation....97
4.4.2 Solution....97
4.4.3 Convergence Rate....101
4.4.4 Efficiency – Base Algorithm....102
4.5 Chapter Summary....102
Chapter 5....104
5.1 Activity Classification....105
5.2 Database Generation....106
5.2.1 Participants....106
5.2.2 Protocol....106
5.3 Gesture Segmentation....107
5.4 Gesture Classification....110
5.4.1 Template-based Classification....110
5.4.1.1 Offline Accuracy....111
5.4.1.2 Real-time Implementation....113
5.4.1.3 Real-time Accuracy....113
5.4.2 Machine Learning-based Classification....115
5.4.2.1 Algorithms....115
5.4.2.2 Experimental Comparison....116
5.4.2.3 Dimensionality Reduction....117
5.4.2.4 Comparison of Algorithms for Real-time Implementation....120
5.4.2.5 Real-time Implementation....122
5.4.2.6 Real-time Accuracy....122
5.5 Guidelines for Practical Use of MMGs....123
5.5.1 Ideal Number of MMGs....123

5.6 Chapter Summary 125
Chapter 6 126
6.1 Introduction 127
6.2 Real-world implementation for Robot Teleoperation 127
6.2.1 Experimental Protocol 127
6.2.2 Results 131
6.2.3 Discussion 131
6.3 Virtual Reality Environments 132
6.4 Hands Free Control within Virtual Environments. 135
6.4.1 Participants 135
6.4.2 Experimental Protocol 135
6.4.3 Results 137
6.4.4 Discussion 138
6.5 Prosthetic Control – Classification in Uncontrolled Environments 138
6.5.1 Participants 138
6.5.2 Experimental Protocol 139
6.5.3 Classification – One vs One 139
6.5.4 Classification – Voting 141
6.5.5 Classification – Many vs. One 141
6.5.6 Fused Mechanomyography and Inertial Measurement 143
6.5.7 Conclusion 144
6.6 Robot Teleoperation in Virtual Environments 144
6.6.1 Kuka Kinematic Derivation 145
6.6.2 Experimental Protocol 145
6.6.2.1 Participants 145
6.6.2.2 Training Protocol 145
6.6.2.3 Testing Protocol 146
6.6.3 Results 148
6.6.4 Discussion 150
6.7 Chapter Summary 151
Chapter 7 154

7.1 Thesis Summary....155
7.2 Summary of Contributions....157
7.3 Suggested Future Directions....158
7.4 Final Comments....159
References....160
Appendix I....178
Appendix II....182
Appendix III....186

List of Tables

TABLE 1 - RECENT EMG CLASSIFICATION STUDIES 45
TABLE 2 - RECENT MMG CLASSIFICATION STUDIES 46
TABLE 3 – DESCRIPTION OF NUIMU HARDWARE ATTRIBUTES 71
TABLE 4 – IMU ERRORS 87
TABLE 5 - CONFUSION MATRIX SHOWING CLASSIFICATION ACCURACIES OF TEMPLATE-BASED CLASSIFICATION FOR ALL NON-AMPUTEE SUBJECTS 112
TABLE 6 - CONFUSION MATRIX SHOWING CLASSIFICATION ACCURACIES OF TEMPLATE-BASED CLASSIFICATION FOR AMPUTEE SUBJECT 112
TABLE 7 - REAL-TIME SEGMENTATION ACCURACIES 114
TABLE 8 - REAL TIME CLASSIFICATION ACCURACIES USING TEMPLATE-BASED CLASSIFICATION 115
TABLE 9 - CONFUSION MATRIX OF CLASSIFICATION ACCURACIES USING SVM 116
TABLE 10 - CONFUSION MATRIX OF CLASSIFICATION ACCURACY FOR AMPUTEE SUBJECT USING SVM 116
TABLE 11 - COMPARISON OF DIFFERENT MACHINE LEARNING TECHNIQUES ON GESTURE GROUPS FOR NON-AMPUTEES (AMPUTEE) 117
TABLE 12 - EFFECT OF PCA ON CLASSIFICATION 118
TABLE 13 – ACCURACIES OF CLASSIFICATION USING EXTRACTED FEATURES 119
TABLE 14 - SUMMARY OF BEST PERFORMING MACHINE LEARNING METHODS FOR CLASSIFICATION 119
TABLE 15 - RELEVANT METRICS FOR CLASSIFIER COMPARISON 121
TABLE 16 – SUMMARY OF TOP PERFORMING CLASSIFIER METRICS 122
TABLE 17 - SEGMENTATION ACCURACY FOR REAL TIME MACHINE LEARNING IMPLEMENTATION 123
TABLE 18 - CLASSIFICATION ACCURACY USING LDA IN REAL TIME APPLICATION 123
TABLE 19 – AVERAGE ACCURACIES OF BAXTER ROBOT CONTROL 131
TABLE 20 – INDEX OF DIFFICULTIES FOR BAXTER TASKS 131
TABLE 21 – LDA ACCURACY WHEN TRAINED AND TESTED ON DIFFERENT POSITIONS 140
TABLE 22 – LINEAR SVM ACCURACY WHEN TRAINED AND TESTED ON DIFFERENT POSITIONS 140
TABLE 23 – CUBIC SVM ACCURACY WHEN TRAINED AND TESTED ON DIFFERENT POSITIONS 140
TABLE 24 – EXAMPLE OF PERFORMANCE OF VOTING CLASSIFIER 141
TABLE 25 – ACCURACIES OF CLASSIFIERS TRAINED ON ADJACENT POSITIONS 142
TABLE 26 – CLASSIFIER ACCURACY WHEN TRAINED ON REPRESENTATIVE DATASET SUBGROUPS 142
TABLE 27 - TIME TAKEN TO COMPLETE VR ROBOT TELEOPERATION EXPERIMENT 148
TABLE 28 – ACCURACY OF COMMANDS IN VR ROBOT TELEOPERATION EXPERIMENT 149
TABLE 29 – PRECISION OF COMMANDS IN VR ROBOT TELEOPERATION EXPERIMENT 149
TABLE 30 – MISCLASSIFICATIONS AND FALSE POSITIVES SEPARATED BY TASK 150
TABLE 31 – SUMMARY OF DENAVIT HARTENBERG PARAMETERS OF VIRTUAL KUKA ROBOT 187

List of Figures

FIGURE 1 - LAYOUT OF LITERATURE REVIEW 31
FIGURE 2 – COMPARISON OF FEATURES OF DIFFERENT SENSING MODALITIES 55
FIGURE 3 – BREAKDOWN OF IC ELEMENTS ON NUIMU BOARDS 66
FIGURE 4 – FINAL PACKAGE OF NUIMU HARDWARE 68
FIGURE 5 – UPDATED IMU ELEMENT ON NUIMU BOARD 69

FIGURE 6 – FLOWCHART DEMONSTRATING PROGRAM FLOW IN NUIMU....70
FIGURE 7 - NUIMU WITH PERIPHERAL IMU....72
FIGURE 8 - NUIMU USED AS A PROSTHETIC HAND DRIVER....72
FIGURE 9 – NUIMU AS AN INFRARED REPEATER....73
FIGURE 10 - MMG DESIGNS....74
FIGURE 11 - REAL-TIME DATA VISUALISATION....77
FIGURE 12 - MAIN INTERFACE WINDOW....78
FIGURE 13 -MAGNETOMETER OUTPUT (LEFT: RAW MAGNETOMETER OUTPUT. MID: CROPPED AND FILTERED. RIGHT: CALIBRATED OUTPUT)....84
FIGURE 14 - OPTOTRAK SYSTEM USED TO EVALUATE ORIENTATION ESTIMATION ALGORITHM....86
FIGURE 15 - EXPERIMENTAL SET UP FOR ORIENTATION ESTIMATION EVALUATION (A – IMU IN POSITION. B- IMU WITH BEACON AFFIXED)....86
FIGURE 16 – IMU ORIENTATION OUTPUT AGAINST MEASURED ORIENTATION....86
FIGURE 17 – MAGNITUDE OF ERROR FUNCTION WITH RESPECT TO ROTATION....91
FIGURE 18 – EVALUATION OF CONVERGENCE OF DIFFERENT GDAS....93
FIGURE 19 – EVALUATION OF CONVERGENCE OF DIFFERENT GDAS (NO INCLINATION)....93
FIGURE 20 – THE EFFECT OF MAGNETIC INCLINATION ON TIME TO CONVERGE....94
FIGURE 21 - RESULT OF ROTATING MEASURED MAGNETOMETER NINETY DEGREES AROUND Z-AXIS....95
FIGURE 22 - EFFECT OF GYROSCOPIC BIAS ON CONVERGENCE....96
FIGURE 23 - THREE 'OPEN' GESTURES IDENTIFIED AND SEGMENTED FROM NON-AMPUTEE SUBJECT....109
FIGURE 24 - ONE HUNDRED INSTANCES OF THE 'OPEN' GESTURE FROM ONE NON-AMPUTEE SUBJECT....110
FIGURE 25 - HOW NUMBER OF TRAINING SAMPLES AFFECTS ACCURACY....121
FIGURE 26 – FLOWCHART SHOWING STEPS FROM DATA ACQUISITION TO GESTURE IDENTIFICATION....124
FIGURE 27 - EFFECT OF NUMBER OF MMGS ON CLASSIFICATION ACCURACY....124
FIGURE 28 - EXPERIMENTAL HARDWARE....130
FIGURE 29 - EXPERIMENTAL SETUP....130
FIGURE 30 -USING RIFT CONTROLLER TO MONITOR HAND POSITION....136
FIGURE 31 - USING IMUS TO DERIVE HAND POSITION....136
FIGURE 32 - AVERAGE DEVIATION FROM A DIRECT PATH....137
FIGURE 33 – POSITIONS OF GESTURES CREATED IN TWO ARM CONFIGURATIONS....144
FIGURE 34 – POSITION OF ARMBAND FOR GESTURE-BASED EXPERIMENTS....146
FIGURE 35 - THE SIX TASKS FOR VR ROBOT CONTROL....147
FIGURE 36 – AXIS OF ROTATIONS OF VIRTUAL KUKA ROBOT....187
FIGURE 37 – VIRTUAL KINEMATIC MODEL FOR SOLVING ELBOW UP INVERSE KINEMATICS....187

Nomenclature

Abbreviations

ACC	Average Amplitude Change
ADC	Analog-to-Digital Converter
ADL	Activities of Daily Living
ANN	Artificial Neural Networks
AR	Autoregressive models
BC	Bayesian Classifier
BCI	Brain Computer Interface
BLE	Bluetooth Low Energy
BMI	Brain Machine Interface
CAVE	Cave Automatic Virtual Environment
CNN	Convolutional Neural Networks
CSVM	Cubic Support Vector Machines
DAMV	Difference Absolute Mean Value
DASDV	Difference Absolute Standard Deviation Value
DoF	Degree of Freedom
DT	Decision Trees
ECoG	Electrocorticography
EEG	Electroencephalography
EKF	Extended Kalman Filter
EMG	Electromyography
emgHIST	Electromyographic Histogram
EPP	Extended Physiological Proprioception
FL	Fuzzy Logic
FSM	Finite State Machine
GUI	Graphic User Interface
HCI	Human-Computer Interaction
HMI	Human-Machine Interface
HMM	Hidden Markov Models
HPE	Human Pose Estimation
I/O	Input/Output
I2C	Inter-Integrated Circuit
IAV	Integral of Absolute Value
IC	Integrated Circuit
IMU	Inertial Measurement Unit
IR	Infrared
KNN	k-Nearest Neighbour
LDA	Linear Discriminant Analysis
LED	Light Emitting Diode
LOG	Log-detector
MARG	Magnetic, Angular Rate and Gravity

MAV	Mean Absolute Value
MEMS	Micro-electro-mechanical systems
MIPS	Million Instructions per Second
MK	Myokinemetric
MLP	Multilayer Perceptron
MMG	Mechanomyography
MMI	Man-Machine Interfaces
NUI	Natural User Interface
PCB	Printed Circuit Board
PR	Pattern Recognition
PTSD	Post-Traumatic Stress Disorder
PWM	Pulse Width Modulation
QDA	Quadratic Discriminant Analysis
RFS	Random Forests
sEMG	Surface Electromyography
SMG	Sonomyography
SPI	Serial Peripheral Interface
SSI	Simple Squared Integral
SSP	Serial Port Profile
SVM	Support Vector Machine
TCP/IP	Transmission Control Protocol Internet Protocol
TMEP	Tongue Movement Ear Pressure
TSS	Toxic Shock Syndrome
UART	Universal Asynchronous Receiver-Transmitter
V	v-Order detector
VAR	Variance
VR	Virtual Reality
VUI	Voice User Interface
WAMP	Willison Amplitude
WIMP	Windows, Icons, Menus and Pointers
ZC	Zero-Crossing

Symbols

$\boldsymbol{q}$	Quaternion
$\boldsymbol{q_t}$	Quaternion at time $\boldsymbol{t}$
$\boldsymbol{v}$	Quaternion which represents a vector
$\boldsymbol{u}$	Vector
R	Rotation Matrix
$\boldsymbol{v}_{r(a)}$	Reference acceleration vector
$\boldsymbol{v}_{r(m)}$	Reference magnetometer vector
$\boldsymbol{v}_{m(a)}$	Measured acceleration vector
$\boldsymbol{v}_{m(m)}$	Measured magnetometer vector

SF	Scaling factor
err	Error
e_{sig}	Energy of MMG signal
T_{MMG}	Threshold of MMG signal
$T_{g_{xyz}}$	Threshold of gyroscopic signal
$\|\|G\|\|$	Energy of gyroscopic signal
N	Number of MMGs
m	Continuous vector containing signal from MMG
m^*	Filtered continuous MMG data
s_N	Vector containing segmented signal from MMG
ρ	Pearson Product-Moment Correlation Coefficient
σ	Standard deviation
p	Pointer
ω	Output of gyroscope

Chapter 1

Introduction to the problem

1.1 Project Background

This thesis details an investigation into the use of biomechanical information to derive user intent for Human-Machine Interfaces (HMIs). Almost all HMIs currently available require the user to perform dexterous manipulations on an application specific interface. This requires fine motor control that makes such interfaces unusable for some. Simultaneously, we are surrounded by an ever-increasing quantity of smart devices, all presenting obtrusive interfaces as they compete for our attention. As these devices become more pervasive, studies on the field of Human Machine Interfaces (HMIs) have identified a push towards more generalized controllers that make use of natural forms of interaction such as gestures. It has been suggested that this will make users more comfortable with these devices. This thesis describes the design and implementation of a sensor suite to enable pervasive monitoring of body kinematics and muscular activity, and the development of algorithms to allow for the volitional control of smart/robotic devices, providing such an interface.

The sensor suite tracks activity through the use of an Inertial Measurement Unit (IMU), and Passive Sensors for monitoring the mechanical properties of muscle during the contraction cycle. The passive muscle sensors monitor the mechanical waves that are produced during contraction, a technique known as Mechanomyography (MMG). All of the sensors used in this investigation are non-invasive and designed to be unobtrusive during prolonged use.

1.2 Motivation

Human-Machine Interfaces are intended to bridge the gap between a person and their technology. The intent is to make the technology as easy to use and accessible as possible, but in some cases, this is achieved by selecting an interaction method that is not accessible to everybody, thereby excluding sections of the population from the intended user base. Unfortunately, these excluded sections tend to be those who could benefit the most from technological assistance. An example of an excluded population section is upper limb amputees; for this group losing a hand often means losing the primary method of interacting with the environment, and therefore the primary method of interacting with technology. A range of assistive technologies have been created, however, in many cases, these are designed to provide an amputee with a method of controlling a second interface, further isolating them from the device they are trying to use. Examples of this include advanced prosthetics which are designed to make it possible to use a computer mouse. In this application, when an actuation in desired, the wearer must focus on controlling the prosthetic hand rather than on the task they are trying to complete with the computer mouse, leading to an increased level of cognitive switching. This can lead to faster fatigue and dissatisfaction with the device. These prosthetic interfaces are often difficult to use and have high rates of dissatisfaction and rejection. Long-term use is also associated with repetitive-strain injuries, soft tissue damage and several other complications including bone fractures, and bone bridging.

The solution is to make one or more of these interfaces easy to use, to the point where it appears transparent to the user. This means they must be so intuitive that the user is not focused on generating the control signals, but instead on the task they are trying to complete. In the case of prosthetic control,

using pervasive monitoring to determine the intent of the prosthetic user will allow them to use the hand as if there is a neurological connection. While truly dexterous control may be beyond our current capabilities, pre-programmed motions, initiated by the corresponding gestures, would provide significantly improved functionality over current prosthetic devices, and may improve the satisfaction with the prosthetic.

Controlling other interfaces could still be challenging, however a generalizable interface that provides gesture control over smart devices may also provide a solution. While it is unlikely the gesture control will provide a complete solution for natural human-machine interaction, when paired with voice control it would provide the most natural form of interface available. Voice controlled smart devices are increasing in popularity, however gesture controllers have been hampered by technological constraints which are inherent in sensing modalities such as electromyography (EMG). These constraints include precise calibration, which leads to reduced functionality over time, a factor that is far less relevant with MMG-based systems.

MMG is an underexplored sensing modality that has the potential to provide robust and highly function method of gesture recognition. Whilst other more mature sensing modalities have been unable to provide this robust control, MMG can provide this type of improved functionality to those who are excluded from traditional human machine interfaces.

1.3 Aim

The primary objective of this study is to assess the utility of fusing muscle activity and inertial measurement for HMI applications. This will be accomplished in two ways: firstly, the precision and accuracy of each of the sensors will be established, to ascertain whether it is possible to accurately determine intent from the muscle information; and secondly, the suitability of the sensors will be examined by assessing the performance of the system whilst performing specific tasks. There are many forms of interaction being researched for HMI, and a complete system will need to be multi-modal. This study therefore identifies tasks best suited to interfaces of this nature.

To demonstrate the diversity in the application of the system described here, three forms of output were chosen that best demonstrate the strengths of sensor suite. It should act as a User Interface for smart devices, facilitate the control of Upper-Limb Prosthetic Devices and allow Robot Teleoperation. Each of these system outputs has different motivations and introduces application specific system requirements.

1.3.1 Smart Systems

In a modern smart home, a user may interact with numerous interconnected application specific computing devices. Reducing the intended functionality of each device enables the user interaction to be simplified, ideally to the point where the user is no longer overtly aware that it is a computer that they are interacting with.

A system of this nature must have three features:

- Pervasive Monitoring - The system should be able to track the user in some way and record possible interactions.
- Context-Aware Computing - The system should be aware of the environment in which the user is performing these interactions.
- Artificial Intelligence - The system should be able to use environmental factors and user interactions to derive user intent.

The sensor suite is utilized in this thesis as a Pervasive Monitoring system and is demonstrated by applying constraints in a known context as a facsimile to a complete system.

The context specific objective is as follows:

- Develop a method of interacting with smart devices using natural gesture commands.

Other systems have been produced with similar objectives to this, however they rely on alternative sensors that have both a greater monetary expense and greater power consumption.

1.3.2 Controlling Upper-Limb Prosthetic Devices

The control of a prosthetic hand is a useful demonstration of the utility of the MMG sensor. For amputees who have undergone a lower limb amputation, active prosthetics are most commonly controlled using inertial measurement units, however this is not practical for upper limb amputees since it relies on cyclical movement patterns to predict future motions. Dexterous hand movement consists of an intended gesture in a specific location, and since this constrains the motion of the arm, monitoring muscle activity is one of the common alternatives for prosthetic hand control.

Active prosthetic hands are most commonly controlled using electromyography (EMG). EMG is the name given to the process of monitoring the electrical impulses that can be recorded propagating through muscles as they contract. Surface EMG (sEMG) involves sensors placed on the surface of the skin and can be used within the socket to detect impulses in the residual limb. This provides the wearer with a binary form of control over their prosthetic. Despite research into methods of increasing the bandwidth of the EMG signal to allow for a more effective control, translation to market has not yet occurred. One reason for this is that calibrated EMG is reliant on skin conditions that are not consistent, particularly inside the airtight socket. While experiments in a controlled environment can generate good results, they do not solve the problems of practical application outside that environment.

MMG has the potential to provide a far more consistent method of control by monitoring the physical movements of the muscles in the residual limb. By monitoring the way that the signals from different muscles interact with one another, it is shown in this thesis that complex gestures involving multiple muscle contractions can be distinguished. The base amplitude of the MMG signal is also much higher, and as a result, the movement required is much smaller, delaying the onset of muscle fatigue so that MMG signals remain a useful method of interaction for longer.

The objectives in this section are as follows:

- Generate a more manageable method of prosthetic hand control than is currently available.
- Design a method of interaction by which prosthetic users can use the control system to interact directly with smart devices without using their prosthetic.

These objectives describe a new method of interaction that may lay the groundwork for new research which can be translated to market, benefiting those currently poorly served by the existing commercial solutions.

1.3.3 Robot Teleoperation

Robot teleoperation demonstrates the full utility of this system for capturing dexterous movement. Data taken from inertial sensors can be used to calculate the movement of the user's limb within a real-world reference frame, and the muscular information is used to determine the intended action of the end effector. In the case of the experiment performed in this thesis, the actuator is a 14DoF robotic platform produced by Rethink Robotics. In addition to the robot teleoperation, the arm has numerous autonomous functions that can be initiated by the user therefore the system can provide the user with a number of seemingly natural outcomes through an intuitive method of directing the semi-autonomous controller.

The application specific objectives of the robot teleoperation are:

- Develop the real-time semi-autonomous controller for the robot, comparing both the system described in this thesis and a solution with is more aligned with industry standard methods.
- Perform this task in simulated real-world conditions.

1.4 Contributions

There are several contributions that have been made to the field of Mechanomyography for gesture recognition, as well as a number of application and algorithm specific contributions. These are summarized in the following points.

- The creation of a new form of multimodal interface that makes use of both the mechanical signals generated through muscle contraction, and inertial information to provide a method of gesture recognition.
- Improvements to the robustness of a pre-existing, commonly used, gradient descent inertial measurement orientation estimation algorithm.
- Introduction of a new IMU orientation estimation algorithm with predictable convergence.
- MMG classification through basic and machine learning classifiers, both offline and in real time.
- Introduction of a new method of fusing MMG signals with inertial data to provide classification accuracy which is not dependent on consistent arm pose.
- Application of this system for prosthetic control and for robot teleoperation, both in virtual environments and in real world applications.

1.5 Publications

1.5.1 Journal Papers

- **S Wilson**, H Eberle, Y Hayashi, S O.H. Madgwick, A McGregor, X Jing, R Vaidyanathan, Formulation of a new gradient descent MARG orientation algorithm: Case study on robot teleoperation, Mechanical Systems and Signal Processing, Volume 130, 2019, Pages 183-200, ISSN 0888-3270. (Impact factor: 5.0)
- W Huo, P Angeles, Y F Tai, N Pavese, **S Wilson**, R Vaidyanathan, "A Sensor System Framework to Quantify Parkinson's Disease Symptoms", IEEE Transactions on Neural and Rehabilitation Engineering. (Impact factor: 3.4)

1.5.2 Conference Papers

- **S Wilson**, R Vaidyanathan (2017) Gesture Recognition Through Classification of Acoustic Muscle Sensing for Prosthetic Control. In: Mangan M., Cutkosky M., Mura A., Verschure P., Prescott T., Lepora N. (eds) Biomimetic and Biohybrid Systems. Living Machines 2017. Lecture Notes in Computer Science, vol 10384. Springer, Cham
- **S. Wilson** and R. Vaidyanathan, "Upper-limb prosthetic control using wearable multichannel mechanomyography," 2017 International Conference on Rehabilitation Robotics (ICORR), London, 2017, pp. 1293-1298.
- P. Angeles, Y. Tai, N. Pavese, **S. Wilson** and R. Vaidyanathan, "Automated assessment of symptom severity changes during deep brain stimulation (DBS) therapy for Parkinson's disease," 2017 International Conference on Rehabilitation Robotics (ICORR), London, 2017, pp. 1512-1517.
- Y. Ma, Y. Liu, R. Jin, X. Yuan, R. Sekha, **S. Wilson** and R. Vaidyanathan, "Hand gesture recognition with convolutional neural networks for the multimodal UAV control," 2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS), Linköping, Sweden, 2017, pp. 198-203.
- M. Admiraal, **S. Wilson** and R. Vaidyanathan, "Improved formulation of the IMU and MARG orientation gradient descent algorithm for motion tracking in human-machine interfaces," 2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), Daegu, Korea (South), 2017, pp. 403-410.
- A. P. H. Needham, F. P. Paszkiewicz, M. F. Md Alias, **S. Wilson**, A. A. Dehghani-Sanij, B. C. Khoo, R. Vaidyanathan, "Subject-Independent Data Pooling in Classification of Gait Intent Using Mechanomyography on a Transtibial Amputee," 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 2018, pp. 1806-1811.
- C. Caulcrick, F. Russell, **S. Wilson**, C. Sawade and R. Vaidyanathan, "Unilateral Inertial and Muscle Activity Sensor Fusion for Gait Cycle Progress Estimation*," 2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob), Enschede, 2018, pp. 1151-1156.
- L. Formstone, M. Pucek, **S. Wilson**, P. Bentley, A. McGregor and R. Vaidyanathan, "Myographic Information Enables Hand Function Classification in Automated Fugl-Meyer Assessment," 2019

9th International IEEE/EMBS Conference on Neural Engineering (NER), San Francisco, CA, USA, 2019, pp. 239-242.

1.6 Layout of Report

The layout of the remaining six chapters of this thesis is described below.

Chapter 2 - Literature Survey. This chapter explores the work that has been done prior to this study. It includes work in the fields of HMIs, pervasive monitoring, prosthetic control, mechanomyography, sensor fusion and signal segmentation, and processing. It examines both the requirements for improved interfaces and the sensing modalities that new interfaces could exploit. It also evaluates each of these modalities against criteria determined through discussion with end users.

Chapter 3 - Hardware Development. This chapter describes the process of creating the new hardware required to perform this study. The design process is documented, including the production of both the hardware and the complementary software.

Chapter 4 - Sensor Validation. This chapter describes the process of evaluating the accuracy of the orientation estimation algorithm used. It details factors that are known to affect this accuracy and then provides several modifications to the original algorithm to improve the accuracy and robustness. Finally, a new algorithm is outlined.

Chapter 5 - Gesture Recognition. This chapter focuses on intent prediction using myographic information. Segmentation and extraction are performed to detect when a gesture may have occurred and then the classification process ascertains which gesture was made. This chapter details both custom template-based methods and more advanced machine learning classification methods. Several algorithms have been tested and evaluated both in offline and online applications.

Chapter 6 - Application: Prosthetic Control and Teleoperation. This chapter outlines the application of this control system to the field of robotic control, both for assistive technology and for advanced robotic control. It evaluates the system using virtual environments that allowed the interface to be evaluated against known positions. It includes modifications required for practical use and then evaluates the usability of the system through several real-world tasks.

Chapter 7 – Conclusion. This chapter provides an overview of this document and describes how the objectives and contributions outlined in this chapter were met.

Chapter 2

A Review of the Existing Literature

2.1 Structure of the Literature Review

This chapter is broken down into two main sections. The first section discusses Human Machine Interface technologies, prosthetics, assistive devices and sensing technologies. This section will contain the literature relevant to the decisions made in Chapter 3. In the second section, orientation monitoring algorithms are discussed, and a brief overview of quaternion operations is given. This section of the literature review leads into the algorithm descriptions in Chapter 4.2. The layout of this chapter is described in the Figure 1.

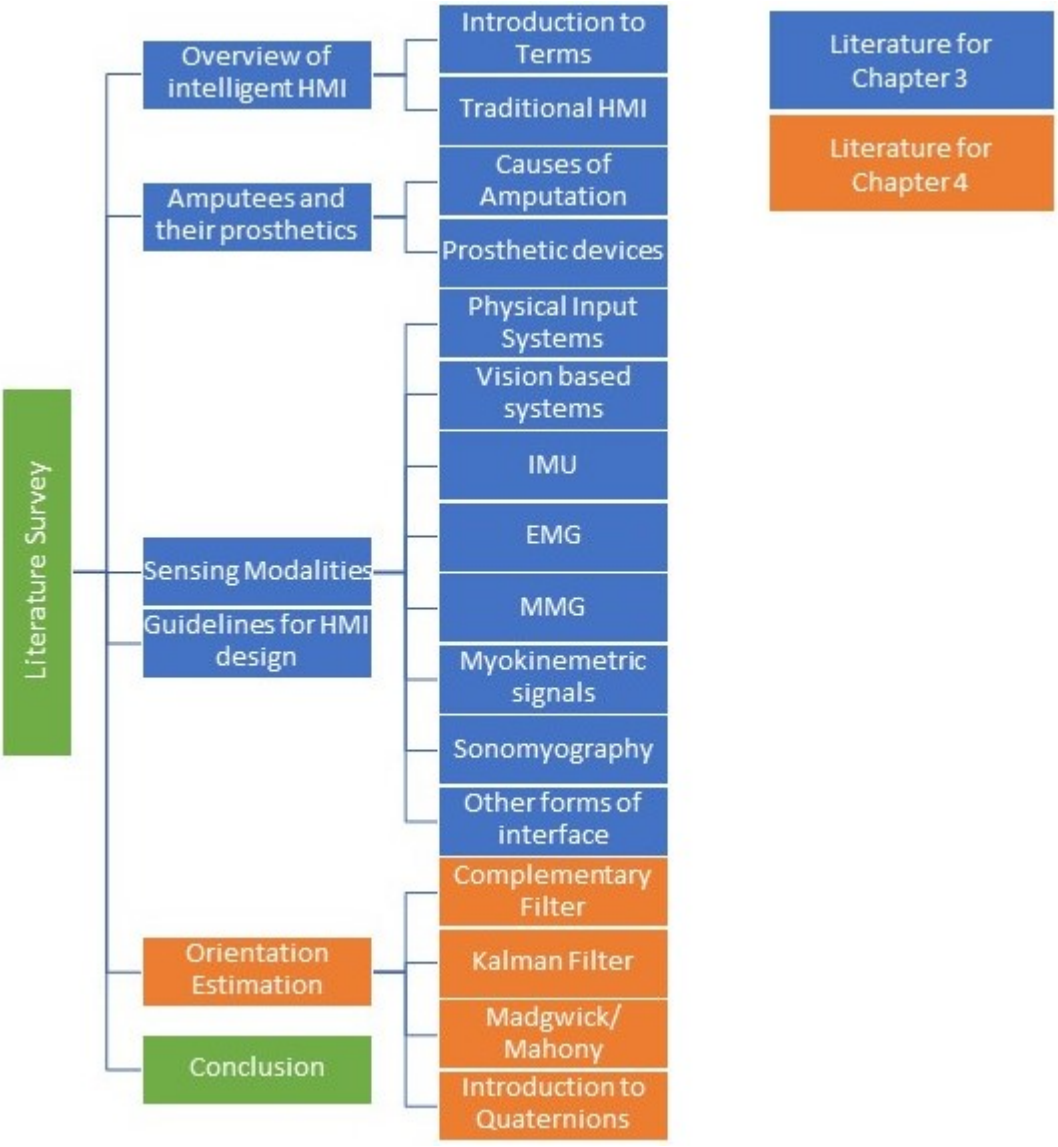

Figure 1 - Layout of literature review

2.2 Overview of Intelligent Human-Machine Interfaces and Sensing Technology

2.2.1 Introduction to Terms

Human-Machine Interfaces (HMIs, also referred to as Man-Machine Interfaces or MMIs) are a technology designed to pass information between humans and machines. HMIs range from those that require volitional control and active engagement to achieve an objective (referred to as active HMIs), to those that function as pervasive monitors where the command instruction may be achieved sub-consciously by those using the interface (referred to as passive HMIs). In the most extreme case of an active HMI, the user is focused on manipulating the interface, whereas at the other extreme the user is functionally unaware of the interface. Active and passive HMIs have very different use cases, and both have been explored extensively

Intelligent HMIs are those that attempt to reduce the cognitive load for the user by presenting them with a representation that matches the model of the task that they have in their head [11]. Nearly all modern HMIs can be thought of as intelligent HMIs, and this can be observed by the language that is used when referencing functions: data is stored in ‘files’, which are organized into ‘folders’; we 'empty the recycle bin' instead of removing headers to delete these files. As these representations become more closely aligned with the users’ mental model, they move closer to becoming a Natural User Interface (NUI). An NUI is described as an interface where the method of interaction is natural to the user and is made evident by the design of that interaction; meaning explicit instruction or training is not required. Whether there is currently such a device can be debated, but the technology that has been developed to behave like an NUI does so by attempting to remove the cognitive barrier between the user and the data they are manipulating. As with an active interface, the user is still actively engaging with it and providing volitional commands, but it is functionally invisible to them.

This chapter draws together the existing literature on HMIs across two developmental work strands, that which focuses specifically on prosthetic control and that which examines more general HMIs. It attempts to compile the existing guidelines on creating a successful interface so that an appropriate list of specifications can created in future chapters.

2.2.2 Traditional HMI

There are a wide variety of conventional technologies that are currently used to provide HMI, such as the monitor, the computer mouse and the keyboard. The keyboard was designed to allow people who were unfamiliar with computers to use prior knowledge from the use of typewriters, thereby reducing the training time and cognitive load. Early keyboards were used to punch holes in paper tapes that carried the code to be loaded to the system. Output from the system was via similar tapes that were replayed through teletype machines to translate the punched holes back into a readable format. In an attempt to make computer manipulation more intuitive and accessible for non-programmers, the computer mouse were developed along with the ability to manipulate graphical objects on the screen in the late 1960s, leading to the Graphical User Interface (GUI) [12]. GUIs most often use a combination of selectable workspaces on the screen, known as Windows, along with Icons, Menus and Pointers (known WIMP-based GUIs) to

simplify the HMI for the user. Other HMI devices, such as joysticks, dials and buttons all predate digital computing, yet are still commonly found as HMIs today.

These traditional HMIs have two distinct disadvantages. The first is that the desire to build on prior knowledge and the technical limitations at the time of development have led to technically inefficient interfaces. The clearest example of this is in the layout of the computer keyboard in the QWERTY format. It has been suggested that this keyboard layout was originally intended to slow down typists on mechanical typewriters to ensure the keys and type hammers had sufficient time to respond and return to rest without getting mechanically jammed together. While more efficient layouts have been proposed, they have not been universally adopted, partly due to extremely widespread use of QWERTY format keyboards leading to almost universal familiarity with the layout among English speaking keyboard users, and party due to the time taken to learn a new key placement [13]. The other disadvantage of conventional interfaces is that they require dexterous manipulation of physical objects to achieve a task, which render the interface poorly suited for some, and impossible to use for others.

These two points form the basis of the motivation for the research that underpins this document. Increasing the efficiency of existing interfaces will not lead to the widespread adoption of a novel interface, however building an interface for those who cannot use the traditional technology will provide a method of validating it. If the interface can be used by the wider population then this validation may lead to a higher chance of adoption of the technology.

Despite the pervasive nature of WIMP HMI there remain many groups for whom the required dexterous manipulation is challenging. The exact nature of the difficulty an individual might experience can vary, but there are many diseases for which a loss of dexterous control may be a symptom. An example of this is Parkinson's disease, a degenerative brain disorder where loss of nerve cells in the brain leads to tremors and stiffness of the muscles. Parkinson's is said to affect 1 in 500 people to some degree. Another cause of loss of fine motor control is Stroke, with up to 90% of Stroke survivors suffering from some degree of paralysis. There are many other diseases that may cause a reduction in dexterity; there are also many reasons why an individual may lose one or both hands completely.

2.3 Amputees and their prosthetics

2.3.1 Causes of Amputation

It is estimated that there are over 2,000,000 people who are living in the US alone with a limb loss, normally as a result of either trauma or disease [14]. Approximately 1 in 4 of these involve the amputation of some part of one or both upper limbs. The leading cause of upper limb amputation is trauma, which leads to 77% of these amputations. This includes both injuries where the limb is determined to be unsalvageable and traumatic amputation, where the limb is severed in the incident. In many cases, individuals who have survived traumatic amputation require a further amputation proximally to create a residual limb that can be used to control a prosthetic in future. The leading cause of traumatic amputation is vehicular accidents, however, they are also a common injury sustained during conflicts.

Congenital amputation/Congenital upper limb deficiency makes up approximately 9% of upper limb amputees. Congenital amputation is caused when the tissue in the limb of the foetus dies, resulting in the loss of part or the whole of the limb. There are various causes of this tissue death, with a common one being amniotic band constriction.

A common form of cancer that can lead to amputation is bone cancer, an example of which is Osteosarcoma. Often, a surgeon will attempt to remove only the tumour, however, in many cases this is not possible. The second option for the surgeon is to perform limb-sparing surgery, where the portion of bone in which the cancer is growing is removed and replaced with a metal prosthesis. If the portion of bone is near the joint, then this may necessitate the removal and replacement of the joint as well. If these options are impossible or fail, or if the body rejects the implants, then the limb must be removed. Amputations due to cancer make up approximately 8% of upper limb amputations.

Another common cause of amputation is disease. Physiological diseases such as diabetes are the leading cause of lower limb loss and can also lead to upper limb amputation. Infectious diseases can also require amputation, either due to the damage done to the limb, or to prevent the disease spreading. Amputation in this case is normally only performed when the disease is progressing too fast to be controlled and where less aggressive solutions have not worked or are more likely to lead to the death of the patient. An example of this is sepsis, where tissue death in a limb can require amputation to stop the spread of infection. Diseases are the cause of approximately 6% of upper limb amputations.

Whatever the cause, it is difficult for amputees to adapt to the loss of one or both hands. Often amputees suffer from depression in the time after their amputation and for those who have lost limbs in a traumatic or distressing event, Post-Traumatic Stress Disorder (PTSD) is also common. Activities of Daily Living (ADLs) become much harder, indeed some may become impossible because so many of the interactions in our lives are designed with the dexterity of the hand in mind. One way to try and restore some of this lost functionality is with prosthetic devices. A prosthetic device that works well is incredibly important for amputees, both in terms of their quality of life, independence and in terms of their mental health.

2.3.2 Prosthetic Devices

A prosthesis is a device that is designed to replace a body part which has been lost. Current prosthetic devices can restore some of the lost functionality, however, they are unable to mimic the high number of degrees of freedom of the hand and wrist (21 Degrees of Freedom in the hand and 6 in the wrist). There are many prosthetics available for those who have lost a hand and the range can be separated into two categories: passive and active devices. There are two types of prosthetic that fit into the passive category, cosmetic and functional. Active prosthetic devices are those that have driven moving parts and can be either body-powered or externally powered.

Traditionally, cosmetic prosthetics are designed to disguise the wearer's amputation. They are often tailored specifically to their wearer's shape and skin colour and can even be customised to include tattoos that may have been on the original limb. More recently, a different kind of cosmetic prosthetic has gained more popularity, in which the device is designed as an art piece to be worn. These are crafted not to hide

the amputation, but to emphasis it with the intent to promote discussions about disability and body diversity, while at the same time giving the amputee something unique that they can be proud of, instead of a less-functional replacement.

Functional passive prosthetics are simply tools with which the wearer can interact with their environment to achieve specific tasks. These have traditionally been shaped metal and more recently tools that allow you to use touch screen devices have also become popular.

Active prosthetics are designed to allow the user to perform a greater number of ADLs by giving them control over a one-or-more degree of freedom manipulator. As previously stated, active prosthetics can be categorized either as body-powered, where the energy required to perform the movement is generated by the user, or externally powered, where the energy is provided by some other source. Body-powered prosthetics are usually restricted to either one or two degrees of freedom. A typical example of a body-powered prosthetic is a split hook prosthetic, where a harness worn around the shoulders can be used to open the gripper, and elastic or springs are used to close it again. These prosthetics can often also be rotated at the wrist to allow for extra functionality. Externally powered prosthetics normally require a power source mounted on the device. They also usually attempt to make a prediction of intent by observing muscle activity and using that to trigger movement in the end effector. Prosthetics that do this are known as myoelectric prosthetics.

The movements that can be triggered in the end effector of a myoelectric prosthetic depend on the physical nature of the device: a split hook will normally open/close based on one control signal and may rotate based on another. Prosthetics that have been designed to mimic biological hands offer a greater range of motion, with the most advanced offering around 7 Degrees of Freedom. Due to the difficulty inherent in controlling all of these individually, the prosthetic will often work as a semi-autonomous system, where the user can select the action they would like the hand to do, and then trigger the hand to perform that action. That hand will then control each of the motors to perform the desired gesture, factoring in each digits resistance to movement. This is known as grip selection and provides the wearer with the tools to complete a wide range of ADLs.

Of all the available options, externally-powered grip-switching prosthetic devices are the most functional option which still maintain the appearance of a hand. Despite this, only 34% of frequent prosthetic users who could use a myoelectric prosthesis regularly wear one [15]. In addition, the rejection rate of upper limb prosthetics is between 20-50% [16, 17]. The studies relating to rejection often use relatively small sample groups, and the participants are often recruited through prosthetic clinics, which mean that the results cannot be reliably scaled to the overall amputee population. This is because amputees who have elected not to use a prosthetic, or who cannot afford one, will not attend a clinic and therefore will not be represented in the studies, leading to a positive skew in the reported results. Internet surveys also may receive an increased return rate from those who have found a prosthetic solution that works for them, as those who haven't may be less likely or unable to complete the survey.

While the use and satisfaction rate should not be extrapolated to the population, the individual responses can be examined to determine some of the factors that may lead to rejection, as well as the prevalence of those factors within the surveys participants. Some of the most common factors with prosthetic rejecters were either physical, such as it being too hot, heavy or uncomfortable, or that it was too difficult to use [18]. A common factor with those who have suffered an amputation distal to the shoulder was that it was easier to do the task without the prosthetic.

Prosthetics can not only be used to assist with ADLs, they can also help prevent the onset of further medical problems such as phantom pain, or depression and other psychological conditions [19]. Whether the amputee has chosen to use a prosthetic or not, they will need to develop methods of compensate for the limb-loss. The amputee's compensation strategies can lead to further problems, such as back pain or overuse injuries in the intact limb [20, 21]. The resistance in body-powered prosthetics is often identified as a causal or contributory factor to these injuries. Provided user intent can be derived, high DoF Myoelectric prosthetics have the potential to offer the most naturalistic movement, and as a result, the user would be able to operate it to the same extent as an intact hand.

The physical factors that may lead to the rejection of a myoelectric hand are limited by the design and material science of the device itself, however the ease of use is dependent on the HMI. With increased functionality and a reduction in cost, the majority of prosthetic rejecters would be willing to consider using a new device, and therefore improving the HMI will be key [22].

2.3.3 Assistive HMIs

Assistive HMIs are a communication technology designed to bypass a disabled individuals impairment and allow them to perform tasks that would otherwise be difficult [23]. Prosthetic controllers can fall into this category, and most studies that aim to develop assistive technology for amputees focus on them, however there are other avenues that have the potential to be equally useful. In this document, 'assistive technologies' will be defined as distinct from prosthetic devices for simplicity. Amputees wear their prosthetics for between 4-8 hours a day on average, mainly due to the weight and discomfort associated with wearing the socket. It would also be highly desirable to identify a technology that also allows an amputee to interact with their environment without having to use a prosthetic, in part due to the amount of time that the prosthetic is not being worn, but also because some tasks are fundamentally just difficult to perform with a prosthetic. A good example activity is changing the channels on a television using a remote control, which requires fine motor control of individual fingers not currently available with modern prosthetics.

Assistive technologies have been extensively examined for individuals with other conditions but are seldom designed with amputees in mind. This may well be due to a perception that a prosthetic device presents a 'solution' to limb loss, as opposed to a solution for specific problems that arise as a result of losing a limb. The 'solution' mind-set does not leave room for development outside of the prosthetic work stream. Despite this, the control signals that could be generated by an amputee are as applicable to a robot, computer or TV as they are to a prosthetic device, and in recent years, some work has begun to emerge that removes the prosthetic device for the HMI, and allows the amputee to interact directly with

the computer [24]. This work is still significantly less common than that which includes prosthetics as the medium for interaction.

Assistive technologies that allow for robot teleoperation are also an under examined research area. While physically controlling a multi-DoF robot is not practically useful at this time for amputees, it is a complex task requiring accurate control in 3D space. As such, robot teleoperation is analogous to both complex direct computer control, and smart home control. A sensor suite that demonstrates this high bandwidth interaction could allow more complex HMIs to be developed to assist amputees with their ADLs, as well as other more adventurous activities they may wish to participate in.

Finding a sensor suite capable of both prosthetic control and control of other assistive technologies is important for giving amputees the ability to interact with their environment, whether this interaction is physical or through a computer. Various sensors have been tested for prosthetic control, and there have been several studies that describe the development of HMIs for healthy individuals, and this will be explored in the following section.

2.4 Sensing Modalities

This section will discuss the existing literature in the fields of prosthetic control and HMI, grouping research by sensing modality and then by application. This will allow us to examine the solution space and create a system capable of fulfilling the design criteria. Since this study focuses on pervasive gesture recognition devices, the section will focus principally on interfaces that do not require dexterous manipulation as their primary input modality. Each sensing modality will be described, and work performed in the field of activity/gesture recognition/monitoring in prosthetics and Human Machine Interface will be outlined. Where appropriate, results of gesture recognition studies will also be included.

2.4.1 Physical Input Systems

Physical input systems are those that require some physical manipulation as a method of interaction. They tend to be the most common form of interface, since they can be unobtrusive enough for a ubiquitous system, and yet still give both an input and an effect that the user can observe, making it relatively easy to learn how to use them.

In the field of upper limb prosthetics, systems that require some form of physical input as their primary control signal are referred to as body-powered prosthetics. As previously indicated, body-powered prosthetics are the most common method for control and have several advantages that are unique to this form of control.

The most common examples of body-powered upper limb prosthetics are cable driven split hooks. Split hooks are normally a single-DoF device that the wearer can open by pulling a cable attached to a harness worn around the shoulders. This is normally achieved by a combination of pushing the shoulders forward and straightening the arm. In addition, some split hooks give the wearer the ability to change the force applied by the elastic, allowing the wearer to select different grip strengths, usually through a linear ratchet moved with a toggle. Finally, the wrist assembly also can be rotated. While strength and rotation

are both physical inputs, they are more akin to settings of the split hook, rather than the physical control signals generated through the cable

The combination of cable and elastic facilitates proportional control of the DoF, which is lacking in many other forms of interface. The amount that the hook opens is directly controlled by the amount of force that the wearer exerts through the cable, and they can learn what is required to open the grippers by any fraction very quickly because it is consistent. The physical system does not introduce any delays, and the user's perception of the force they are applying allows them to feel feedback. Due to this, a user who has been using one of these prosthetics for an extended period of time is able to accurately use the gripper without visual feedback. This ability for a healthy individual to perceive their joint angles and therefore position is called proprioception. When a user becomes so capable with a tool that they understand where the end of it is in relation to them, this is known as Extended Physiological Proprioception (EPP). An example of this principle that will be familiar to most is a white cane, used by people who are blind or have low vision to feel objects and obstacles in their environment.

In the field of prosthetic control, the term Extended Physiological Proprioception has garnered a more specific meaning in recent years. Employing the principles of EPP into prosthetic devices is desirable, since the ability to sense the position and state of a gripper allows the prosthetic to be considered to be a closed loop system [25]. This allows the wearer to develop control strategies that are both intuitive and feel natural. More recently, papers referring to the development of EPP controllers have become more prevalent. These papers use the term to refer to controller that attempt to close the feedback loop without the gripper being a part of that loop. This proprioception lets the user know that they have made the correct control signal, regardless of whether the prosthetic has moved or not. A common example of this is to use shoulder position relative to a relaxed zero point to control an upper limb prosthetic [26]. Incorporating the concept of EPP into a prosthetic controller is an important desired requirement for upper limb prosthetics, as the intuitive control strategies are vital for reducing rejection, and by their nature, body-powered prosthetics achieve this.

Body-powered prosthetics are not a complete solution to the problem of controlling upper limb prosthetics for two reasons. First, wearing a body-powered prosthetic means repurposing functions of the body. This may be as simple as remapping one-DoF in the shoulder to control a hand or elbow, or as in the case of commercial harness driven systems, it may require a more complex movement to actuate a single-DoF gripper. In either case, the number of joints that can be controlled is limited to 1-2. The second and potentially more serious problem with body-powered prosthetics is the range of long-term health problems associated with them. Many users report persistent back and shoulder pain, and this is a greater cause of rejection than the lack of functionality. This is due to the stresses associated with performing the required movements, which are not necessarily natural to the body. As mentioned previously, RSIs are very common with users of body-powered prosthetics, and this can be attributed to both the use of the device, and the user's compensation strategies. Prosthetics with an elastic element require repeated resisted movements and are often heavy. The weight of the device is considered a problem for two reasons: the effort required to move it, and the poor weight distribution, with the majority of mass often

being the gripper/hand at the end of the arm. The nature of the prosthetic socket mean that the force applied to the residual limb by the weight of the prosthetic is applied mostly on a small area at the end of the residual limb and the force that the wearer is applying to actuate the arm is also transmitted primarily by the end of the arm. This can cause damage to the soft tissue on the residual limb, which must be maintained in a healthy state to preserve the remainder of the limb. It can also lead to other complications such as bone bridging or fractures.

Physical input systems are almost exclusively used as a control strategy for single-DoF body-powered prosthetics. Due to the increased complexity of multi-DoF systems, these are almost always externally powered and require either multiple proportional control signals or some form of grip selection. For both these cases, one of the more advance control inputs described in the following sections will be required.

Irrespective of the type of system being controlled, most commercial HMIs use some form of physical system as an input. This is most often in the form of buttons and touch screens, but joysticks and dials are also very prevalent, particularly in the field of robot teleoperation [27]. In most cases, the input from physical systems is very simple, and the functionality associated with it is extrapolated from dexterous manipulation of the input. One reason NUIs are less dependent on physical interface systems is that in cases where this dexterous manipulation is required, the burden of learning the system is on the user, and their use of the interface becomes a learned behaviour, potentially reducing accessibility. It may be that despite potentially more intuitive and accessible interfaces being available, the perceived effort required to move away from a known system is higher than the general population is willing to undertake, as has been the case with many technologies in the past [13].

While most physical input systems focus on the hands to perform the required action (such as in robot assisted surgery [28]), there are several alternatives that are more aligned with the intent behind of this thesis. HMIs for use in virtual environments are often geographically constrained, and in the case of CAVE (Cave Automatic Virtual Environments), physical input devices may be present either to map body kinematics, to observe interactions within the environment, provide haptic feedback to the individual, or for some combination of all these tasks. These devices are often mounted in a known position, allowing the interaction to be observed precisely from a known reference point. Devices like this have been used in CAVE-based experiments as inputs to simulate completing specific tasks [29].

Wearable exoskeletons are similar in that they provide a direct measure of input, but these systems are referenced to the wearer, not the environment. This technically removes the geographic constraint, although such systems are often too large to be considered pervasive, and therefore are often geographically constrained for practicality. Nevertheless, many systems have been developed for teleoperation of robotic platforms [30], and for use in stationary conditions for applications such as monitoring symptoms such as tremor or rigidity [31], or for rehabilitation in cases where kinematic monitoring provides useful information, such as for stroke patients [32]. Systems have also been developed commercially for motion capture; an example of which is the Gypsy 7 Motion Capture System.

Attempts to increase the applications for exoskeletons and wearable robotic systems usually do so by reducing the number of joints being examined. Wearable goniometers and optical encoders can be used to directly measure individual joint angles, usually for activity classification [33] or limb tracking [34]. Smart materials can also be used to achieve this [35].

2.4.2 Vision-based systems

Camera-based tracking systems are very common in detecting control movements. They are also useful for detecting the context of the control signal, which may also allow the system to perform different actions to achieve different tasks based on the same control signal. The intelligence in these systems can make them easier to use, provided that the context is apparent to the user as well as the device.

Camera-based control systems for prosthetic control are often used in grip selection strategies to perform object recognition. This can allow the system to pick the most suitable grip for the target object. DeGol et al. [36] used a camera located on the palm of the hand to detect objects, and select from 5 grips with an accuracy of 93.2%. Gardner et al. [37] mounted their camera under the forearm of a commercial prosthetic, and were able to classify 6 object for actuation of 2 grip patterns with an accuracy of 69.2% - 90.8%. Dosen et al. [38] introduced different grasp sizes to the grip selection protocol, classifying 9 combinations with an accuracy of 84%. These applications use the cameras to provide context to the user's intent to move. Intent is often picked up using another sensor such as EMG or IMUs. One of the difficulties associated with using grip patterns based on object recognition is the variety of objects that an amputee may be required to hold. This means that a large database of objects and their associated grip patterns is required. Markovic et al. [39] overcomes this limitation by approximating the shape of the target device to a common geometric model, allowing the system to cope with arbitrary and unseen objects. In this case, the camera provided information on size and shape, allowing the controller accurate proportional control of the hand.

An alternative to using cameras that operate in the visual spectrum is to use depth cameras as a method of approximating object shape, and therefore required grip. Ghazaei et al. [40] used this technique to obtain a grip selection accuracy of 88%.

There are a wide variety of camera-based HMIs, but they can broadly be broken down into those that use wearable cameras and those that use stationary cameras. Stationary cameras are most commonly used in HMI applications to either observe full body kinematics (also referred to as Human Pose Estimation or HPE), or a subclass of HPE such as hand kinematics.

There is a large body of research dedicated to HPE for gait analysis and motion capture. The image processing techniques used can be categorized as either marker based or non-marker based [41]. Markers are physical devices that are visually unique to the image processing system. Combinations of these markers can be placed on the body to derive the orientation of the joint. The observed constellation can then be applied to a skeleton to create a kinematic model. Markers are often used in clinical environments to detect subtle differences between normal and pathological locomotion, due to the high accuracy of the advanced systems [42-44].

Marker based systems can be prohibitively expensive, so many applications no longer use them. An alternative is to segment the subject within the image, and then apply pose estimation based on a known humanoid model [45-47]. This approach works well, but makes assumptions based on the model it uses, which can cause problems if those assumptions prove to be false. It has been demonstrated that a model can be generated by observing the movement of the subject [48]. This removes the limitations imposed by a generic model.

The Kinect Xbox 360 from Microsoft is often used to detect body kinematics due to its camera paired with dual infrared depth sensors. Body kinematics detected using a Kinect Xbox have been used to generate control signals for humanoid robots [49-51], however this requires favourable lighting conditions, and also requires the user to stay within the field of view of the camera, limiting the available workspace.

Cameras can also be used to detect hand movements to generate control signals. This is more common, and has resulted in the creation of the Leap Motion controller [52], a commercial product consisting of a desk mounted camera used for gesture recognition for computer interaction. More generally, hand gesture recognition using either the Kinect [53] or a custom camera arrangement have been used for both sign language recognition [54] and computer/robot control. While most systems use skin detection [55] as a starting point for either fitting or generating a model, [56] and [57] both use coloured gloves to identify individual parts of the hand. This approach allows different parts of the hand to be identified more easily, improving the capacity for gesture recognition.

The use of body, or prosthetic -mounted cameras for HMI are typically found in augmented reality applications, such as the HoloLens. They tend to have accurate hand tracking in uncluttered environments [58].

2.4.3 Inertial Measurement Units

Inertial Measurement Units (IMUs) are one of the easiest ways to determine body kinematics. The sensor is placed at a known orientation on the target joint, allowing movement and orientation of that joint to be observed. It is often not necessary to derive the entire kinematic pose, partial subsections of the kinematic tree are all that is required in many applications.

In the field of prosthetics, IMUs are almost exclusively used to provide context to the semi-autonomous control system within the prosthetic. This is normally achieved in one of three ways. The first is through activity recognition, most commonly used in lower limb prosthetics for gait detection. Lower limb powered prosthetics can apply power or resist movement but rely on gait models to do this effectively. The model selection is based on detecting current movement, and IMUs have proved to be an effective way of doing this, both in the research community [59] and in industry [60].

The second method of providing context is through state observation. There are a number of ways this could be implemented, but most commonly it is used in conjunction with control signals taken through other sensing means. Woodward et al. [61] uses a combination of IMU and MMG, where wearers can express an intent through muscle contraction, but that intent can be used to either change grips or actuate

the hand based on the orientation in which the wearer is holding their arm. Gardner et al. and Kyranou et al. [62, 63] use the information provided from the IMUs to supplement their primary control system using the nature/angle of approach to the target to select different grips.

The third method of providing context is to identify sources of interference in the primary control signal and reject commands that may have been generated through that interference. MMG and EMG are both subject to motion induced noise, so using an IMU to detect that motion has been used to detect when a signal is likely to be a false positive [61, 64].

There are several handheld HMI devices that use IMUs as either their primary or secondary input. Kinematic monitoring systems based on IMUs have also been developed for a variety of applications, including robot teleoperation [65], gait recording [66], activity recognition [67, 68], virtual presence and rehabilitation [69-72].

IMUs can be found in most smart phones and smart watches, providing the capability for general activity tracking. Additionally, several companies that use custom, stand-alone IMUs for activity monitoring, such as Fitbit and Athletec have achieved commercial success. Devices that specifically generate control signals based on movement are less common, although Thalmic Labs have created a wearable interface device that used a combination of EMG and IMU data.

2.4.4 Electromyography

Electromyography sensors (EMGs) are the most common sensors for detecting muscle activity. They work by detecting the electrical signal produced during muscle contraction, referred to as Myoelectric Signals (MES). EMGs have a wide variety of uses within the medical industry, and there are several commercial products that also rely on them. The EMG signal can be measured from the surface of the skin (known as surface EMG or sEMG) or using implantable sensors.

There has been a large amount of research into extracting more control signals from EMG sensors. Excluding the On/Off control, there are six other commonly explored methods of myoelectric control: Proportional, Regression, Direct, Finite State Machine, Posture and Pattern Recognition [73].

Proportional control – It has been demonstrated that the amplitude of the EMG signal can be representative of the amplitude of the intended movement [74]. A proportional control structure is one in which the amplitude of the EMG signal is used to set one of more of the mechanical output properties [75] of the prosthetic in a practically continuous scale. The mechanical properties may be velocity, force or position depending on the intended use.

Regression control – It has been shown that using regression-based methods, simultaneous proportional control can be achieved, where multiple independently-controllable analogue control signals can be extracted [76-78]. This allows multiple joints to be controlled in a proportional manner at the same time.

Direct control – Direct control is the term given to a system that attempts to restore functionality by observing individual muscle contractions in the residual limb and then moves the prosthetic as if those

muscles were still connected. Due to electrical crosstalk, this is often achieved using implantable electrodes, although source separation can be used to overcome the crosstalk and attribute a signal to its source muscle group [79, 80].

Finite State Machine – Finite State Machine (FSM) control is relatively common in commercial externally-powered prosthetics. Instead of the On/Off control where the control signal always invokes the same action, in FSM-based controllers the actions are determined from both the control signal and the current state of the device. Additionally, control signals can be used to change the state of the device with no actuation. Commercial devices use a range of inputs to change the current state, ranging from co-contraction, to buttons, to physically changing the device state. FSM-based controllers are also explored in the literature to increase the functionality of the device when the number of observable control signals is limited [81-83].

Posture control – Posture control is similar to regression control, but instead of each channel controlling a separate DoF, they instead weight different hand postures, meaning that the wearer can specify any point representing a combination of the postures [84]. This means that a comparatively small number of templates can be used for a diverse range of applications

Pattern Recognition – As prosthetics have become more functional, the required complexity of the control signals has also increased. The technical limitations of EMG mean that other forms of control that impose a direct control usually have a low dimensionality. Pattern Recognition (PR) allows the prostheses user to trigger autonomous behaviours that are analogous to the control signal. Pre-programmed grip patterns mean that the user must make one control signal (usually an imagined gesture) to begin each gesture. These grip patterns are designed to cover as much functionality with as few different control signals a possible. The user provides high-level commands, and the individual joints are actuated according to the selected grip pattern.

Pattern recognition controllers can have a number of different steps, but they usually consist of segmentation, feature extraction and classification. They can also have additional steps such as pre/post processing and feature selection or a proportional control as a secondary controller. Segmentation is often performed using a combination of signal detection and windowing, and once the signal has been segmented, appropriate features can be extracted.

There is no definitive guide as to which features are most indicative of the gesture being performed, but attempts have been made to rank features. Zardoshti-Kermani et al. [85] performed a classification based on a number of feature sets: the Integral of Absolute Value (IAV), mean Absolute Value(MAV), EMG Histogram (emgHIST), Variance (VAR), v-Order detector (V), Zero-Crossing (ZC), Willison Amplitude (WAMP), Log-detector (LOG) and 4^{th} Order Autoregressive models (AR), and found that the EMG Histogram and v-order detector gave the best overall performance. Other commonly used time domain features include the number of Slope Sign Change [86], Average Amplitude Change (ACC) and the Simple Squared Integral (SSI) [87].

Due to the simplicity and low computational cost, many papers exclusively examine time domain features, however, there has been some investigation into the use of frequency domain and time-frequency domain methods. These methods involve transforming the segmented signal to the frequency domain, either using Fourier analysis [88] or by using a wavelet transform [89, 90]. Nazmi et al. [91] presents a more complete list of time domain, frequency domain and time-frequency domain features. In most cases, the feature vector produced from these transforms has a high dimensionality, and therefore these methods often require dimensionality reduction, either through feature selection or feature projections [73].

Once the features have been extracted, they can be used to classify which gesture the wearer is performing. Again, there are a variety of methods that are employed for the classification of hand gestures using EMG signals including: Artificial Neural Network (ANN) [92, 93] such as Convolutional Neural Network (CNN) [94, 95] and Multilayer Perceptron (MLP) [96, 97], Bayesian Classifier (BC) [98], Decision Trees (DT) [99], Fuzzy Logic (FL) [97, 100], Hidden Markov Models (HMM) [101], k-Nearest Neighbour (KNN) [102], Linear Discriminant Analysis (LDA) [102-105], Quadratic Discriminant Analysis (QDA) [102], Random Forests (RFS) [106] and Support Vector Machines (SVM) [101, 107-109]. It has been shown that the choice of features has a far bigger impact than the choice of classifier, and provided an adequate feature set is being used, most classifiers will have a similar performance [110].

EMG sensors are heavily used in the prosthetics industry and are the primary method of control for externally powered prosthetics. Commercial prosthetic devices usually use either one or two EMG sensors placed on the residual limb. The internal circuitry in these sensors detects gestures through thresholding the myoelectric signal, and then converting this to a control signal that it sends to the prosthetic. Two sensors placed on the arm can control the prosthetic to open or close. Controllers that rely on thresholding in this manner are typically referred to as exhibiting On/Off control.

The signal processing is independent of the end effector, so the techniques described for gesture recognition are also application in the fields of HMI/HCI.

EMG has been used in Human Machine Interface devices both commercially and in academia. Thalmic Labs created a wearable EMG array known as the Myo, which has been used to control a number of smart peripherals, including computers, robotic platforms and quadcopters through gestures. The Myo has also been used in research for controlling wheelchairs [111], translating sign language [112] and for robot teleoperation [113].

The Thalmic Myo used dry electrodes on unprepared skin to record the EMG signals from the arm. Since this has been shown to suffer from a higher and more unstable impedance than either wet electrodes or a prepared surface of the skin [114], many studies opt for more sophisticated sensing equipment. EMG is frequently used to provide qualitative control signals in the field of robot teleoperation, where the

Table 1 - Recent EMG classification studies

Author	Number/Type	Classification algorithm	Number of gestures/ Accuracy/ Online? (y/n)	Ref.
Park et al., 2016	10 / dry electrodes	CNN	6/~92%/n	[2]
Atzori et al., 2016	10 / dry electrodes	CNN	52/~66.59%/n	[4]
Geng et al., 2016	128 / electrodes	CNN	52/96.7%/n	[6]
Purushothaman & Vikas, 2018	8 / dry electrodes	SVM (BC)	15/>95% (>91%)/n	[8]
Teh et al., 2018	6 / dry electrodes	LDA	8/96.53%/y	[9]
Zhang et al., 2019	8 / dry electrodes	ANN	5/98.7%/n*	[10]

* implemented online, but accuracy not reported

continuous control signals are generated through some other means [115-117]. It has also been used to provide a computer interface for users unable to use traditional forms of interface [118].

Table 1 summarises example papers representing typical accuracies achieved in EMG activity classification for pattern recognition applications in recent years.

2.4.5 Mechanomyography

Mechanomyography is the term used for the monitoring of mechanical signals that are generated during a muscle contraction. These mechanical signals are produced through a combination of the gross lateral movement of the muscle during the initial contraction, the lateral vibration generated at the muscles resonant frequency, and the dimensional changes of the motor unit's fibres themselves [119, 120]. It is generally accepted that MMG signals and EMG signals are generated through the same process, as they appear to propagate through the muscle at the same rate [122], however, it has been found that EMG and MMG are affected differently by muscle fatigue [121, 123]. In addition, the signals from MMG have a much higher base signal to noise ratio than EMG, and can therefore be recorded reliably outside the clinical environment [124].

MMG signals are usually recorded by placing a sensor on the surface of the skin, normally either an accelerometer [125, 126], microphone [124, 127] or a piezoelectric contact sensor [128]. It is also possible to monitor the MMG signal externally by using a Laser Distance Sensor [128-130]. It has been shown that the placement of the sensors needs to remain consistent to ensure a consistent signal [131]. Work has also been conducted in the design of the MMG sensors, with the microphone-based systems potentially requiring the most external components. The current design of existing microphone-based MMG was proposed by Silva et al [132] and optimised by Posatskiy et al [133]. These designs are referred to in Chapter 3.

Table 2 - Recent MMG classification studies

Author	Number/Type	Classification algorithm	Number of gestures/ Accuracy/ Online?(y/n)	Ref.
Alves & Chau, 2009	6/Microphone & Accelerometer	LDA	8/ 93.3%/n	[1]
Alves & Chau, 2010	6/Microphone	LDA	8/ 85%/n	[3]
Cao et al., 2011	2/Accelerometer	Generalised Discriminant Analysis	4/95.12%/n	[5]
Ma et al., 2017	6/Microphone	CNN	5/94%/n	[7]

There has been significant work performed on using MMG to study the behaviour of motor units during contraction [134, 135] and in detecting fatigue [136, 137]. Work has also been conducted to detecting neuromuscular disorders [138] and in creating tools for rehabilitation [139]. Much of this work is performed by performing actions that activate individual known muscle groups. The sensors are placed over the body of the target muscles so that the signals are attributable to their source.

MMG is not dependent on transient environmental conditions such as skin impedance. This means that it is potentially well suited to the conditions within an airtight socket, which are likely to change over the course of a day. Despite this, there are currently no commercial MMG systems offered by prosthetic manufacturers, who favour EMG.

There has been some research into the use of MMG for prosthetic control, but it is less mature than its EMG counterpart. As a result, mechanomyographic control systems only fall into four categories: On/Off control [37], Direct control [140], FSM control [61, 62] and Pattern Recognition [3]. The descriptions and methods in these control strategies are the same as their EMG counterparts.

Beyond prosthetics, very few practical applications of MMG have been explored. MMG has been used for gear switching on a bike [141], but a large proportion of the work conducted to this point has been to validate the sensors as opposed to implementation in practical applications. As such, the work of sensor fusion has been limited. Since EMG is considered the standard for muscle activity sensing, work has been performed using both EMG and MMG for validation [142]. Woodward et al [143] used a multimodal approach, combining IMU and a single MMG sensor for intent derivation, however in this application, each sensor provided a single channel of binary data for on/off control. Some examples of papers that have implemented multichannel MMG for pattern recognition are given in Table 2.

2.4.6 Myokinemetric signals

Myokinemetric (MK) signals are recorded by observing the dimensional changes of the muscle, and therefore are a direct measurement of one of the components of the MMG signal. It can be observed by recording the changes in the diameter of the limb, meaning that the signal is a summed measurement of the cross-sectional area of the muscles in the region being measured.

MK signals have been used for prosthetic control, although primarily for the proportional control of a single degree of freedom device [144-146]. One possible reason so little study has been performed on this type of interface for prosthetic control is that amputees often lose muscle mass in the amputated limb, resulting in a smaller myokinemetric signal. Another reason is that sonomyography provides a more detailed view of the same information.

2.4.7 Sonomyography

Sonomyography (SMG) is a more advanced method of detecting the myokinemetric signal. SMG describes the use of ultrasound imaging to provide information about muscular activity, most often by observing the dimensional changes of the muscle [147]. Other features that can be observed include muscle fibre pennation angle and muscle fascicle length. Unlike MK, SMG can be used to examine individual muscle contractions, meaning that more complex control structures can be achieved. SMG has also been used for other applications such as determining fatigue [148] and muscle function assessment [149-151].

SMG has been used to find a correlation between the deformation of the muscle and the joint angle with the intent to provide proportional control of single DoF prosthetic devices [152, 153]. More recently, individual finger flexions/extensions have been distinguished either using ultrasonic imaging [154, 155] or using an array of single element ultrasonic transducers [156, 157]. Outside prosthetics, SMG has not been applied as an HMI for any other applications. A number of papers examine the repeatability of the SMG signals to assess their applicability to HMI problems, but at this point no implementations have been created.

2.4.8 Other forms of interface

There are several other forms of interface that have been used for both prosthetic control and general HMI applications. These following interfaces do not record dexterous gestures but can be used either as a direct method of control, or as a transparent method of providing context to a controller. They are not directly in line with the research aims of this document but could be used to extend the work in the future.

2.4.8.1 Gaze-based interface

Gaze-based interfaces estimate where the user is looking to provide a control signal, most often using a camera focused on the user's eye. They were proposed as an interface for individuals who are unable to use other forms of interface due to injury, stroke or disease [158]. More commonly, gaze-based interfaces are used to determine Areas of Interest (AoI), which allows researchers to explore how people observe their environment, useful for a number of tasks including UI design and advertising [159].

Gaze tracking is normally used in prosthetic control in one of two ways. It has been used to move prosthetic arms [160, 161], but more often it is used for target selection. This use is similar to the head mounted cameras discussed in the vison section but include a second camera for eye tracking to determine where in the field of view the user is looking. This allows for a more robust solution than through vision alone [162]. There is a large body of research on gaze tracking for computer control [163-165].

2.4.8.2 Tongue-based interface

Tongue movements can also be used as a form of interaction. Tongue-based interfaces are also useful for those who are unable to use other forms of interface, and they do not require any external equipment such as a camera. Tongue-based interfaces fall into two categories, intra-oral and intra-aural. Intra-oral interfaces usually consist of pressure sensitive sensors placed on the roof of the mouth or on the cheeks [166, 167]. Intra-aural sensors consist of pressure sensors worn within the ear. It was noted that tongue gestures could be differentiated by looking at the changes of in-ear pressure caused by tongue movements, known as Tongue Movement Ear Pressure (TMEP) signals [168, 169]. Tongue-based interfaces have been used to control wheelchairs [167] and for robot teleoperation [170].

2.4.8.3 Brain Machine Interface

Brain Machine Interfaces (BMIs)/Brain Computer Interfaces (BCIs) provide a method through which a user can provide control signals through cerebral activity. They are usually used as a primary communication interface for individuals who are completely paralyzed, either through injury or degenerative disease. Brain activity detected through BCIs can be the only indication of awareness in some individuals [171], and therefore it is a vital area of study for those creating assistive devices. There are several methods of detecting brain activity for BCI, both non-invasive, such as electroencephalography (EEG) and functional Magnetic Resonance Imaging (fMRI), and invasive, such as Electrocorticography (ECoG) and brain implants.

Cortical activity has been used for the operation of robotic platforms previously [172, 173], however it cannot detect dexterous movements in real time at this point.

2.4.8.4 Voice

Voice User Interfaces (VUIs) are distinct from the other interfaces described here, in that it is the only interface not to monitor some form of (real or imagined) gesture control. VUIs have become increasing popular in recent years. They are included here because they adhere to the principles of NUIs, and commercial solutions are both efficient and have a high user satisfaction. They do not constitute direct competition as they record a distinct form for natural movements. A system that aims for completely generalizable HMI will need to monitor both voice commands and gesture commands.

VUIs for prosthetic control has been implemented as an alternative to the EMG-based solutions [174, 175], and comparative studies have been performed that show that voice control is a more efficient strategy than EMG-based systems [176]. Voice systems combined with EMG have also been tested and shown to be more efficient that EMG alone [177].

Outside the lab, voice-controlled prosthetics have not had any commercial success. Possible contributors to this include the desire not to draw attention to the prosthetic by talking to it, unresponsiveness in noisy environments, the desire to control the device in a way natural to the traditional control strategies, and the concern that others nearby will have just as much control as the wearer will.

VUIs for general HMI applications are becoming more prevalent, with several smart home controllers such as the Google Home and Amazon's Echo devices on the market. Voice assistants like Alexa, Cortana and OK Google are also becoming more popular. Currently, these devices are very suited for tasks that do not require context, such as general inquiries, or specific tasks where there is only one corresponding action. Where VUIs are less useful are for tasks that are difficult to put into words, such as moving a display between several screens.

While VUIs remain a permanent method of smart device control, they do not provide a complete solution in their current form.

2.5 Guidelines for HMI Design

Choosing the most appropriate sensors to include in the HMI requires the designer to consider factors that lead to both high and low adoption rates for HMI devices. These factors have been examined in the fields of prosthetics, assistive devices and in the development of NUI-based devices, so a list can be compiled of those factors that must be considered when building a device that bridges all three fields.

Many studies examining rejection rates for prosthetic devices categorize and rank the factors leading to rejection and those that regular users prioritize. This ranking can be used to derive new design considerations for potential devices, allowing different technologies to be evaluated for their suitability. Similar work has also been undertaken in the other HMI fields, allowing those sensors to be evaluated for suitability as well. From these studies, the following factors have been identified as features that can be used to compare input devices:

- Functionality/Robustness
- Usability/Comfort
- Form/Pervasiveness
- Accessibility/Cost

The information presented in this chapter is summarized according to these features in the following sections.

2.5.1 Functionality/Robustness

Ideally, the sensor should have a high bandwidth, capable of capturing a large range of movements. It should not be susceptible to environmental factors and should be independent of transient calibration values.

- **Physical input systems** – Physical input systems are very robust. The functionality of a physical input system is dependent on the number of individual physical input components (e.g. Buttons, dials, joysticks, etc). While most interfaces have many of these components, in both assistive technologies and prosthetic devices, the number of components tends to be much lower. A split hook device may be operated through a cable-based system, allowing a single analogue channel

of data to be collected. Physical input systems can therefore be classified as medium functionality with a high level of robustness.

- **Vision-based system** – Vision-based systems have been explored as interfaces for both highly dexterous movements and for capturing full body pose information. They are also capable of gathering not only control signals, but the context of those commands, a vital component in the implementation of NUIs. They are reasonably robust under ideal conditions, but performance is affected by environmental factors such as light level. Additional functionality can be achieved by increasing the complexity of the hardware (e.g. inclusion of infrared information). Assistive technology makes use of vision-based systems in situations where physical input systems are not practical, but in prosthetics their use has been limited to research so far.
- **Inertial Measurement Units** – IMUs provide the ability to capture motion gestures. They broadly fall into two categories, those where the IMU is an interface device (something that the user moves to create control signals) and those where it is a monitoring device (something that derives control signals or contextual information by observing the natural movement of the user). As such, their functionality can be high, however they can be susceptible to a number of interfering factors, including vibration and magnetic fields. Though they are often used in assistive technologies and prosthetics, it is rarely as a primary interface. Instead, they are often used to monitor the environment for signals that may interfere with other sensing technologies.
- **Electromyography** – EMG allows muscle contractions to be observed, which can allow both real and imagined movements to be inferred. Again, this can be used either to detect specific control signals or to observe the wearers muscle activity. The functionality can be high, with the ability to detect both dexterous movements and muscle condition, however it is reliant on favourable environmental conditions. Skin impedance and temperature, pressure on the sensors and movement of the limbs all affect the EMG signal. In controlled environments, EMG is both robust and functional, and is used for diagnostic purposes, although the functionality decreases dramatically when used in the uncontrolled environment of a prosthetic socket.
- **Mechanomyography** – MMG also allows muscle contractions to be observed. It has been shown as a comparable signal to EMG, although it is a far less advanced area of research, so it is too early to say whether it can achieve the functionality that EMG has demonstrated. It is unaffected by environmental conditions such as impedance, humidity and skin temperature, although it is still affected by pressure and movement. MMG has not been used either clinically or commercially at this point since EMG-based interfaces became standard practice for medical diagnostics.
- **Myokinemetric signals** – MK signals provide muscle information at a much lower resolution than any of the other muscle sensing technologies described here. At this point they have been used to provide a single control signal for a single degree of freedom device, and so the functionality is low. The robustness is reasonable, but pressure on the sensors or passive movement of the sensor will both lead to false signals.
- **Sonomyography** – SMG provides high resolution information on the muscle activity within the arm. It can detect individual muscle movements, and therefore can give a clear picture of the task the user is trying to achieve. The effect of environmental conditions is limited to those that may

take the sensors out of contact with the skin, and when contact is maintained the signal is very robust.

2.5.2 Usability/Comfort

The ideal system needs to be easy to use and should aim to use control signals that the user is already familiar with, whether this is through the use of other devices or through natural movement. Long-term use may require recalibration of some of the sensors, and this is factored into the rating for ease of use. It should also be comfortable to use for long periods of time. The comfort level refers to both the physical comfort of the device and the comfort of long-term use.

- **Physical input systems** – Physical input systems rate highly in terms of usability. The individual components are easy to learn how to operate and have the addition benefit that physically manipulating something results in haptic feedback, which allows the user to know that they have completed the movement. The long-term comfort associated with physical input systems is poor however. Simple interfaces such as the computer keyboard and mouse are associated with RSIs in the hands and wrist, and the more force required to operate the interface the greater the risk of injury. To operate a prosthetic, the user must apply the force they wish the prosthetic to exert through their residual limb, which has been shown to have long-term health problems associated with it. The comfort therefore receives a low rating.
- **Vision-based system** – Vision-based systems are very easy to use since they capture natural movements, and often do not require the user to do anything that might be detrimental their comfort. They often rely on visual feedback, which means that they can be slightly harder to use then physical input systems due to the extended period between action and feedback. Vision-based systems rarely need recalibration within a known environment.
- **Inertial Measurement Units** – IMUs can be manipulated very easily, since they record the physical movement of the device. As an interface, they are very easy to use and the output from them can be predicted accurately by the user, allowing them to provide a form of interface that is easy to operate. If the system is being worn, modern IMUs are small and lightweight, and can record activity without inhibiting the wearer in any way. This makes them comfortable to use for extended periods of time. IMUs do occasionally require recalibration, which may be difficult for a user who is unfamiliar with the technology.
- **Electromyography** – EMG sensors have the potential to create very easy to use interfaces. The ability to detect hand gestures and other natural movements means that the user does not have to learn specific action, but instead can perform actions that they are already familiar with. However, due to the robustness of the sensors, EMG-based interfaces for prosthetic control normally rely on deliberate contractions of muscle groups. This leads to a less natural interface since the wearer must learn to associate these deliberate contractions with the intended output. Additionally, the sensors must remain in consistent contact with the skin, which is normally achieved by increasing the pressure on the sensors. Since the calibration of these sensors is dependent on environmental conditions that change constantly, the signal detection threshold

for systems such as these is normally high. This means that the user must focus on generating a large muscle contraction to create their control signal, and this can quickly lead to fatigue and discomfort.

- **Mechanomyography** – MMG is similar in terms of initial ease of use to EMG. Interfaces using MMG that are currently being researched do not rely on generating specific signals but monitor natural movements that can then be used to generate control signals. Long-term usability has not yet been tested, but in short term tests it has been reported that the muscle contractions needed to generate control signals are lower than those necessary for EMG, and therefore more commands can be generated before the wearers fatigue level become uncomfortable.
- **Myokinemetric signals** – MK signals rely on voluntarily changing the dimensions of the limb, and therefore large muscle contractions are required to reliably generate discernible MK signals. While this does make the system relatively easy to use it can quickly lead to fatigue. In addition, it normally requires a tight band to be work around the limb, which could become uncomfortable over time. The resistance of the band does provide a form of feedback to the wearer, so they can quickly learn how to use the system.
- **Sonomyography** – SMG can easily determine intent from the user, and so can be used as a fast form of interface. It relies on natural movements and does not require the user to increase the strength of a movement to detect it. It does require some pressure over a larger area of the limb than many sensors and is the largest and heaviest of the sensors listed here.

2.5.3 Form/Pervasiveness

- **Physical input systems** – Physical input systems come in a huge variety of forms and can be either wearable or a separate object that the user interacts with. Most interfaces are comprised of several components that can be operated individually and rely on some form of dexterous manipulation to generate the control signals. Simpler interfaces designed either as assistive devices or as prosthetic controllers are usually much larger and may span a number of joints. Physical input systems are pervasive, but normally either inhibit the user in some way (such as resisting movement) or require deliberate and cognitive engagement (such as pressing a button).
- **Vision-based system** – Vision-based systems can be placed in a defined environment to provide information about both the environment and the user within it. While the cameras themselves can be small, the user must remain in the Field of View (FoV) of the camera to allow it to be used. This can significantly restrict the user's freedom to move, and they must be aware of both the limits of observation and the occlusion within the FoV, and as a result the user must be aware of the location of the camera, adding to their cognitive load. Cameras can also be worn to provide control signals. Applications using wearable camera often require them to be mounted out from a limb so they can observe the end of that limb, on the head to observe where the user is looking or facing towards the users face to observe eye movements. In most applications, these require relatively large housings to hold the cameras, inhibiting free movement and adding weight.
- **Inertial Measurement Units** – IMUs are low volume sensors that can be entirely self-contained. They measure the forces applied to them internally and have no external moving parts or sensing

surfaces. As a result, objects containing IMUs can be designed in almost any form that the user desires. In addition, their low power consumption can result in small stand-alone modules that can be used to monitor movement of a wearer with negligible additional weight.

- **Electromyography** – EMG signals can be measured using either wet or dry electrodes. Wet electrodes use a gel to maintain the local conditions of the skin, which allows a reliable signal to be obtained for as long as the gel is stable. Dry electrodes normally have three contacts that must be kept in place on the skin. They do not maintain their local environment but degrade at a much slower rate than wet electrodes. EMG sensors are pervasive, they can be used to infer intent and movements by measuring the signals produced during the generation of the movement, rather than measuring the movement directly. They do not inhibit movement but can be larger than some alternatives.
- **Mechanomyography** – The two most common types of MMG sensors are accelerometer-based and microphone-based. Both have a comparatively small footprint and depth. MMG sensors can also be used to infer activity and generate control commands from muscle activity instead of direct observation of kinematic movement, and so are also pervasive. They also tend to be lightweight and low power. Microphone-based MMGs normally contain an acoustic chamber, which limits their minimum dimensions.
- **Myokinemetric signals** – MK signals are observed through a band placed around the limb. The band does not inhibit movement, and while the footprint is normally larger than most alternatives, the depth can be small. It is highly pervasive.
- **Sonomyography** – SMG signals are currently most commonly observed using an ultrasonic imager. An ultrasonic probe is worn using custom housings that hold it perpendicular to the surface of the limb. The probe normally protrudes a long way from the housing, since a large amount of damping material is required in order to stop echoes from the back of the sensor. In addition, a large amount of processing power is required to turn the signal data into an image that can be used to identify muscle activity, and so in most cases the sensor needs to be plugged directly into a computer. While SMG does measure muscle activity to infer control commands in the same way EMG, MMG and MK devices do, the size, weight and form of the sensor does count against it in terms of pervasiveness.

2.5.4 Accessibility/Cost

- **Physical input systems** – Physical input systems are among the most diverse group of input devices. They tend to be low material cost, and in many cases contain a single electromechanical device, and no active components. They can be made into forms to suit their function, whether they are intended to be used for fine control, as an assistive devise or as a prosthetic control system. There are very specific cases where physical input systems are completely inaccessible, but for the majority of users, some form of system could be utilized.
- **Vision-based system** – Vision-based systems are also accessible. In most cases, a vision-based system could be used as an alternative to physical input systems, since they can observe the physical movement with a high level of accuracy. One of the reasons these systems are less

common is due to the cost associated with them, both monetary and in terms of processing power. In some situations where a user may be unable to make the movements required to operate a physical interface, a camera-based interface can be used instead.

- **Inertial Measurement Units** – IMUs vary hugely in price, but for HMI applications, low end IMUs are normally sufficient. These IMUs have a medium level component cost in the context of the sensors described here. Much like physical input systems, they can be used by most of the potential user population as either a primary interface sensor or to provide some context to the system as a whole. Assistive technologies also often make use of IMUs for activity monitoring, rehabilitation and symptom monitoring.
- **Electromyography** – EMG requires volitional muscle contraction, either as a control signal or through movement. It also requires good electrical contact to the skin, which can be impeded if the individual has a large amount of scar tissue over the muscles that are to be observed. This is a significant problem in prosthetics, and leads to the amputees having to generate larger, more fatiguing contractions as control signals. EMG sensors require electrodes and active circuitry to help remove noise from the signal. As such, they are a comparable price to the IMUs described above.
- **Mechanomyography** – MMG sensors also require volitional contractions to generate the control signals, however MMG signals are much less affected by factors such as scar tissue. They also have the capability to pick up smaller contractions that may be indistinguishable from noise using other sensors. MMG sensors are very cheap to produce, and most systems use a single component for signal capture. MMG data also require little pre-processing before classification can occur, reducing the need for other components.
- **Myokinemetric signals** – MK signals require large muscle contractions, which may be problematic for amputees who are suffering from muscle atrophy, as well as individuals suffering from neuromuscular disorders. Healthy subjects should have less difficulty in generating the required signals. The cost of these sensors is also low, and the sensors can be constructed from inexpensive passive components if required.
- **Sonomyography** – SMG signals give the best indication of the activity of the body of the sensors listed here. They can observe muscle contractions that are not detectable at the surface of the skin, and therefore can be used by the greatest proportion of the population. SMG sensors requires ultrasonic imaging, the equipment for which is a minimum of an order of magnitude more expensive that the IMU/camera-based systems.

The rankings given to each of these sensors in the different categories are shown in Figure 2.

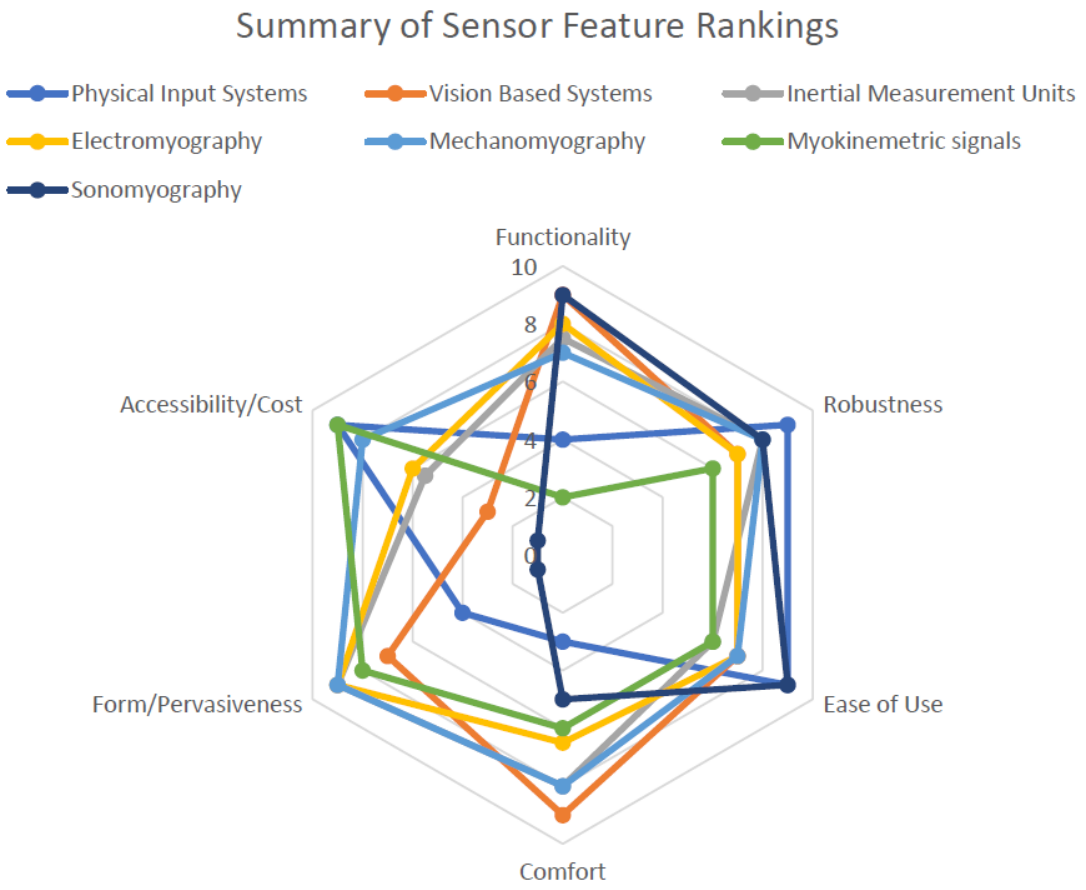

Figure 2 – Comparison of features of different sensing modalities

2.6 Overview of Orientation Estimation

Orientation estimation allows a system to approximate its orientation within a known reference frame. In uncontrolled environments, this is usually achieved through the use of an IMU or MARG (Magnetic, Angular Rate and Gravity) sensor, which observes internal forces generated through a combination of movement, geomagnetic forces and gravity. The applications for this nature of information are varied, but include medical diagnostics/rehabilitation [178-180], health/sport tracking [181], localization [182-185], aerospace applications [186, 187] and autonomous robotics [188-191], as well as the HMI, assistive device and prosthetics applications described in the previous section. These devices should ideally be small, low cost packages capable of determining orientation without environmental knowledge.

MARG sensors typically consist of a tri-axis accelerometer, a tri-axis gyroscope and a tri-axis magnetometer. Orientation estimation is performed by fusing the information gathered by these three sensors. Gyroscopic data can be used to record changes in orientation through integration [192], however this requires a known start point. Additionally, gyroscopic integration is susceptible to cumulative errors introduced through noise and high frequency rotations, leading to a decrease in accuracy over time. The accelerometer and magnetometer can be used to correct these errors. Accelerometers are sensitive to gravitational acceleration, which represents a consistent direction in the global reference frame. Similarly, the magnetometers are sensitive to the earth's geomagnetic field, which is also assumed to be consistent in the global reference frame. Both the accelerometer and the magnetometer are sensitive to other sources of acceleration and magnetic field respectively and can both also be distorted be sensor bias and so all three sensor modalities are used in the most accurate orientation estimation algorithms.

The most prominent techniques for MARG sensor fusion are the complementary filter, the Kalman filter and the optimization filters such as the Madgwick and Mahony algorithms [193].

Complementary filter – The complementary filter is arguably the simplest of the filters listed here and works by applying weights to the gyroscopic integration and accelerometer output to provide an estimation for orientation. The errors in gyroscopic integration result from a combination of low frequency bias and drift, whereas the errors in the gravitational estimation are the result of high frequency noise and motions. Applying a large weight to the gyroscopic data and a small weight to the acceleration data will effectively allow the sensors to correct each other. More specifically, the weights are derived from the frequency boundary between the signal and the noise. In more advance complementary filters, these weights can be adapted in real time depending on the task to be achieved [194].

- **Advantages** – the filter is very easy to design, and computationally efficient, meaning it can be run on very basic hardware. It does not require previous sensor inputs to perform its estimation.
- **Disadvantages** – the filter is slow to converge, and since it has no active method of compensating for sensor noise, it can be less accurate than the alternatives.

Kalman filter – the Kalman filter was designed to counteract the effects of noise on linear estimation problems [195]. As such, it is not exclusively an orientation estimation algorithm, and can be applied to a wide range of problems. It functions as a recursive algorithm, applying the current input data, the previous

input data and the previous output state prediction to a recursive function to predict the current output state. Since movement is a non-linear problem, Extended Kalman Filters (EKF) must be used [196, 197]. Kalman filters are often used in commercial orientation monitors, including products designed by Intersense [198], VectorNav [199], xsens [200], micro-strain [201] and PNI [202].

- **Advantages** – EKFs tend to provide the most accurate estimations for orientation. They are also strong in dealing with a significant level of noise, making them suitable for applications such as quadcopter control.
- **Disadvantages** – The two biggest arguments against the use of EKFs are the difficulty in implementation and the high computational cost.

Optimization filters – Optimization filters are a computationally lean alternative to Kalman Filters. They work by estimating their orientation based on the gyroscopic integration and comparing expected sensor output to actual sensor output. The difference can be expressed in the form of an error, which can then be corrected. There are two commonly used algorithms that use this approach for orientation estimation, the Madgwick algorithm and the Mahony algorithm. Madgwick used two sequential gradient descent algorithms to correct the orientation estimated based on both the accelerometer and the magnetometer [203]. Mahony used a PI controller to achieve similar results [204]. Due to their high accuracy and ease of implementation, The Madgwick and Mahony algorithms are often used together when benchmarking other orientation estimation algorithms [190, 205-207]. Admiraal has presented an improved formulation of the original gradient descent algorithm [208].

- **Advantages** – The Gradient Descent Algorithm (GDA) approach has been shown to achieve similar levels of accuracy to EKF's, with a greatly reduced computational cost [209]. The algorithm is also open source and is easy to adapt for specific activities.
- **Disadvantages** – The algorithm does not decouple pitch and roll from magnetic interference, which results in changes in yaw having unwelcome effects in the other axes [210]. Additionally, while easier to implement than Kalman Filters, the algorithm does have a higher computational cost than is desirable on embedded systems. Finally, the presence of two non-perpendicular, sequential gradient descent steps impacts the convergence speed of the algorithm.

2.6.1 Introduction to Quaternions

There are a number of ways of representing rotations, but quaternions have been shown to be the most stable [211]. Quaternions are a four dimensional number system first described by Hamilton [212] between 1844-1850. Quaternions provide a consistent rotation and avoid gimbal problems, and so are often used in computer graphics and IMU/MARG sensors.

Quaternions are represented in the form $\boldsymbol{q} = w + x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k}$, where w, x, y and z constitute real numbers, and $\boldsymbol{i}$, $\boldsymbol{j}$ and $\boldsymbol{k}$ denote imaginary units. Quaternions extend the definitions for imaginary

numbers so that $i^2 = j^2 = k^2 = ijk = -1$, $\boldsymbol{ij} = \boldsymbol{k}, \boldsymbol{jk} = \boldsymbol{i}, \boldsymbol{ki} = \boldsymbol{j}, \boldsymbol{ji} = -\boldsymbol{k}, \boldsymbol{kj} = -\boldsymbol{i}$ and $\boldsymbol{ik} = -\boldsymbol{j}$. It is therefore useful to define q as:

$$q = [q_w, q_x, q_y, q_z] \tag{1}$$

This extension can be used to derive the non-commutative product of two quaternions, which simplifies to:

$$\begin{aligned} \boldsymbol{q}_1 * \boldsymbol{q}_2 = {} & w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2 \\ & + (w_1 x_2 + x_1 w_2 + y_1 z_2 - z_1 y_2)\boldsymbol{i} \\ & + (w_1 y_2 - x_1 z_2 + y_1 w_2 + z_1 x_2)\boldsymbol{j} \\ & + (w_1 z_2 + x_1 y_2 - y_1 x_2 + z_1 w_2)\boldsymbol{k} \end{aligned} \tag{2}$$

The conjugate of a quaternion is defined as:

$$\boldsymbol{q}^* = w - x\boldsymbol{i} - y\boldsymbol{j} - z\boldsymbol{k} \tag{3}$$

which allows the norm to be calculated using:

$$\|\boldsymbol{q}\| = \sqrt{\boldsymbol{q}\boldsymbol{q}^*} \tag{4}$$

The reciprocal, which represents the reverse of the given rotation, is defined as:

$$\boldsymbol{q}^{-1} = \frac{\boldsymbol{q}^*}{\|\boldsymbol{q}\|} \tag{5}$$

Through the use of the reciprocal, it becomes possible to rotate a vector by a quaternion. This is performed by converting the vector ($\boldsymbol{u}$) into a quaternion ($\boldsymbol{v}$) with the *w* component set 0, as per:

$$\boldsymbol{v} = 0 + u_x\boldsymbol{i} + u_y\boldsymbol{j} + u_z\boldsymbol{k} \tag{6}$$

and then calculating the points describing the new position, contained in $\boldsymbol{v}'$, as per [213], which states that:

$$\boldsymbol{v}' = \boldsymbol{q} * \boldsymbol{v} * \boldsymbol{q}^{-1} \tag{7}$$

An equivalent process could be to use a rotation matrix that represents the same rotation as the quaternion, which can be defined as R_q. As a result, it can be stated that:

$$\boldsymbol{q} * \boldsymbol{v} * \boldsymbol{q}^{-1} = R_{\boldsymbol{q}} * \boldsymbol{v} \tag{8}$$

The relation between rotation matrix R_q and the quaternion q can be given by:

$$R_q(q) = \begin{bmatrix} q_w^2 + q_x^2 - q_y^2 - q_z^2 & 2q_xq_y + 2q_wq_z & 2q_xq_z - 2q_wq_y \\ 2q_xq_y - 2q_wq_z & q_w^2 - q_x^2 + q_y^2 - q_z^2 & 2q_yq_z + 2q_wq_x \\ 2q_xq_z + 2q_wq_y & 2q_yq_z - 2q_wq_x & q_w^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{9}$$

N.B.

For normalized quaternions:

$$q_w^2 - q_x^2 - q_y^2 + q_z^2 = 2(q_w^2 + q_z^2) - 1 = 1 - 2\left(q_x^2 + q_y^2\right) \tag{10}$$

This can be proved using:

$$1 = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2} \tag{11}$$

Squaring gives:

$$1 = q_w^2 + q_x^2 + q_y^2 + q_z^2 \tag{12}$$

Therefore (12) can therefore be used to convert between the forms given in (10).

Finally, since orientation estimation algorithms can require the conversion between axis-angle representations and quaternions, the definition is given in [214], and is included here. For a rotation of θ around and arbitrary axis defined by the unit vector $u = u_x + u_y + u_z$ is given by:

$$\boldsymbol{q} = e^{\frac{\theta}{2}(u_x i + u_y j + u_z k)} = \cos\frac{\theta}{2} + \left(u_x i + u_y j + u_z k\right)\sin\frac{\theta}{2} \tag{13}$$

2.6.2 Gradient Descent Algorithm – In-depth Review

The basic gradient descent algorithm proposed by Madgwick in [203] contains two distinct steps. Orientation is maintained in the form of a quaternion and is updated by each step through each iteration.

The first step uses the gyroscopic integration to approximate the rotation that has occurred in the last time step. Provided the sampling rate of the gyroscope is high enough with respect to the frequency of the movement, this gives a good approximation for the rotation between time steps. This can therefore be stated as:

$$q_t = q_{t-\Delta t} + \omega \Delta t \quad (14)$$

where ω is the output from the gyroscope and t represents time. The algorithm then uses a second step to correct for any cumulative errors that may be generated through this process. It does this by providing a partial correction to the quaternion, which is representative of a step towards a calculated 'correct' orientation, derived using the accelerometer and the magnetometer. It assumes the following:

- Acceleration due to gravity in the global reference ($\boldsymbol{v}_{r(a)}$) frame is given by:

$$\boldsymbol{v}_{r(a)} = [0,0,-1] \quad (15)$$

- The accelerometers only detect gravity, and not acceleration due to motion (this is only assumed in this step and not in the complete algorithm since the high frequency noise induced by movement in the accelerometer is effectively filtered by the small size of the corrective step).
- The geomagnetic force $\boldsymbol{v}_{r(m)}$ is given by:

$$\boldsymbol{v}_{r(m)} = [\cos\theta\,, 0, -\sin\theta] \quad (16)$$

  where θ is the local level of magnetic inclination. This value varies globally and is therefore often assumed to be 0.

- The magnetometers only detect the geomagnetic force and are not affected by external interference.

To estimate the rotation between steps, these reference vectors $\boldsymbol{v}_r$ can be converted from the global to the local reference frame. The error in the rotation is then proportional to the difference between the rotated reference vector and the measurement of that vector ($\boldsymbol{v}_m$). This allows the orientation estimation to be expresses as a minimization problem in the form:

$$\min_{\boldsymbol{q}} f(\boldsymbol{q}, \boldsymbol{v}_r, \boldsymbol{v}_m) \quad (17)$$

Where: $$f(\boldsymbol{q}, \boldsymbol{v}_r, \boldsymbol{v}_m) = \boldsymbol{q}^{-1} * \boldsymbol{v}_r * \boldsymbol{q} - \boldsymbol{v}_m \quad (18)$$

The error vector generated by this function can then be minimized using gradient descent:

$$\boldsymbol{q}_t = \boldsymbol{q}_{t-1} - \alpha \nabla_{\boldsymbol{q}} F(\boldsymbol{q}) \quad (19)$$

Where $$\nabla_{\boldsymbol{q}} F(\boldsymbol{q}) = J_{\boldsymbol{q}}(\boldsymbol{q}^{-1} * \boldsymbol{v}_r * \boldsymbol{q} - \boldsymbol{v}_m)^T * (\boldsymbol{q}^{-1} * \boldsymbol{v}_r * \boldsymbol{q} - \boldsymbol{v}_m) \quad (20)$$

And α is the step size.

This process is performed twice, once using the accelerometer data ($\boldsymbol{v}_{r(a)}$ and $\boldsymbol{v}_{m(a)}$), and once using the magnetometer data ($\boldsymbol{v}_{r(m)}$ and $\boldsymbol{v}_{m(m)}$) to achieve consecutive steps down the two gradients, which forms the second corrective step.

2.7 Chapter Summary

This chapter aimed to provide an overview of the current literature pertaining to both Human-Machine Interfaces in general, and specifically those associated with prosthetic control and assistive technology. It first provides a description of the most common commercial technologies and gives some of the reasons why they are not sufficient for a number of the intended end users. In the field of HMI this includes both the required dexterous manipulation that some may struggle with, and the intrusive and application specific nature of current interfaces that are not necessarily conducive to natural use. It has been identified that only around 34% of people who could potentially use a powered prosthetic do so, and this is partly due to the difficulty of use and partly due to the lack of functionality, both features of a limited interface. Alternative interfaces often contribute to medical problems, most commonly RSIs but also potentially soft tissue damage or other complications.

This chapter then provides an overview of specific human machine interface modalities, including physical input systems, inertial systems and vision-based systems, muscle-based input systems, as well as several possible complementary interface technologies. EMG is also explored in depth, since the signals are potentially comparable, and there is a significantly larger body of literature that could potentially inform MMG signal analysis. In particular, the type of control that EMG can provide are described, with a focus on pattern recognition. Several methods of feature extraction and pattern recognition that have previously been used for EMG pattern recognition are detailed.

Previous MMG work is explored, giving an indication of the current development of the technology. Literature describing both previous validation studies of microphone-based MMGs and MMG design optimisation are identified. When examining the literature, a lack of MMG-based sensor fusion was identified. Studies which use a multimodal approach processed each channel individually to provide binary information, which was then used for intent derivation. A system which implemented more in-depth fusion between multiple MMG sensors and IMUs could not be found.

In order to provide some method of comparing the various interface technologies, four criteria that the end users commonly assess these technologies were identified. Each of the seven major interface technologies were then subjectively assessed against these four criteria, and a brief summary of how each of them performed was given. Finally, each interface technology was given a ranking. No single technology definitively outperformed the others, and therefore this ranking provided an indication of areas where each technology could be improved.

The final two sections of this chapter are intended to give a mathematical background to the orientation estimation algorithms that are used in the pervasive monitoring system described in this thesis. Most orientation estimation algorithms use some sort of sensor fusion to correct the errors that can be introduced when relying on a single sensor. There are several methods of performing this sensor fusion, and the methods that provide the greatest accuracy for the least computational expense are optimization methods that typically work by minimizing error functions. Madgwick's gradient descent algorithm for

orientation estimation is a commonly used example, and an in-depth description is provided along with the quaternion mathematics on which it relies.

Chapter 3

Hardware and Software Development

3.1 Introduction to Hardware Development

A sensor suite capable of evaluating user intent was required as the input for this control system. As discussed in the previous chapter, gesture-based control was chosen as the interface modality. When determining intent for a control system, it is insufficient in most cases to only classify a gesture into discrete commands. It is difficult to derive meaning from a gesture without also knowing the context in which it was made. A sensor suite capable of capturing both gesture and context is therefore required to create an appropriate interface. As a prosthetic controller, the system also needed to be capable of capturing imagined gestures.

3.1.1 Key User Requirements

Through discussion with the end users for the prosthetic controller, it was determined that a wearable solution would provide the most convenient level of usability. The requirements for a wearable sensor capable of performing both gesture recognition and being context aware were defined as follows:

- **Non-Restrictive** – the sensors should not inhibit the users in their range of movement.
- **Unobtrusive** – the sensors should be small and lightweight so that they are not uncomfortable.
- **Easy to use** – the sensors suite should not require preparation or calibration before it can be used.
- **Consistent** – the sensors should provide a predictable response.
- **Low-Power** – the sensors should be low power and include their own power source.
- **Wireless** – the suite should not require a wired connection to a computer to function.
- **Internal Storage** – the device should have the ability to record data or save settings when it is not connected to a computer.
- **Configurable peripherals** – the suite should be capable of capturing the signals from peripheral sensors, such as the MMGs, and providing real-time fusion.

Due to the potential of the sensors identified in the previous chapter, combined with the current limited attempts to exploit them, a combination of Magnetic, Angular Rate and Gravity (MARG) sensors, and Mechanomyography was chosen as the principal sensing technique. MARG sensors provide the context of the gesture by monitoring changes in orientation using a 3-axis gyroscope. They can also use accelerometers and magnetometers to observe gravity and the earth's magnetic field when the device is stationary. The MARG sensors can therefore be used to estimate orientation in a global reference frame. Mechanomyography was chosen as the method of monitoring muscle activity (from either an intact or residual limb), as it offers the most reliable signal over extended periods of time without the need for skin preparation or sensor replacement. Mechanomyography is also more suited to the environment inside a socket than other muscle sensing methods, such as EMG. Cameras, exoskeletons and controllers were excluded because they limit the geographical location or the range of movements that the user is able to perform and were not conducive to the creation of a generalised pervasive monitoring system.

As the mechanomyography signal falls within the range of 2-125Hz, the sample rate for the MMG needed to be at least 250Hz to avoid aliasing. The low-power and wireless communication requirements suggest that a simple, off-the-shelf communication protocol such as Bluetooth could be suitable. Due to the device limit in a Bluetooth piconet, it was desirable that the MMGs and the MARG data were connected to the computer through a single Bluetooth connection, therefore externally powered ADC (Analogue to Digital Converter) ports were identified as a derived requirement.

There are a wide variety of MARG sensors available on the market, ranging from low cost breakout boards through mid-range self-contained devices to high cost commercial solutions. The requirements for the device were compiled and the available devices were assessed for their suitability. A commercial solution that met all the requirements could not be found at an acceptable price point; therefore, the decision was made to design and build a device capable of sampling the sensors at an acceptable rate. While there are several methods of obtaining MMG data, there are no commercially available solutions to do so. As a result, the MMG sensors used in this project were also custom built.

3.2 NUIMU Development

3.2.1 Hardware Design

The hardware was designed specifically for this investigation and will be referred to in this document as NUIMU. The board has six Integrated Circuit (IC) elements, which can be seen in Figure 3:

- Microcontroller (PIC24FJ64GA104)
- Wireless Module (BT900)
- UART-USB converter (FT232R)
- IMU (LSM9DS0)
- Power Management (LP2985 and MCP73831)

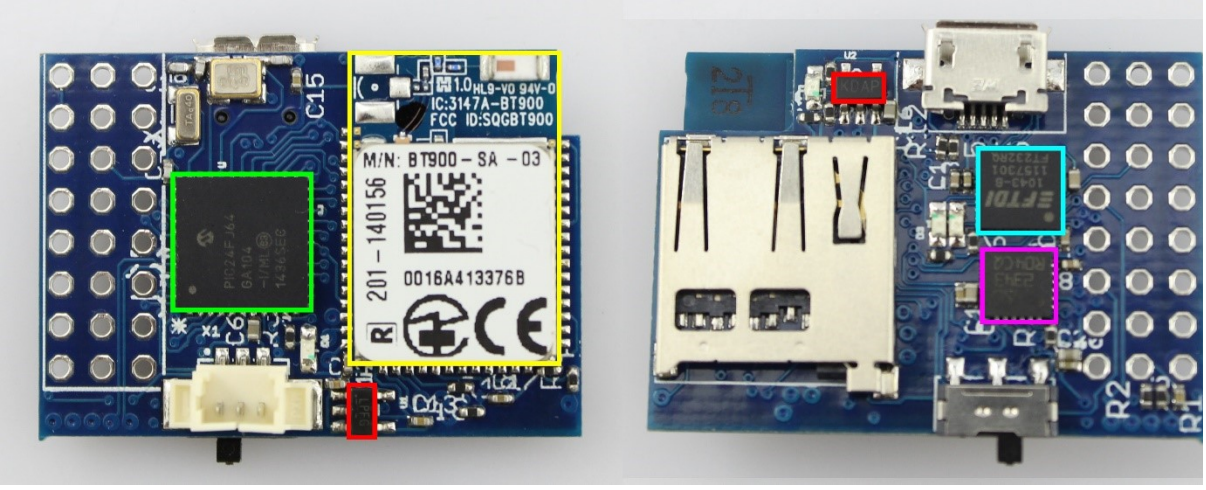

Figure 3 – Breakdown of IC elements on NUIMU boards

3.2.1.1 Microcontroller

The PIC24FJ64GA104 is the primary controller on the board. It is a 16-bit microcontroller from Microchip with a processor speed of 16 Million Instructions per Second (MIPS), and is programmed in the C language. It was chosen because it contained all the peripheral modules that were required for this application. When the circuit is powered, PIC24 sets its internal registers to their required initial state. This includes setting up the pins for communication and mapping the correct interrupts to specific functions within the code. The microcontroller then waits for one of several commands to arrive through one of the two UART communication modules.

3.2.1.2 Wireless Module

A wireless connection to the host device was defined as one of the key user requirements. Bluetooth was chosen over the alternatives due to its relatively low power consumption, and its ease of implementation. The Bluetooth module chosen for this project was a BT900, produced by Laird. It is a Bluetooth v4.0 dual mode module, providing the highest level of flexibility for the design at the time. This module was chosen for its ability to run dedicated code, allowing the module to exercise complete control over the Bluetooth connection, resulting in a faster boot for the device. The module is programmed in smartBASIC to operate a Serial Port Profile (SPP). This profile is the most suited to transferring bursts of data, such as a packet of inertial data. It communicates with the microcontroller through UART with a high baud rate.

3.2.1.3 UART-USB Converter

A form of wired communication was also implemented as a secondary method of interacting with the host device. It allows the Bluetooth module to be reprogrammed through the USB, and for debugging the microcontroller without having to set up a wireless connection. The IC chosen for this was an FT232R from FTDI, as it handles the entire USB protocol and is powered through the USB, therefore it does not draw from the battery resulting in a longer operating period between charges. It is also programmed so that the 'friendly name' of the device identifies the board ("NU USB XXX" where XXX is the NUIMU UID).

3.2.1.4 Inertial Measurement Unit

The IMU chosen was an LSM9DS0 from STMicroelectronics, which contains 3-axis Accelerometer, a 3-axis Gyroscope and a 3 axis Magnetometer, providing many references for the movement of the device. The sample rates these individual sensors are configurable. The range of each sensor can also be set from a list of options. The module is configured and sampled by the microcontroller over a Serial Peripheral Interface bus (SPI). It was chosen primarily due to the large number of reference outputs, small footprint and low cost.

3.2.1.5 Power Management

There are two power management ICs on the board. The first is a 3.3V voltage regulator (LP2985), which operates to ensure that the voltage provided to the ICs on the board is constant. This provides a known voltage for the Analogue to Digital converters on the board, allowing the voltage to be converted into useable data. By using this known voltage combined with a potential divider circuit on the battery line,

the voltage from the battery itself can be calculated. In the LiPo batteries used in this project, this can be used to estimate the remaining battery life.

The second power management IC is a battery charge management controller (MCP73831). This IC is connected to the power line of the USB port, and is used to safely charge the battery connected to the board. It powers an LED that is lit when it is charging and goes off when the battery has reached the defined maximum charge.

3.2.1.6 Auxiliary Ports

The board has 8 powered (3 pin) analogue ports to connect to external sensors. Each port has one data pin, which is connected to a unique channel of the 10-bit High-Speed Analogue/Digital Converter module in the microcontroller.

Application specific firmware can be written to give these ports other functions. All eight of them can be used as digital inputs or outputs. In addition, six of them can be connected to the remappable peripherals on the microcontroller. This extends their functionality to include: Communication (UART/I2C/SPI), Advanced I/O (Capture/Compare), and External Interrupts.

3.2.1.7 Manufacturing

Each part of the circuit was assembled and tested by hand. Once this process was finished and the board design was finalized, it was sent to a manufacturer for fabrication and assembly. The final board was a four-layer PCB (thickness 0.6mm) with dimensions of 23mm x 33mm and 52 surface mount components.

3.2.1.8 Finalizing Hardware

The NUIMU is powered by a 265mAh battery. The 24-pin external connector can be fitted either vertically, or at a 90-degree angle. A case was designed for each version and printed using a Projet 3500 HDMax 3D printer.

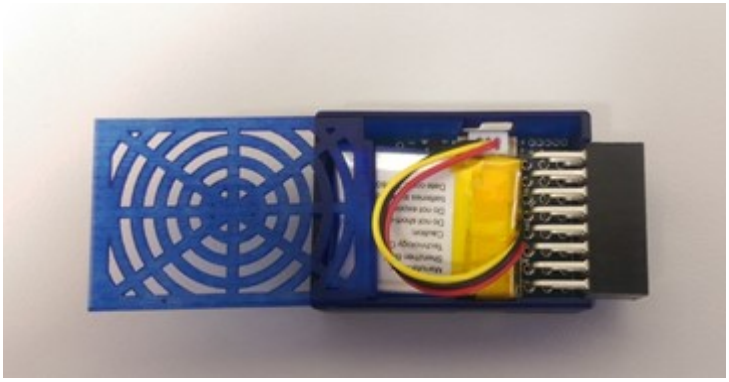

Figure 4 – Final Package of NUIMU hardware

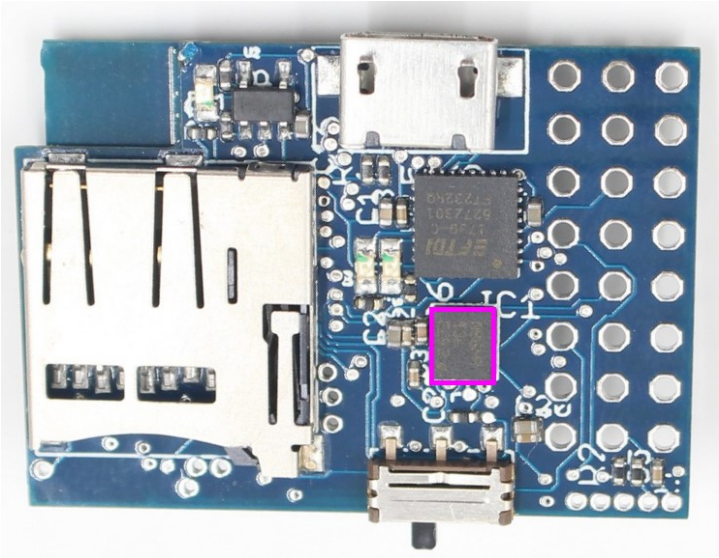

Figure 5 – Updated IMU element on NUIMU board

3.2.2 Hardware Improvements

3.2.2.1 Improved Inertial Measurement Unit

It became necessary over the course of the development to upgrade the IMU, as STMicroelectronics stopped producing the LSM9DS0. The LSM9DS1 was selected as a replacement since it was functionally similar. The new IMU also has a smaller footprint that its predecessor, which required a modification to the board. The internal layout of the new sensor is different, but by updating the firmware of the microcontroller, the IMU could be configured to perform the same way.

3.3 Firmware

The microcontroller firmware responds to commands given by the host device. They also have a page of the Program Space in the Flash Memory set aside for a custom device registry. Saving to this non-volatile memory means that the device can be configured for a particular use case without having to reprogram the board. The data commands are sent in 6-byte packets that have one of two formats, either:

$0x07$	b_1	b_2	b_3	b_4	$0x0B$

where b_{1234} is a four-character command, or

$0x07$	$'C'$	$address$	v_1	v_2	$0x0B$

where $address$ denotes a registry location, and v_{12} is the value to be written to that location.

The registry of the NUIMUs contains the following information, which can be set through the second data format:

- Sample rate for the analogue channels (referred to as base sample rate)
- Sample rate for the IMUs (as a divisor of the base rate)
- Which ADC channels to sample
- Internal sample rate for the LSM9DS0/1 (One byte for each sensor)
- Range for each sensor LSM9DS0/1 (One byte for each sensor)
- Offset values for the Magnetometer (Two bytes per axis)
- IMU Firmware ID (read only)

Four-character commands to the NUIMU can either return a value (such as battery level) or initiate new behaviour (such as starting a data stream). Figure 6 shows the basic functionality of the firmware.

The NUIMU sends packets in the following format:

$0xDD$	$0xAA$	$0x55$	$Length$	$Type$	d_1	…	d_N	$Clock$	$Length$

where $Length$ denotes the number of packets in the data and $Type$ tells the host what kind of data it is receiving. The $Length$ is sent at both the beginning and the end of the packet to allow the host to detect if any bytes have been lost during transmission.

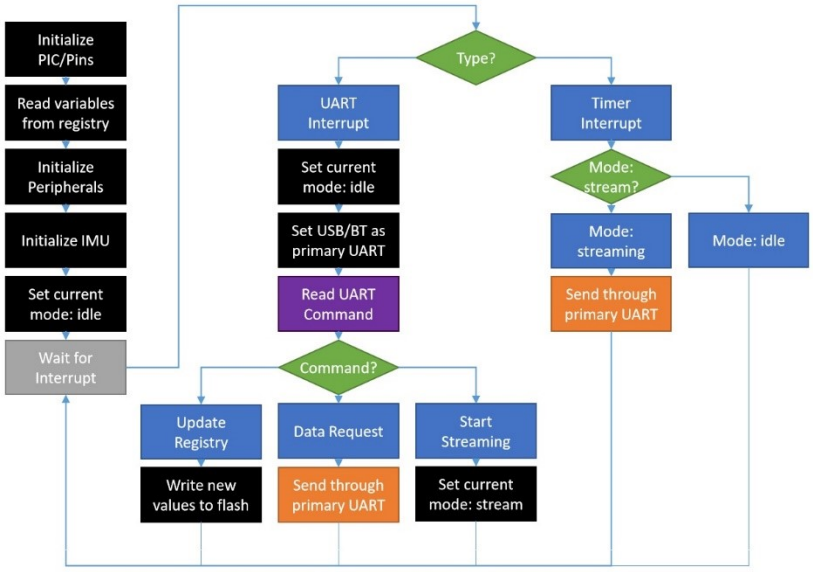

Figure 6 – Flowchart demonstrating program flow in NUIMU

3.4 NUIMU Attributes

A description of the characteristics of the NUIMU can be found in Table 3.

Table 3 – Description of NUIMU hardware attributes

Inertial Sensors	3-Axis Accelerometer, 3-Axis Gyroscope, 3-Axis Magnetometer
Inertial Sampling Rate	Up to 1kHz
No. of Auxiliary Ports	8
Auxiliary Port Sampling Rate	Up to 1kHz (analogue mode)
Auxiliary Port: Functions	Analogue I/O, Digital I/O, SPI, I2C, UART, PWM, Interrupts
PIC clock speed	30MHz (Primary), 32.678kHz (Secondary)
Communication Speed	5MHz (SPI), 400kHz (I2C), 115200bd (UART USB), 4Mbd(UART BT)
Battery	265mAh Lithium Polymer
Battery Life	2.5 hours
Charging Time	1.5 hours

3.4.1 Other Applications

Re-configuring the NUIMU Auxiliary port allows for a large range of additional applications. This has proved useful at a number of different points, both during this project and in other workstreams. The different uses are documented here.

3.4.1.1 Secondary Inertial Measurement Unit

Some human monitoring applications of this hardware required up to four IMUs to be connected and streaming simultaneously. The Bluetooth modules were presenting themselves as SPP devices, and when connected, both transmit and receive channels register as separate devices. The Bluetooth protocol limits the number of devices that can be actively connected in a network (known as a piconet) to eight. The host device acts as the piconet master, resulting in a maximum of seven slave devices. As each NUIMU takes up two slots, this resulted in the fourth IMU not connecting to the network in many cases (More advanced Bluetooth hosts were able to run more than one piconet).

The solution to this was to increase the number of IMUs for each Bluetooth module. A number of breakout boards of the LSM9DS1 were purchased and connected to the NUIMU via a cable. The NUIMU could then sample both the on board IMU and the external IMU and send both in the same packet. The external IMUs used I2C as a communications protocol, so two channels of the auxiliary port were re-purposed to facilitate this. The two IMUs can be seen in Figure 7.

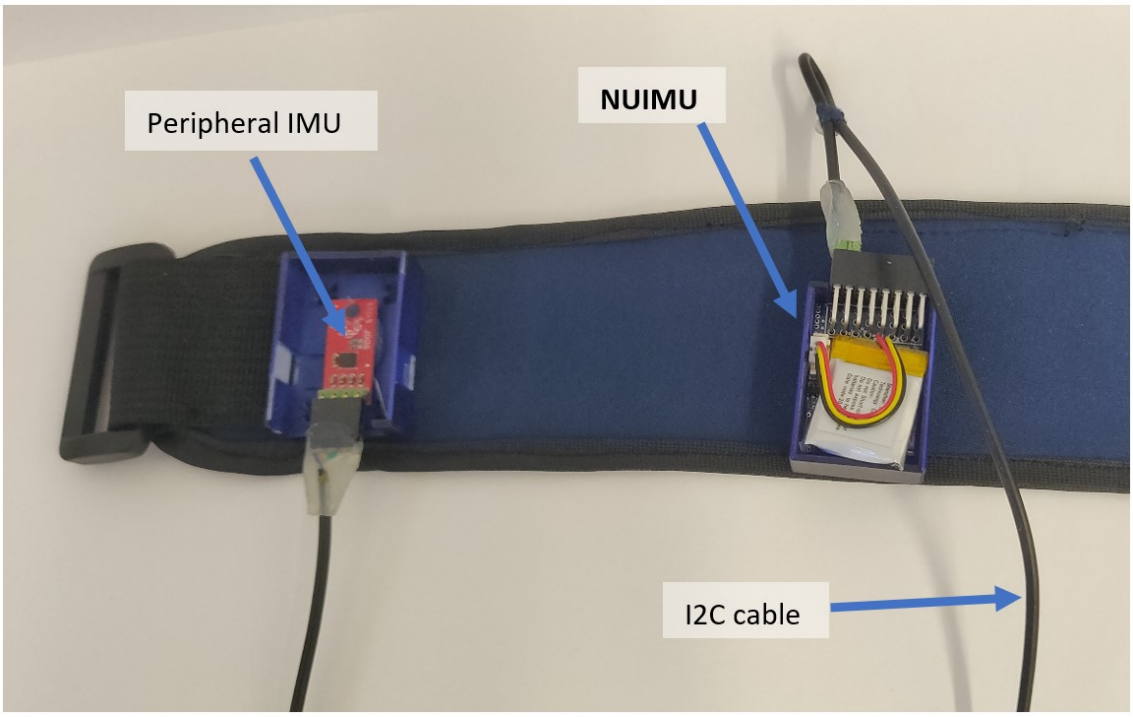

Figure 7 - NUIMU with peripheral IMU

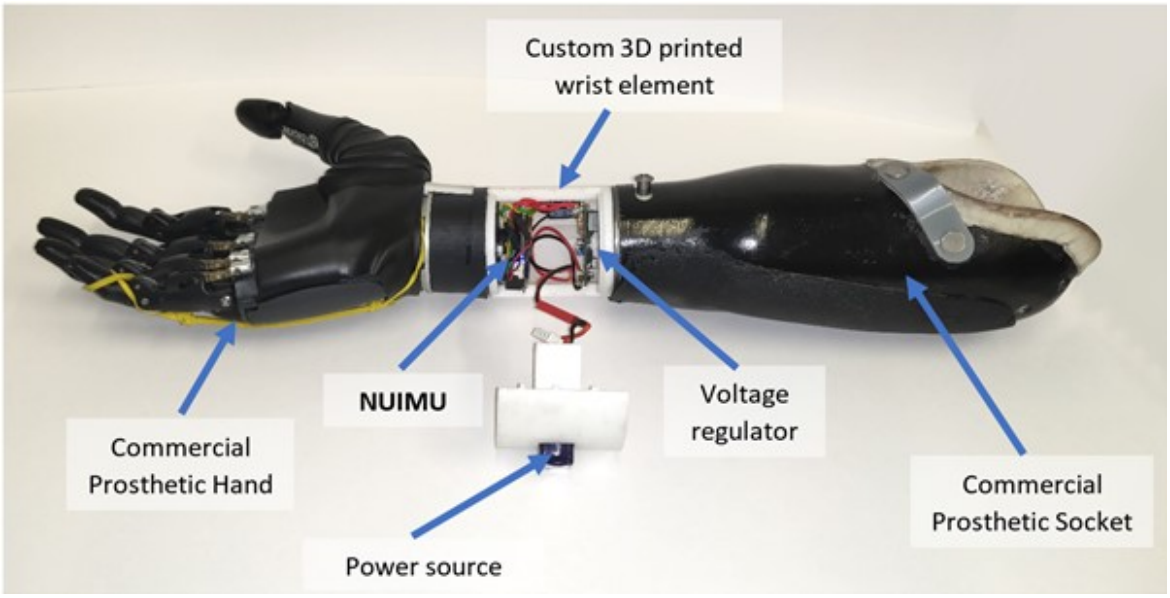

Figure 8 - NUIMU used as a prosthetic hand driver

3.4.1.2 Prosthetic Hand Driver

As part of this investigation, a prosthetic hand was used as a test platform. The prosthetic hand is powered by an 11.1V battery inside a prosthetic wrist, this is regulated down to 8.3V by an external voltage regulator. The hand is controlled by two signal channels, which operate at 3.3V, which is the same as the NUIMU. It was desirable to make the prosthetic hand wireless, as along with the battery this would make it portable, and so a NUIMU was used to add this functionality to the prosthetic.

In order to do this, two of the auxiliary ports were configured to be driven by the microprocessor on the NUIMU and connected to the inputs of the prosthetic hand. This allowed the system to trigger the hand to either open, close, or to switch to a different gesture. The complete wrist unit is presented in Figure 8.

3.4.1.3 Infrared Remote

Another potential application of this system is to allow for the direct control of smart devices. It is common for smart devices in the home to be controlled using an Infrared (IR) remote, therefore mimicking those signals was thought to be an appropriate way of controlling the devices without having to pre-code specific IR commands. An NUIMU was adapted with custom firmware to do this.

The IR NUIMU has an IR receiver and an IR LED plugged into the auxiliary port, and ten memory locations for data storage. There are ten characters that represent a command to record to the ten locations, and ten that represent a command to transmit the stored IR patterns. Once a command to record an IR pattern is detected, the NUIMU monitors the IR receiver data line to observe when an IR signal is first received. At this point, it starts a fast timer and records the time between the signal switching, resetting each time.

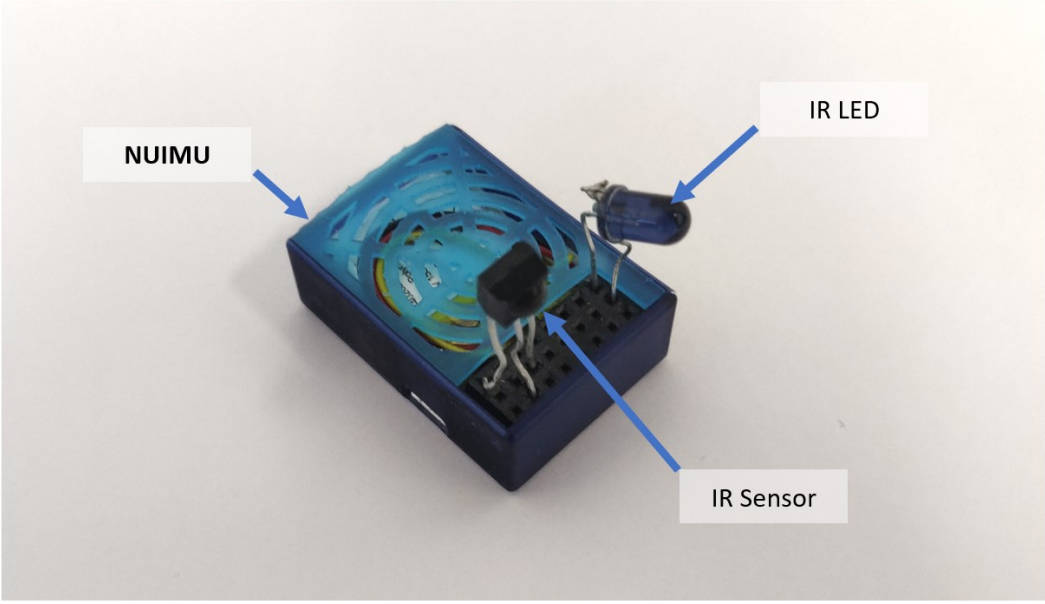

Figure 9 – NUIMU as an infrared repeater

Once the timer overflows, it is determined that the signal has ended, and an array containing the length of each signal segment is recorded. To transmit the signal, the LED is set on and off for the lengths of time that were recorded in the timer. This has been shown to work for a variety of household devices, ranging from SkyTV boxes to remote control candles.

3.5 Mechanomyography Sensor Design

The design of the MMG sensors used in this research is based on the work of R. Woodward, and of Posatskiy et al. The sensor houses a Knowles SPU1410LR5H MEMs microphones, seated at the apex of a rigid conical chamber (height x ⌀ - 5mm x 7mm). This shape and dimensions were found by Posatskiy to provide the highest gain of the MMG signal while maintaining the flattest frequency response. The microphone is sealed, and the mouth of the chamber is covered by a 4-micrometre thick Mylar membrane. The signal recorded from the microphone represents the distortion of the membrane. When the membrane is placed on the skin above a contracting muscle, the vibrations caused during that contraction propagate through the skin and through the Mylar, inducing a change in pressure within the chamber.

Each MMG sensor consists of either two or three 3D printed parts. The MMG housing holds the microphone PCB and provides the conical chamber. The PCB is placed above the chamber, and silicon is used to seal the top. This is intended to ensure that air cannot escape from the chamber, ensuring that when no signal is present the sensor output is predictable. The ring is used to hold the membrane in place, sealing the other end of the chamber. Finally, the clip is used to hold the sensor in place. All three parts were manufactured using a ProJet 3500 HD Max 3D Printer (layer resolution - 0.016mm, typical accuracy - 0.025-0.05mm). The material was Visijet M3 Crystal, which has USP Class VI certification, and therefore is safe for use when manufacturing devices that will be in contact with the skin for extended periods. The MMG housing was redesigned to make assemble and maintenance easier. In this new version, the ring was moved to allow membrane replacement without requiring disassembly.

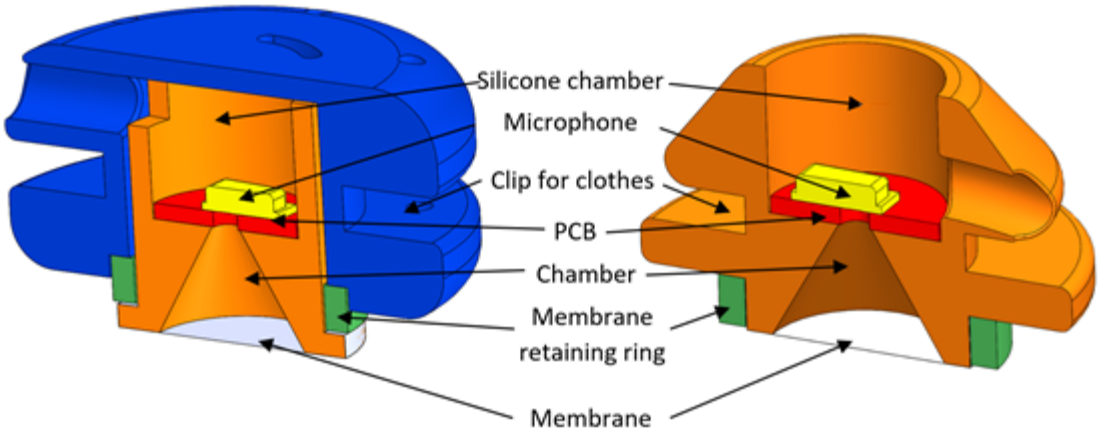

Figure 10 - MMG designs

Both MMG housings, as well as the PCB can be seen in Figure 10. The MMG PCB was redesigned with pads instead of through-hole connectors for the wires. This ensured that the bottom of the PCB could be kept entirely flat, giving a better seal to the board.

The three wires from the MMG PCB plug into the power and analogue channels of the NUIMU, and a maximum of eight of these MMGs can be combined with inertial data to form one module of the sensor suite.

3.6 Sensor Suite Evaluation

The sensor suite is evaluated by comparing against the key user requirements:

- **Non-Restrictive** – the sensors are body mounted, self-contained, and do not inhibit the user. The only requirement is that they must be in physical contact with the skin, either directly or through thin clothing.
- **Unobtrusive** – the sensors are smaller than many other solutions (h x w x l) and lightweight. They are attached through a comfortable neoprene band.
- **Easy to use** – the sensors can be placed on anybody and require no user specific calibration to produce data.
- **Consistent** – the response from the sensors is consistent, it is not affected by transient environmental factors such as skin conductivity.
- **Low-Power** – the sensors are low power, with a 265mAh battery lasting approximately 2.5hours.
- **Wireless** – the device can both communicate and stream data in real time over Bluetooth.
- **Internal Storage** – the microcontroller can write to its own memory to store registry values. It can also write to an SD card for data logging.
- **Configurable peripherals** – the microcontroller can communicate with peripheral sensors as it is capable of both analogue to digital conversion and digital communication through its auxiliary ports.

Based on these criteria, the NUIMU is suitable for these experiments.

3.7 NUIMU Host Software

To perform real time analysis, the NUIMUs require a host device. Two have been used over the course of these experiments, a Windows Personal Computer and an Android Phone. These host devices are able to connect to multiple NUIMUs, allowing a greater range of information to be considered.

3.7.1 C# code for Windows (The NU Interface)

The NU Interface is a GUI that operates primarily through a text box. It is designed to run the IMUs for every application for which they can be used. The program is split into two major classes: the User Interface Class (UIClass), and the NUClass. There are several other minor classes that have specific functionality within these two. The functionality of each of these major classes is described here, starting with the child class.

3.7.1.1 NUClass

The NUClass is a manager class for each individual NUIMU and is created as a child of the UIClass whenever the user attempts to connect to an IMU. The NUClass is generalised and has been used in a number of other projects to control the NUIMUs.

Connection Manager

When the NUClass is created, it attempts to connect to the specified NUIMU, first through Bluetooth, and then through USB if this is not successful. When a connection is successfully made, a thread is started which attempts to extract complete packets from the data stream and fires an interrupt whenever one is received. The class also sends commands to the device.

If a Bluetooth connection is successfully made, this class records the Bluetooth address. In future attempts to connect to the NUIMU, the code will first attempt to connect directly to this address without having to go through the Bluetooth manager, and only returns to the manager if this is unsuccessful. This reduces the time required to establish a connection from ~30s to ~5s. It also has the added benefit that the device does not need to be paired prior to the connection attempt if the address has previously been saved, as this will be prompted.

Quaternion Generation

The raw inertial data from the NUIMU is used to generate world reference quaternions as a measure of orientation. The code includes variations of Madgwick's algorithm, which are described in detail in Chapter 4. The sample period is calculated based on the registry values of the NUIMU and updated to account for lost packets using a clock byte in the packet. The NUClass also has the capability to run more than one quaternion generation algorithm simultaneously for the same data, allowing a comparison between them to be performed.

MMG Signal Processing

The specific methods of MMG signal processing are described in Chapter 5. They use a thread pool based in this class to ensure that they can operate at the maximum frequency of the device. The variations of this class either detect whether any signal that could represent a gesture has occurred, or which specific gesture has been performed. All the processing for the classification is performed in a child class to NUClass.

Data Visualization

The data visualization was performed using a number of sub classes. Visualization of the orientation was performed by rendering a cuboid on the screen, initially using the DirectX API to generate the appropriate 3d rendering, and later using SlimDX so that the code was not reliant on that API. Vectors in the form of 3D arrows were also added to help visualize the algorithms estimation for gross acceleration and for the gross magnetic field. An example of the data visualisation interface can be seen in Figure 11.

Updating the Registry/Calibration

The NUClass contains a small dedicated GUI for changing the registry values of the NUIMU. The GUI has three pages of settings.

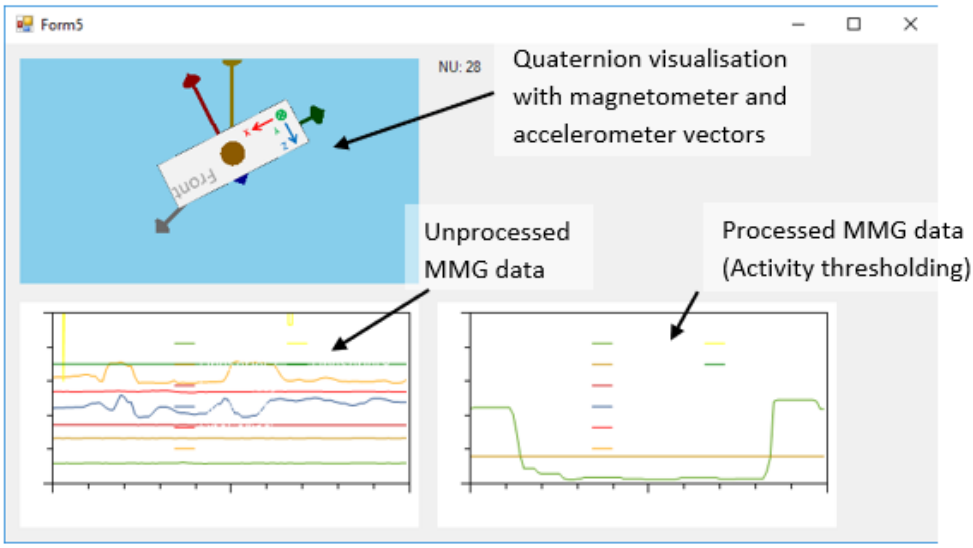

Figure 11 - Real-time data visualisation

The first page allows you to configure the function of the IMU. It allows you to pick a desired base frequency of the IMUs and converts this into the two settings needed to configure the timer on the PIC24. It then offers you factors of this frequency to allow you to slow the rate of inertial data while maintaining a high rate of MMG data. It is the inverse of this process that is used to calculate the sample period for the quaternion generation. This page also allows you to turn off ADC channels, which shorten the packet and allow for a sample rate above the theoretic maximum.

The second page of the GUI is used to change the set-up values for the on-board IMU, allowing you to change both the update rate and data range of each of the three sensors. Using the firmware version number, this page will give different options based on whether the on-board IMU is an LSM9DS0 or an LSM9DS1.

The final page of this GUI allows you to store the calibration values of the magnetometer. For later firmware versions, this is stored in the NUIMU flash memory, and a backup of this information is made each time the IMU is first connected. Earlier versions store this information in the backup file.

Faux NUClass

The NUClass has the capability to create and manage child versions of itself for specific applications. The most common reason to do this is to allow more than one IMU to be connected through the same Bluetooth channel. A Faux NUClass will not have any active connections, instead the appropriate data received in the parent NUClass will be passed to it to generate orientation.

Other than the connection manager, the rest of the functionality described in this section remains the same. The only deviation from this is that a data file is used in place of the registry. This allows to IMU calibration values to be saved onto the computer, instead of on the IMU.

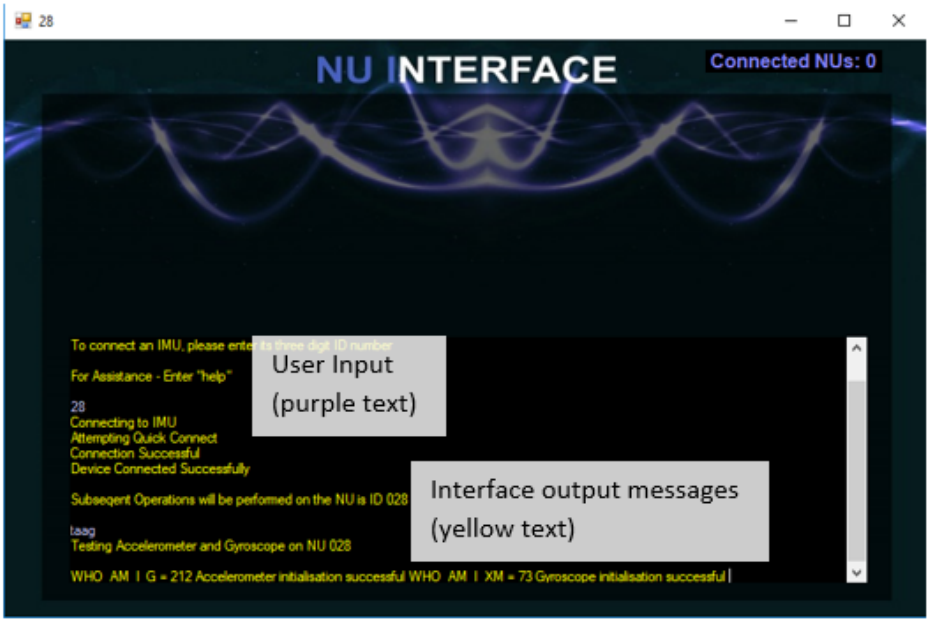

Figure 12 - Main interface window

3.7.1.2 UIClass

The User Interface class (UIClass) is the entry point to the code, and the ultimate parent class of every other class used in the application. It is designed both to perform the experiments described in this thesis, and as a demonstration for how to use the NUClass in other projects.

User Interface

The primary form of interface for this code is through text commands in this interface, although some of the more advanced functions create smaller GUI interfaces for specific tasks. A text-based interface was chosen as the use of a scrolling text window ensures that a record of each interaction is available to the user, including any messages that have been sent from NUIMU. Every line written to this textbox is timestamped and written to a LOG file. The basic UI interface is shown in Figure 12.

NUClass Manager

The UIClass creates new instances of NUClass for each NUIMU connected. When the user enters a command, UIClass is responsible for triggering the correct processes in the correct NUIMU. Each instance of NUClass is stored in a List within the class, with a record maintained of which IMU the user last interacted with. This record can be updated by the user if they wish to send a command to a new IMU.

Recording Data

It is frequently useful to store data recorded from the NUIMUs during different applications. This is possible in the UIClass through an interrupt that can be generated by NUClass whenever new data presents itself. The data is stored in a time stamped file that can be named dynamically. The reason this

functionality was added to the UIClass was to allow more than one IMU to be recorded simultaneously in the same file. This ensures that the data is temporally aligned. A small executable file is included to split this file into many, one file per NUIMU if that is preferable.

Angle Generation

There are several applications for which only knowing orientation is not enough. This class can make comparisons between two quaternions generated by two IMUs to determine the angle between them. This is useful when examining body kinematics.

Piping Data

For many of the more complex visualizations, other programs were used. Using the pipes class enabled data to be sent to programs such as Unity, which could provide more advanced simulations. In these cases, quaternions were generated for each NUIMU, and then repackaged and sent to the target program using a pipe. The new packets used a similar packet structure, consisting of a three-byte header, a packet identifier and an end byte.

TCP/IP

Several of the experiments required data to be sent from one computer to another, most often from a Windows environment to a Linux environment. This required the use of a TCP/IP class that is managed by the UIClass as a server. It also required a C++ client to be written to receive the data on the target machine and convert from the transmitted bytes back to the original data format.

Simulation

In order to visualize recorded data, a simulation environment was created. This environment loaded data that had been previously saved, loaded a file of calibration values, and created a Faux NUClass. It then used this faux class to generate quaternions and displayed them to the screen. It gives the user the option to have as many sub-windows open as they choose, meaning that they can process and simulate many raw files simultaneously, or the same file with different calibration files. It also saves a copy of the generated quaternions for future use.

Other Applications

This UIClass is the basis of any function which require data from more than one NUIMU, or which give real time feedback to the user. It operates all outputs that are displayed in Chapter 6. It also can create wrappers for android phones running a compatible app, or third party IMUs, allowing them to be controlled through NUClass.

3.7.2 Android code for Mobiles (The NU App)

The NU App is an android app designed to interact with the NUIMUs. It is far more basic than the NU Interface and has several application specific forms. It currently can only communicate with a single IMU and is intended more primarily as an experimental platform to enable data collection in difficult environments. The app has two major iterations:

3.7.2.1 First Iteration (Android Studio)

The first iteration of the app was written purely in Android Studio, an Integrated Development Environment (IDE) that uses a variation of Java. The app was able to establish a Bluetooth connection to the IMU and send it the command to start streaming data. It required a lower data rate than the NU Interface, as it was unable to process the data as fast. The Madgwick algorithm was implemented and allowed for the real time generation of quaternions. Along with the raw data, these quaternions were then saved to a file to be used at a later date. This was beneficial because it removed the requirement to stay within Bluetooth range of the host computer, allowing many more activities to be monitored.

This app has two additional functions. The first is as a Bluetooth Low Energy (BLE) to Bluetooth converter. This allows other BLE IMUs to be evaluated alongside the NUIMUs by connecting them to the NU Interface via the phone. The second was as an input for the NU Interface directly. Smart phones have an IMU built into them that could be used as an input.

3.7.2.2 Second Iteration (Unity)

The second iteration of this app was created in Unity. Unity provided the capability to easily create 3D environments in an android environment. This implementation required three component parts. The 3D models were imported into Unity's IDE, and positioned with cameras, lighting and materials that made them visible. A C# script was written to update the position of the models based on the calculated quaternions. The quaternions were calculated in an android library written in Java in Android Studio. The Android Studio library was able to access the hardware functions of the phone, allowing it to make and monitor Bluetooth connections. It therefore also acted as a wrapper for the hardware reliant functions. The combination of these three allowed the Unity app to connect to the NUIMU and provide a real-time representation of the current orientation on the phone.

3.8 Chapter Summary

This chapter reports on the development of an experimental hardware/software platform for the pervasive monitoring of motion and muscle activity. Motion is observed using an inertial measurement unit with a MARG sensor housed in the main case. Additionally, up to eight muscle sensors can by plugged into this primary board, which can wirelessly transmit data from both the muscle sensors and the inertial sensors at rates up to 1kHz. The NUIMU can be wirelessly configured and has in-built long-term storage capabilities. The IMU was created through an iterative process, and the final version was assessed against the Key User Requirements and found to be fully compliant.

The NUIMU can act as the primary controller in a reconfigurable sensor array. Other sensing elements can be included in the array through the auxiliary port, which can be used to connect up to eight powered MMGs, other analogue sensors, or digital sensors through inter-IC protocols such as SPI of I2C. This allows the real-time fusion of multiple sensors to be easily implemented, either on-board or using the temporally aligned data.

A number of custom programs were created to communicate with this hardware from other host devices. Among these was a Windows form-based environment that included both hardware and application specific code. This code was broadly split into two component parts, a class that managed each individual IMU and a form through which the user could manage many IMUs simultaneously. This allowed the code and the IMUs to be used for a wide variety of applications.

In addition to this, two lightweight phone applications were created. These were used for data gathering when a computer was not appropriate. The hardware was designed to be reconfigurable depending on the required application. Many other applications were also explored, including networked IMUs and interfacing with third party hardware, such as a prosthetic hand or smart TV.

Chapter 4

Detailing Improvements to Orientation Estimation

4.1 Initial Implementation of Orientation Estimation Algorithm

Of the algorithms described in Section 2.4, the gradient descent optimization algorithm (Madgwick algorithm) was chosen for implementation because it provided a good accuracy for the computational resources it required. The inertial measurement units were validated by comparing their output at known orientations. For this comparison, the algorithm was implemented in the host software and once validated the algorithm was moved from the host software to the embedded firmware. This resulted in an increased power efficiency by reducing the quantity of data that needed to be transferred over the Bluetooth connection. During this process it was noted that there were several changes that could be implemented in order to improve the rate of convergence, the computational efficiency and the accuracy of the algorithm. This resulted in several iterations of the orientation estimation algorithm, which are described in this section. In order to ensure that all algorithms received the same input data, a virtual IMU was created that provided consistent and repeatable sensor inputs from known orientations. This allows the variations of the algorithms to be directly compared, both to each other and to a system truth.

4.1.1 Sensor Bias Removal

Before the algorithms can be used, sensor bias needs to be removed. It is desirable to do this when using the gyroscopes, as sensor bias can lead to an unpredictable speed of convergence, or 'dragging' away from the corrected value, but it is a requirement for magnetometer use, since the orientation will aim to ensure that the measured vector for the geomagnetic field is aligned to the systems magnetic north.

4.1.2 Gyroscopic Calibration

The gyroscopic calibration is performed each time the orientation tracking algorithm is initialized. To function correctly, the IMU must be stationary during this initialization period, which lasts for 200 milliseconds. The IMU gathers 200 samples of gyroscopic data and computes the average value. This value is then written to the IMUs internal registry and removed from subsequent gyroscopic data before it is used to estimate rotation.

$$GyroBias_{(x,y,z)} = \frac{\sum_{i=1}^{200} GyroValue_{(x,y,z)}}{200} \tag{21}$$

4.1.3 Magnetometer Calibration

Magnetometer calibration was vital, since a poorly calibrated magnetometer could lead to an error in rotation equivalent to up to 180˚ in the yaw direction. Since pitch and roll are not decoupled from yaw, a fast rotation around the Z-axis that will be 'observed' by the gyroscopes will be 'corrected' through the gradient descent, which may lead large deviations around all three axes. It must be noted that magnetic north when measured in three dimensions has an element that is parallel to the earth's surface, and an element that is perpendicular to it. The angle between the measured magnetic field and the earth's surface is known as magnetic inclination.

There are four types of magnetometer interference that could lead to unpredictable results:

- **Hard iron offsets** – Hard iron offsets are the result of a source of magnetic field on the sensor/PCB. They are often present as a result of the ferromagnetic components on the board and can change both over time and as the result of exposure to other magnetic fields. Since they are generated by components fixed to the PCB, they are rotated as the sensor is rotated, and therefore have a constant additive effect on sensor output.
- **Soft iron distortion** – Soft iron distortion is caused by materials that are not themselves magnetic, but which can interfere with the geomagnetic field. The effect of this distortion is that magnetic field can appear to change amplitude and direction in according to sensor orientation. This is normally compensated for by assuming that the scaling effect is linear and modelling the correction.
- **Tilt** – Tilt is when the axes of the sensor do not align with the global axis when no rotation has occurred. It is usually due to the sensor mounting on the PCB but can be modelled and corrected if necessary.
- **Environmental interference** – Environmental interference is another form of additive interference. It differs from Hard Iron Offsets because its effects are consistent in the global reference frame instead of the IMUs local reference frame. An operator must be aware of it if a) it is inconsistent over time and b) they require a global reference frame that is aligned with true magnetic north.

Magnetometer calibration was performed to remove the Hard Iron Offsets in two ways; both of which required the operator to perform known movements.

During the first method of magnetometer calibration, the IMU was placed on the side designated the top, and a button was pressed. The IMU recorded fifty samples of data from all three axis of the

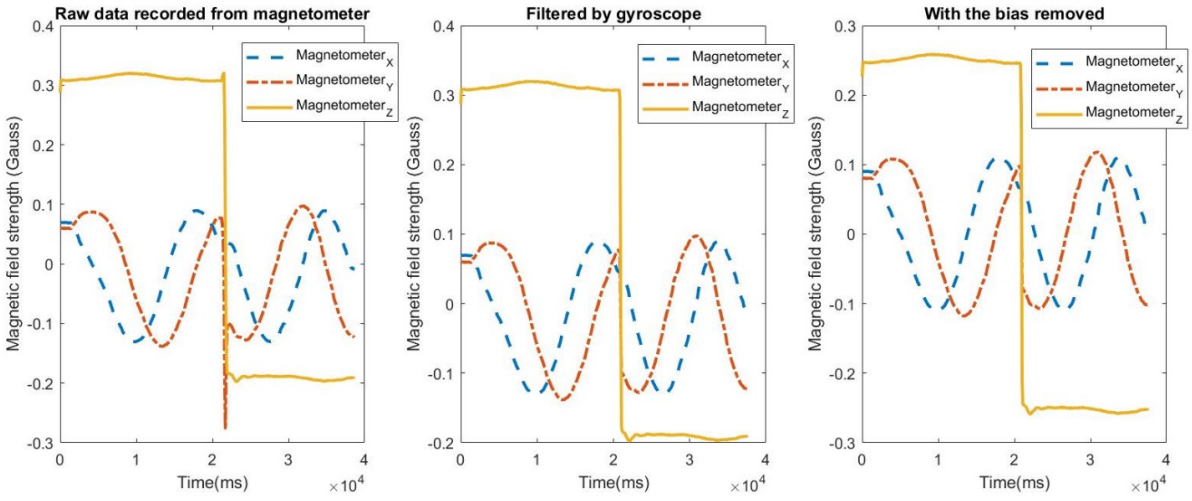

Figure 13 -Magnetometer output (Left: Raw magnetometer output. Mid: Cropped and Filtered. Right: Calibrated output)

magnetometer. The operator was then required to rotate the IMU ten degrees around the vertical axis and resample. This was repeated until all 36 positions had been sampled, then the IMU was placed on the side designated as the bottom, and the process was repeated. The magnetometer bias for each axis could then be found simple by taking the average of all samples taken along that axis. The intent behind this method was a) that many samples would negate the effect of any high frequency noise and b) that by sampling every ten degrees would ensure that the maximum range was found for each axis. The additional benefit of this method was that it allows the sensors to be tested for the effects of soft iron distortion and tilt, both of which were found to be negligible for this board layout and sensor. This method provided accurate measurement of sensor bias but was time consuming.

The second method of sensor calibration was to replace the spin and stop technique with a constant rotation of 360˚, turning the device over and completing a second rotation to finish the data collection. Instead of averaging, this method finds the maximum and minimum of the data and takes the midpoint as the bias. This calibration algorithm detects the increased gyroscopic activity that indicates that the user is turning the device over, and removes the magnetic information for this period, ensuring that it is not included in the bias calculation, which is as follows:

$$MagnetometerBias_{(x,y,z)} = \frac{\max\left(MagData_{(x,y,z)}\right) - \min\left(MagData_{(x,y,z)}\right)}{2} + \min\left(MagData_{(x,y,z)}\right)$$

The implementation of this algorithm can be seen in Figure 13.

This has the benefit of being significantly less time consuming, but it is also more likely that the operator will tilt the device as they rotate it. Since magnetometer calibration can be changed when the IMU is passed through strong magnetic fields, the calibration process had to be performed regularly, and so this was the preferred method.

4.1.4 Comparison Against Known Orientation

Initially, the orientation estimation algorithm was implemented in the C# host application, and raw sensor data was transmitted from the IMU directly to this application over Bluetooth.

To compare this implementation against a known orientation, a second form of orientation monitoring had to be implemented. A 3D motion tracking system (Optotrak, NDI Ontario, Canada) was used to provide this tracking, due it its high accuracy. The tracking system uses a beacon comprised of 3 markers, which allow the position and orientation of the beacon to be observed. An image of the beacon can be seen in Figure 14.

Figure 14 - Optotrak system used to evaluate orientation estimation algorithm

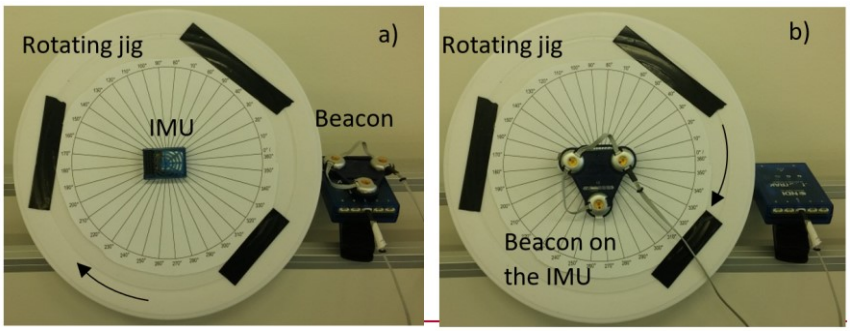

Figure 15 - Experimental set up for orientation estimation evaluation (a – IMU in position. b- IMU with beacon affixed)

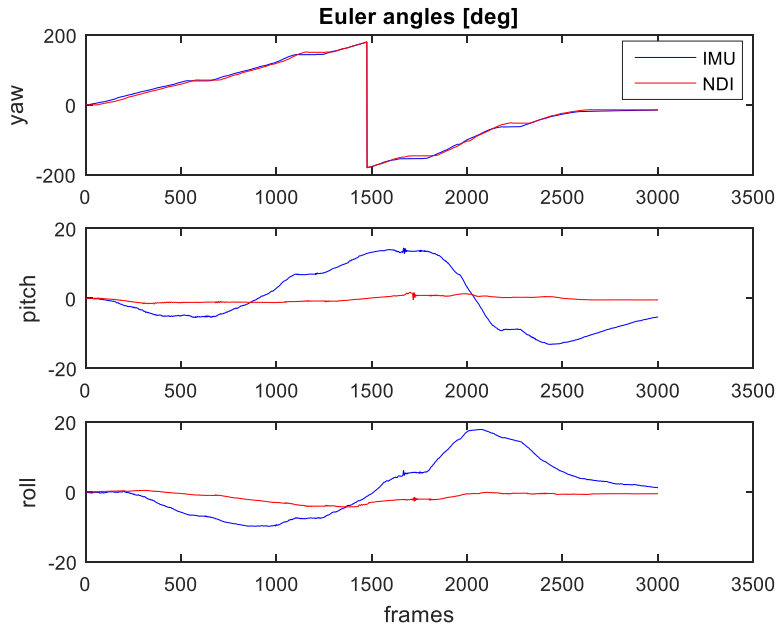

Figure 16 – IMU orientation output against measured orientation

The tracking system is able to locate the beacon to a 3D accuracy of 0.1mm, and so its accuracy was adequate for evaluating the accuracy of the gradient descent algorithm. The beacon was rigidly fixed to the IMU, and the IMU was recalibrated to compensate for any new hard iron offsets introduced by the additional hardware. The IMU was then attached to a vertically mounted rotating platform, which was manually rotated through a full rotation, as shown in Figure 15.

Each rotation lasted approximately 30 seconds. The quaternions generated by both the IMU and the tracking system were recorded during the rotation. After rotation the quaternions were converted into Euler angles in the IMUs local reference frame for ease of comparison, an example rotation can be seen in the Figure 16.

The errors are described in Table 4:

Table 4 – IMU errors

Mean error (degree)	**Maximum error (degree)**	**RMSE (degree)**
5.69	10.30	6.12

A mean error of 5.7 degrees was observed during these rotations. This is comparable to the accuracy offered by other commercial low cost IMUs and it was concluded that this accuracy was sufficient to move forward tracking human motion.

4.2 Improving the Efficiency of the Algorithm

The algorithm was implemented in the firmware of the IMUs. This step reduced the volume of data transmitted over Bluetooth, therefore reducing the power consumption of the device and leading to longer functional battery life. The high sample rate defined as a requirement in Section 3.2.3 increases the accuracy of the finite sums approximation for angular rotation that forms the gyroscopic element of the orientation estimation algorithm. It was noted that when the algorithm was embedded into the firmware, the maximum sample rate dropped to approximately 300Hz.

The maximum sample rate is determined by the processor speed, and within the context of the IMU, it is fixed based on the number of clock cycles required. In order to increase the sample rate of the algorithm to that defined in the specification, the number of clock cycles required has to be decreased by approximately two thirds. This required the number of operations in the algorithm to be decreased.

The algorithm requires normalized vectors and quaternions, and as a result uses data types capable of represented decimal values, specifically floating-point numbers. In the PIC24F family, floating point numbers are represented using 32-bits, and their operations are significantly slower than their integer equivalents. To remove the floating-point numbers, they were instead replaced with 16-bit integers. As a result, values <1 were disregarded.

To ensure that enough of the relevant information remained, one integer unit ($1_{intUnit}$) was defined by:

$$1_{intUnit} = \frac{1}{scalingFactor} \; where \; scalingFactor \; (SF) = 2^{14}$$

The scaling factor was chosen as the highest number that could perform normal operations to normalized values without overflowing.

This required modifications to several mathematical operators, for example:

$$a * b = c$$

was replaced by:

$$\frac{a * b}{SF} = c$$

since:

$$\frac{a}{SF} * \frac{b}{SF} = \frac{c}{SF}$$

This resulted in the potential generation of 32-bit values during the process of multiplication, so monitoring data typing was required to ensure no overflows occurred.

In addition to this multiplication, simple multiplications and divisions were replaced directly with binary operations, for example:

$$a * 2 = a \ll 2$$

Normalization of vectors and quaternions normally requires a division by the norm of each axis.

The norm is computed by taking the root of the squares of the axis:

$$norm\left(v_{((w),x,y,z)}\right) = \sqrt{(v_w^2 +)v_x^2 + v_y^2 + v_z^2}$$

The square root is a computationally heavy function, and as a result it was replaced with an approximate, which calculates the root by guessing each bit (most significant to least significant) and comparing the squared guess to the number that is to be rooted. This allows the square root to be found in a number of steps equivalent to the number of bits in the rooted number data type. Additional speed is possible in exchange for ignoring a number of the least significant bits. This introduces an error at each step, but these errors are corrected by the gradient descent during future steps. Once the norm value has been calculated, each element in the vector can be divided by it to create a normalized vector.

Division operations are computationally expensive in the PIC24F family, requiring approximately 8 times longer than multiplication, and as a result it was desirable to reduce the number of times a division occurred.

A normalized axis (m'_x) can be calculated using the original value (m_x) and the vector norm ($norm$) through:

$$\frac{m_x}{norm} = m'_x$$

Since the values are scaled, the scale factor must be included in this calculation:

$$\frac{m_x/SF}{norm/SF} = m'_x/SF$$

A 32-bit value was chosen to abstract the division from each vector. The value 2^{29} was chosen, as this made the order of the calculation easier to avoid overflow errors. Multiplying the fraction by 1 expresses as $2^{29}/2^{29}$ gives.

$$\frac{\frac{m_x}{SF}/2^{29}}{\frac{norm}{SF}/2^{29}} = \frac{m'_x}{SF}$$

Which simplifies to:

$$\frac{1}{2^{29}} * m_x * \frac{2^{29}}{norm} = \frac{m'_x}{SF}$$

Therefore:

$$\frac{1}{2^{29}} * SF * m_x * \frac{2^{29}}{norm} = \left(SF * m_x * \frac{2^{29}}{norm}\right) \gg 29 = m'_x$$

If:

$$\frac{2^{29}}{norm} = r$$

Then

$$\left(SF * m_{(w),x,y,z} * r\right) \gg 29 = m'_{(w),x,y,z}$$

As a result, the three divisions of a vector normalization are replaced by a single division to generate the value for r. This makes normalization approximately three times faster for vectors and four times faster for quaternions.

Making these modifications led to the required improvement in performance, the time taken to execute one step when from ~3ms to ~1ms, allowing the specified 1kHz sample rate to be maintained.

4.3 Gradient Descent Modifications

4.3.1 Motivation

The gradient descent orientation estimation algorithm consists of two parts, which are weighted and combined. The first is the gyroscopic integration, and the second is the correction step. In order for the algorithm to be robust in the many applications in which it can be used, both of these steps should be predictable for a given level of noise.

It has been noted in Chapter 2 that in original formulation of the gradient descent algorithm, yaw is not decoupled from pitch and roll. This is undesirable in many applications but can be particularly problematic in applications such as quadcopter control, where unexpected changes in pitch or roll could lead to unpredictable lateral movement. Additionally, it was noted that two subsequent steps down gradients that are not perpendicular could lead to an inconsistent step size in the first gradient, and a smaller than expected step in the second direction. This can lead to a slower than optimum convergence. Finally, the algorithm relies on an accurate value to inclination when calculating the magnetometer reference vector. If this is not available then the amount of time the algorithm takes to converge will increase significantly, as the two gradients will quickly begin to work in opposite directions. In situations where an accurate measure for inclination cannot be provided, the algorithm will rotate the measured vector into the correct plane, however this is inefficient since information is lost.

4.3.2 Solution

The solution to this is to remove the effect of magnetic inclination so the vertical component (parallel to the gravity vector) is removed. This is achieved through the generation of new vectors for the geomagnetic field that are perpendicular to the corresponding gravity vectors. These vectors can be calculated by taking the cross product of the acceleration vectors and magnetometer vectors:

The reference vectors (2.9) and (2.10) are used to create a new reference vector, referred to as the reference vector for magnetic east:

$$\boldsymbol{v}_{r(e)} = |[0,0,-1] \times [v_{rx}, 0, v_{rz}]| \tag{22}$$

$$\boldsymbol{v}_{r(e)} = [0,-1,0] \tag{23}$$

The measured vector for magnetic east is calculated using the measured vector for acceleration ($V_{m(a)}$) and magnetic field ($V_{m(m)}$) in the same way:

$$\boldsymbol{v}_{m(e)} = \left|\boldsymbol{v}_{m(a)} \times \boldsymbol{v}_{m(m)}\right| \tag{24}$$

Equation (3) and (4) can be substituted into (2.14) to generate the second of the two gradients to give:

$$\nabla_{\boldsymbol{q}} F(\boldsymbol{q}) = J_{\boldsymbol{q}}\left(\boldsymbol{q}^{-1} * \boldsymbol{v}_{r(e)} * \boldsymbol{q} - \boldsymbol{v}_{m(e)}\right)^{T} * \left(\boldsymbol{q}^{-1} * \boldsymbol{v}_{r(e)} * \boldsymbol{q} - \boldsymbol{v}_{m(e)}\right) \tag{25}$$

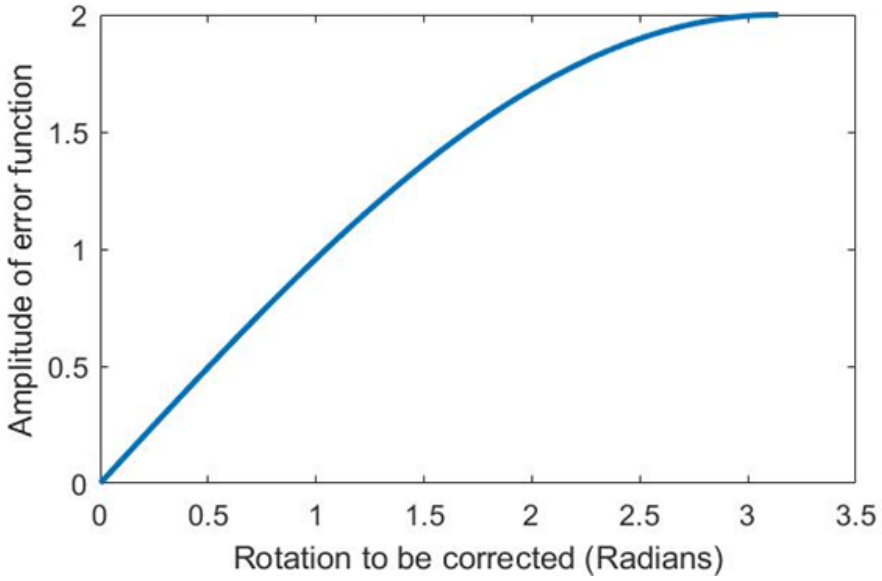

Figure 17 – Magnitude of error function with respect to rotation

4.3.3 Algorithm Convergence

Since the yaw is now decoupled from the pitch and roll, a new problem is presented when the predicted orientation is exactly π from the true orientation. During the gravitational correction step, the error vector is at its theoretical maximum, therefore the gradient at this point is 0 and no correction occurs. The magnetometer correction will rotate around the gravitational vector, but since this will no longer affect pitch and roll, the gravitational vector will remain at its theoretical maximum. This scenario is easy to test for within the algorithm, however in most cases it is unnecessary since gyroscopic information, IMU rotation and sensor noise will all displace the sensor from this maximum error state, at which point the gradient will move away from zero and the algorithm will converge. This can be demonstrated by demonstrating that the gradient can be calculated at every point in the relevant vector space. This is achieved by examining the error function, defined as:

$$\boldsymbol{v}_e = \boldsymbol{q}^{-1} * \boldsymbol{v}_r * \boldsymbol{q} - \boldsymbol{v}_m \tag{26}$$

This can be simplified by considering this function in a new reference frame (l), which in constructed so that $\boldsymbol{v}_m$ is parallel to ${}^{l}X_{axis}$ and the cross product of $\boldsymbol{v}_r$ and $\boldsymbol{v}_m$ is parallel to ${}^{l}Z_{axis}$. In this reference frame, the rotation denoted by $\boldsymbol{q}$ can be expressed as a rotation of θ around ${}^{l}Z_{axis}$. In this case, the two vectors $\boldsymbol{v}_r$ and $\boldsymbol{v}_m$ can be described as:

$${}^{l}\boldsymbol{v}_m = [1,0,0] \tag{27}$$

And:

$$^{l}\boldsymbol{v}_r = [\cos\theta\ , \sin\theta\ , 0] \tag{28}$$

The vector describing the error in this frame can therefore be expressed as:

$$^{l}\boldsymbol{v}_e = [\cos\theta - 1, \sin\theta\ , 0] \tag{29}$$

And therefore, the magnitude of the error (M_e) is given by:

$$M_e = \sqrt{(\cos\theta - 1)^2 + (\sin\theta)^2} \tag{30}$$

The value for θ has a range 0 to π, and as a result, the M_e can be calculated for every value of θ. Figure 17 shows that this function has a single minimum and will converge at all points excluding the single point where $\theta = \pi$ as the gradient is zero here.

4.3.4 Algorithm Assessment

In order to assess the performance of this algorithm, and to evaluate its performance against both the original algorithm and the improved formulation proposed by Admiraal, the virtual IMU was used. The virtual IMU placed at a random target orientation, and the theoretical sensor outputs for that orientation were passed to the algorithm. The time taken to converge to the target orientation from the initial position was recorded. The values for α (gain) were kept consistent between algorithms to ensure that convergence speed was representative of the number of iterations required to achieve the convergence. Convergence time provides a useful metric for evaluating modifications to the algorithm since it is proportional to the efficacy of the gradient descent. Since the step size is kept consistent, faster convergence means that the interference between the two gradient descent steps has been reduced. The magnetic inclination was set to the local field, which is ~60 degrees. They gyroscopic values were set to 0 degrees per second. All three algorithms were implemented on the same virtual IMU at a given orientation and monitored until all three predicted orientations had converged to within 1˚ of the given orientation. At this point, the virtual IMU was instantaneously rotated randomly, and this process was repeated until 1,000 convergences had been completed.

The number of steps taken for each algorithm to converge was recorded. The average for the 1,000 convergences was converted into a ratio, where:

Original_GDA(Madgwick):Improved_formulation(Admiraal):Extension(Described_here)

was found to be approximately:

6:7:1

An example of the convergence of the three algorithms can be seen in Figure 18.

In order to provide a comparison in ideal circumstances for the original algorithms, the inclination of the simulated magnetic field was changed to be equal to 0˚ (a rare scenario globally, but one where the changes proposed here should have little effect).

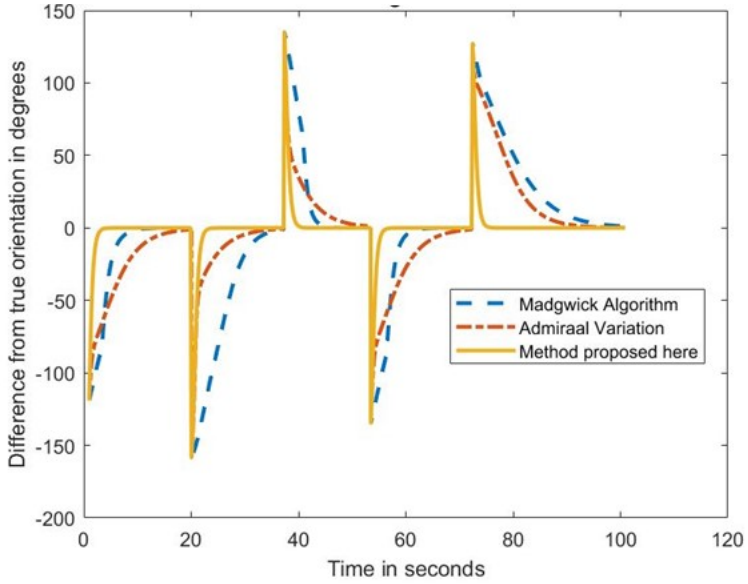

Figure 18 – Evaluation of convergence of different GDAs

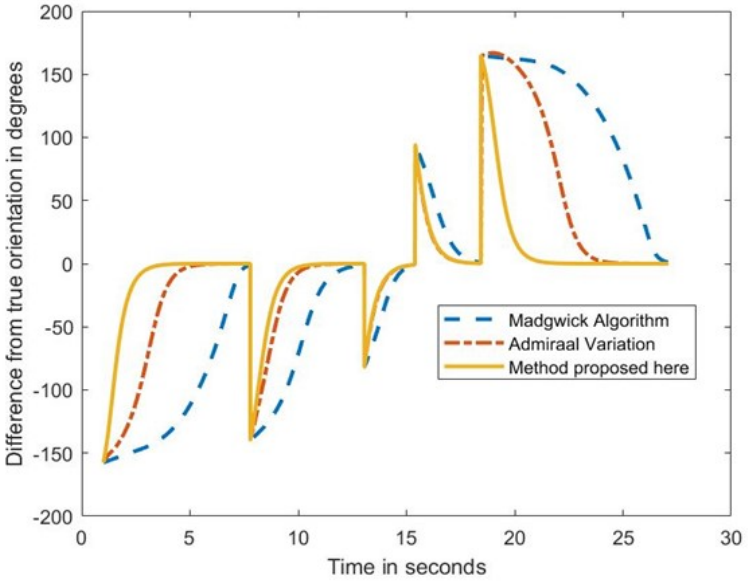

Figure 19 – Evaluation of convergence of different GDAs (no inclination)

In this situation, the ratio of the number of samples to complete 1,000 convergences in the form:

Original GDA (Madgwick):Improved formulation (Admiraal):Extension (Described here)

was found to be approximately:

1.9:1.2:1

An example of the convergence under these circumstances can be seen in Figure 19. This indicates that despite the inclination compensation found in both the Original algorithm and the improved formulation, the amount of magnetic inclination still has a significant effect on convergence time.

To allow the effect of magnetic inclination to be observed in more detail, a final set of convergence tests were run. For these, the virtual IMU was placed at a given orientation, the level of magnetic inclination was set to 0˚, and all algorithms were allowed to converge. The IMU was fixed in the same orientation, but the inclination value was incrementally increased. The results from this test can be seen in Figure 20.

From this, it can be seen that the effects of magnetic inclination are negated through the extension to the algorithm proposed here. This is particularly important for locations where the level of magnetic inclination is naturally high, but it also gives a higher level of robustness in environments where magnetic interference may artificially increase the perceived inclination.

The final point to address from the motivation section is the decoupling of yaw rotation from pitch and roll. To determine whether the axes are now decoupled, the virtual IMU was placed at a random orientation. The three algorithms were allowed to converge, and once they had completed their

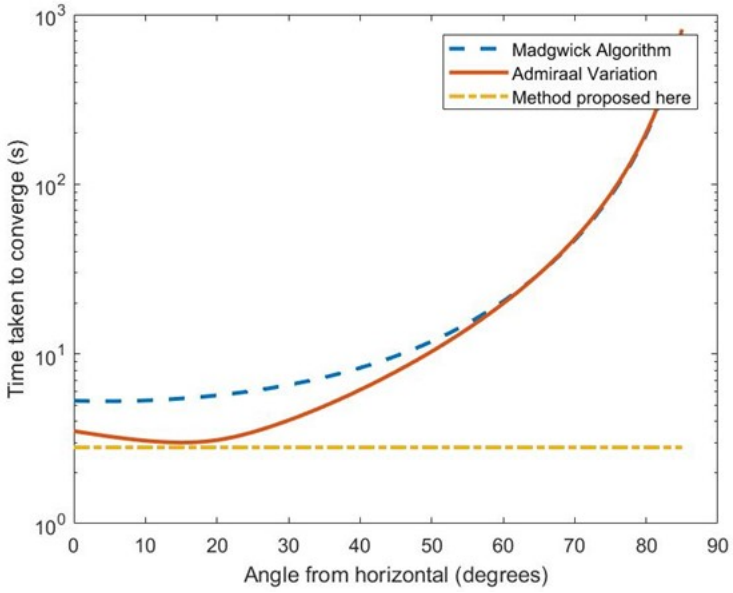

Figure 20 – The effect of magnetic inclination on time to converge

convergence, the MARG was rotated π/2 radians around the z-axis. The algorithm was then allowed to converge to this new orientation, and this convergence was recorded. The quaternions were then converted to be expressed in terms of roll, pitch and yaw. This π/2 radians rotation constitutes a change of π/2 radians in yaw, and therefore when monitoring the second convergence, only the yaw should be affected, pitch and roll should remain the same.

When using the algorithm extension presented here, it was found that the rotation did occur exclusively in the yaw direction, whereas in the original algorithm and the improved formulation, pitch and roll were

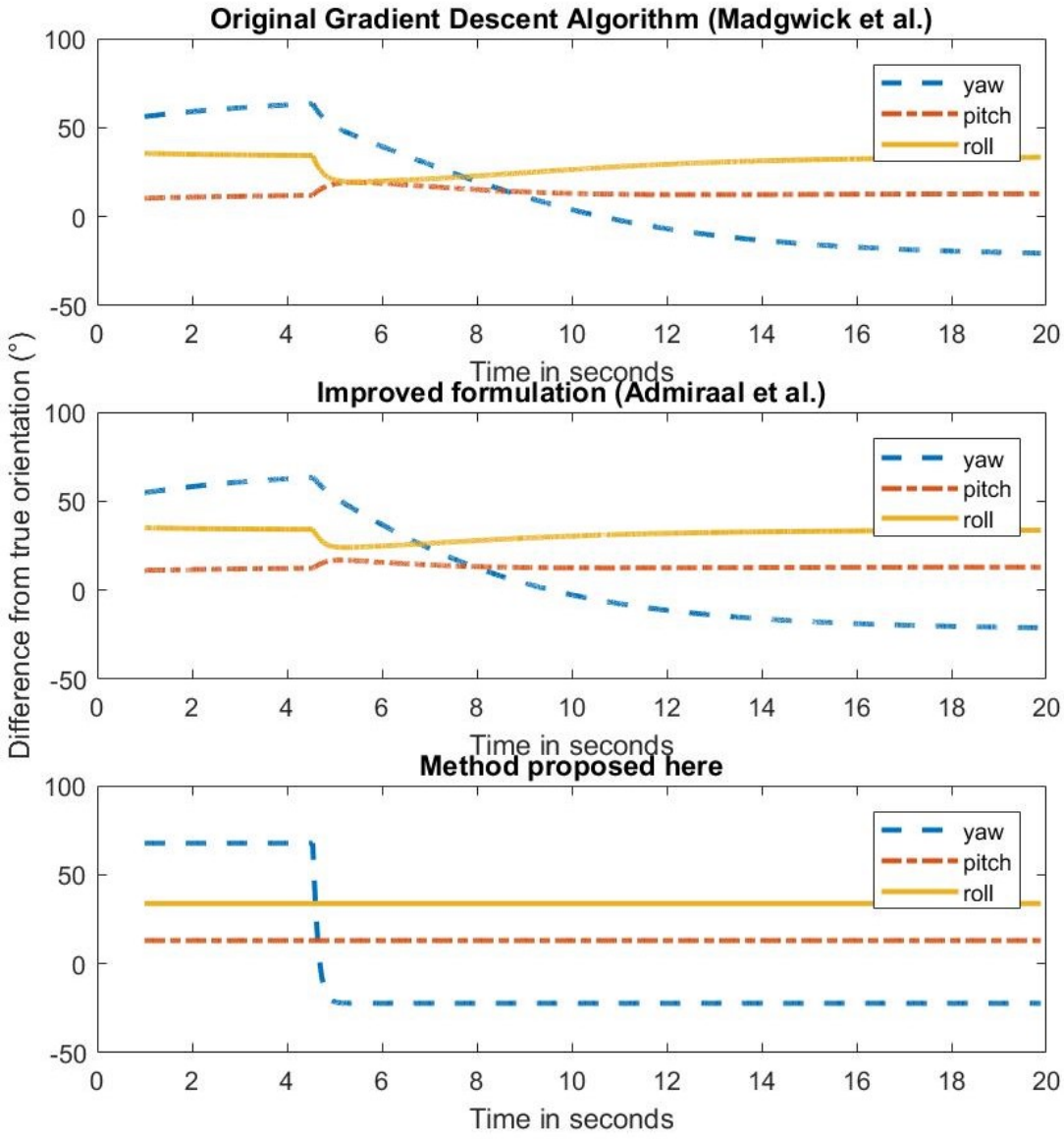

Figure 21 - Result of rotating measured magnetometer ninety degrees around z-axis

both affected. This demonstrates that the yaw is now decoupled from the pitch and roll, addressing the problems discussed in Section 2.4. An example of this second convergence can be seen in Figure 21, the instantaneous rotation of the IMU sensor in the yaw direction takes place at time 4.5s.

The extension presented here addresses a number of the problems with the original gradient algorithm, and therefore can be implemented to achieve a more stable and more predictable corrective step to the gradient descent algorithm.

4.3.5 Robustness to Gyroscopic Bias

Despite the removal of bias in the sensor calibration phase, it is possible that bias may return as the device is used for long periods of time. Introducing a bias to either the magnetometer or the accelerometer will create offsets in the predicted orientation, however this will not have an effect on the output stability. This is not true for biases introduced to the gyroscopic data, where a large enough bias can create an unstable output. The corrective element of the algorithm created by the fusion of magnetic and accelerometer data is intended to compensate for errors introduced by the gyroscopic integration, however if the error is introduced by a large bias, then it may no longer be possible to correct.

In order to examine the effect of gyroscopic bias, the virtual IMU was used. The IMU was placed in a known orientation, and the corresponding sensor inputs were generated. A bias was then introduced in one axis of the gyroscopic sensor data, and the algorithms were observed compensating for this

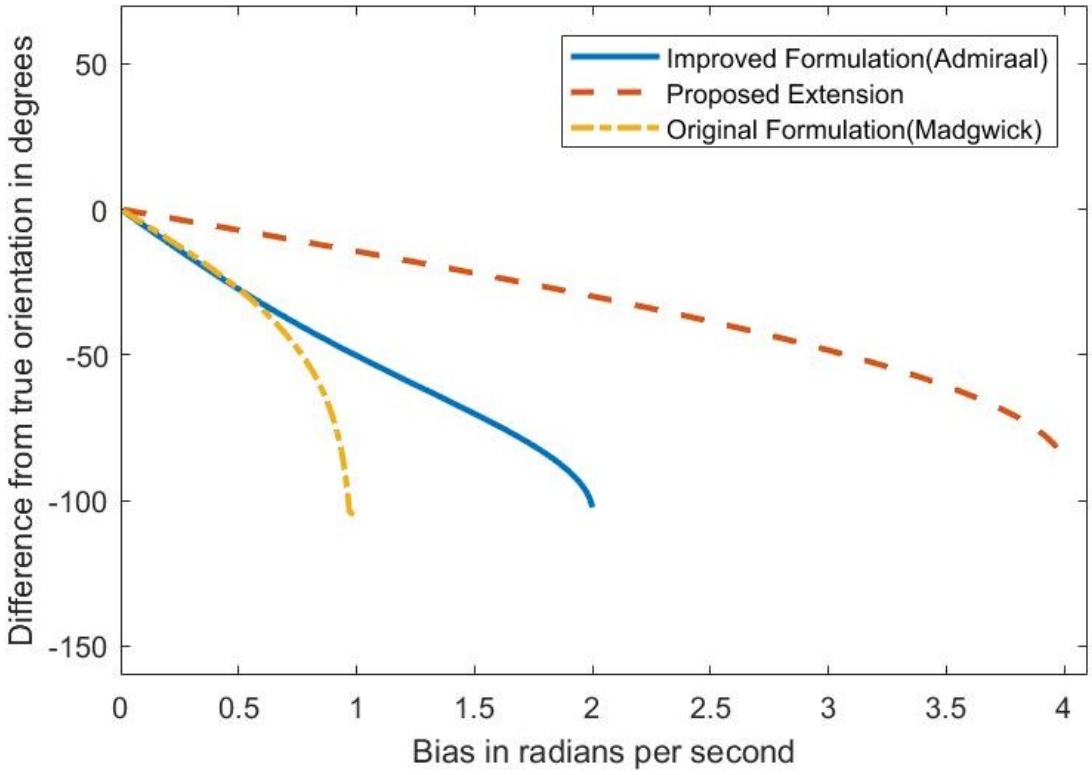

Figure 22 - Effect of gyroscopic bias on convergence

disturbance for 250,000 steps. The final 100 steps were analysed to determine whether the output of the algorithm was stable, and if it was found to be stable, the final orientation was recorded. The bias was then increased, and the process repeated until the algorithm produced an unstable result.

Through this experiment, it was found that the method proposed here was able to compensate for a larger range of gyroscopic biases. The difference between the output and the true orientation was smaller for low biases, and the algorithm remained stable for larger biases, which caused the other tested algorithms to become unstable. The results of this can be seen in Figure 22.

4.4 New Algorithm Formulation

4.4.1 Motivation

While the improvements to the gradient descent improve the accuracy of the corrective step, there are still undesirable features. The primary problem is that the rate of correction cannot be accurately defined, and therefore determining the best gain for given frequencies of signal and sensor noise is difficult. In addition, the rate at which the algorithm converges is dependent on the amplitude of the error, and therefore the speed of convergence is inconsistent. Finally, since the error is modelled on spherical coordinates, as the error approaches π, the gradient begins to become less steep, leading to a slower initial convergence speed.

4.4.2 Solution

The solution proposed here addressed these problems through the calculation of an instantaneous convergence. As a stand-alone algorithm, it therefore assumes that the measured vectors for gravity and the geomagnetic field are accurate. As a result, the full implementation requires a dynamic weighting system, which determines when the measured vectors can be fully trusted, and when the gyroscopic data will be more reliable. Gradient descent-based approaches do not use this nature of dynamic weighting, since small regular corrective steps will act as a low pass filter on these vectors. This is an inefficient method of providing this filtering, since it requires the algorithm to be executed at the introduction of any new data. Performing this filtering through a combination of lightweight signal processing and simple sensor fusion to provide weighting, combined with one-step convergence removes the requirement to execute the algorithm at every step. It can instead occur at a lower frequency, dependent on the rate of change of the sensor outputs. Assuming initially that the sensors are providing reliable geomagnetic and gravitational data, the orientation can be determined using the following process.

As Madgwick observed, orientation can be defined through two non-parallel vectors, which can be found in the forms of the gravity vector and the magnetometer vector. The rotation that is required to align both the measured and reference accelerometer vector and the measured and reference magnetometer vector can be broken down into two consecutive rotations, one that corrects based on the accelerometer reading and one that corrects based on the magnetometer reading. The process here states that the orientation estimation is correct if the measured vectors are aligned with the reference vectors after the reference vectors have been transformed to the local reference frame using the predicted orientation.

As in previous notation, the reference vector for gravity ($v_{r(a)}$) is defined as:

$$v_{r(a)} = [0,0,-1] \tag{31}$$

While the calculation can be conducted in either reference frame without affecting the efficiency, in the implementation outlined here, this reference vector is rotated into the local reference frame as described in by multiplying by the current estimation for orientation (q), to give (${}^{L}v_{r(a)}$).

Using (6) in (2.8) gives:

$${}^{L}v_{r(a)} = \boldsymbol{q} * \boldsymbol{v}_{r(a)} * \boldsymbol{q}^{-1} = R_{\boldsymbol{q}} * \boldsymbol{v}_{r(a)} \tag{32}$$

Which can be easily solved using the rotation matrix given in (2.9) to give:

$${}^{L}v_{r(a)} = \left[2\left(q_x q_z - q_w q_y\right), 2\left(q_w q_x + q_y q_z\right), 2(q_w^2 + q_z^2) - 1\right] \tag{33}$$

The rotation required to correct the measured gravity vector ($v_{m(a)}$) to equal the current predicted sensor output (${}^{L}v_{r(a)}$) can be broken down into two parts, the angle between the measured vector and the rotated reference vector must be found, and the axis around which a rotation of this angle should occur must also be found.

The axis of the rotation ($axis_a$) required to bring two vectors into alignment is one that is perpendicular to both, and therefore can be found using the cross product of the two vectors:

$$axis_a = {}^{L}v_{r(a)} \times v_{m(a)} \tag{34}$$

Next the angle (θ_a) between the two vectors is generated according to:

$$\theta_a = \cos^{-1}(\widehat{{}^{L}v_{r(a)}} \cdot \widehat{v_{m(a)}}) \tag{35}$$

This correction can then be expressed as a quaternion (q_a) through:

$$\begin{aligned} q_{aW} &= \cos\left(\frac{\theta_a}{2}\right) \\ q_{aX} &= axis_{aX} \cdot \sin\left(\frac{\theta_a}{2}\right) \\ q_{aY} &= axis_{aY} \cdot \sin\left(\frac{\theta_a}{2}\right) \\ q_{aZ} &= axis_{aZ} \cdot \sin\left(\frac{\theta_a}{2}\right) \end{aligned} \tag{36}$$

The second step requires correction based upon the magnetometer vectors. As in the previous section, the effects of magnetic inclination should be removed. This is performed using the same method:

$$v_{r(m)} = [v_{rx}, 0, v_{rz}] \tag{37}$$

$$V_{r(e)} = |[0,0,-1] \times [V_{rx}, 0, v_{rz}]| \tag{38}$$

$$V_{r(e)} = |0,-1,0] \tag{39}$$

The reference vector is again transformed to the local reference frame:

$$^{L}v_{r(e)} = q.v_{r(m)}.q^{-1} \tag{40}$$

Which is simplified using (2.8) and (2.9) to give:

$$^{L}v_{r(e)} = \left[2\left(q_x q_y + q_w q_z\right), 2\left(q_x^2 + q_y^2\right) - 1, 2\left(q_y q_z - q_w q_x\right)\right] \tag{41}$$

And the measured magnetic vector $v_{m(m)}$ must also be converted to magnetic east to remove inclination. The rotation defined through the accelerometer correction must then be applied to the measured vector to ensure that the effect of this rotation is only applied once. This can be summarized as:

$$v_{m(e)} = q_a \cdot (\hat{v}_{m(a)} \times \hat{v}_{m(m)}) \cdot q_a^{-1} \tag{42}$$

The axis of rotation (parallel to the measured acceleration) is given by:

$$axis_e = {}^{L}v_{r(e)} \times v_{m(e)} \tag{43}$$

Again, the angle can be calculated:

$$\theta_e = \cos^{-1}(\widehat{{}^{L}v_{r(e)}} \cdot \widehat{v_{m(e)}}) \tag{44}$$

And the second correction rotation q_e can be calculated:

$$\begin{aligned} q_{eW} &= \cos\left(\frac{\theta_e}{2}\right) \\ q_{eX} &= axis_{eX} \cdot \sin\left(\frac{\theta_e}{2}\right) \\ q_{eY} &= axis_{eY} \cdot \sin\left(\frac{\theta_e}{2}\right) \\ q_{eZ} &= axis_{eZ} \cdot \sin\left(\frac{\theta_e}{2}\right) \end{aligned} \tag{45}$$

The overall correction q_C to be applied is the product of these two correction quaternions:

$$q_c = q_a * q_e \tag{46}$$

This allows the correction to be applied in a single step. This immediate convergence is useful for stationary and noiseless data, however for practical use, it is more useful to implement this algorithm within a complementary filter. This may require the correction rotation to be broken down into steps. The simplest way of doing this in a system with enough computational resources is to use spherical linear interpolation, which is commonly used in computer graphics, and therefore found in many graphical toolboxes. It may be desirable to avoid this method of interpolation if computational resources are low.

Since this algorithm used Axis-Angle rotations to calculate its corrective rotations, this can be approximated without the added computational power of spherical linear interpolation by applying a weight to the calculated angles.

For a given weight (k), where:

$$0 \leq k \leq 1 \tag{47}$$

The rotations representing the partial gravity correction q_{ap} and partial magnetometer correction q_{ep} can be calculated as:

$$\begin{aligned} q_{apW} &= \cos\left(k \cdot \frac{\theta_a}{2}\right) \\ q_{apX} &= axis_{aX} \cdot \sin\left(k \cdot \frac{\theta_a}{2}\right) \\ q_{apY} &= axis_{aY} \cdot \sin\left(k \cdot \frac{\theta_a}{2}\right) \\ q_{apZ} &= axis_{aZ} \cdot \sin\left(k \cdot \frac{\theta_a}{2}\right) \end{aligned} \tag{48}$$

and

$$\begin{aligned} q_{epW} &= \cos\left(k \cdot \frac{\theta_e}{2}\right) \\ q_{epX} &= axis_{eX} \cdot \sin\left(k \cdot \frac{\theta_e}{2}\right) \\ q_{epY} &= axis_{eY} \cdot \sin\left(k \cdot \frac{\theta_e}{2}\right) \\ q_{epZ} &= axis_{eZ} \cdot \sin\left(k \cdot \frac{\theta_e}{2}\right) \end{aligned} \tag{49}$$

respectively.

There are a number of reasons that this approach is desirable. The first is that the rate of convergence is predictable. For (N) steps and (k) gain, if the initial error is defined as (er_0), the error at each step is given by:

$$er = er_0 - \sum_{n=0}^{N} er_0 * k^n \tag{50}$$

If $k = 1$ then convergence will occur in one step, otherwise it will tend to convergence as described above.

Since this algorithm does not rely on an iterative process to achieve convergence while stationary, there are a number of modifications that can be made, for example, if the convergence rate should be limited to a maximum speed in yaw, then $\frac{\theta_e}{2}$ can be limited to this maximum (maximum will be achieved if $axis_e \| v_{m(a)}$).

Another potential benefit of this immediate convergence is that the calculation of the required corrective step does not need to be made at every time interval, since the corrective step is designed to compensate for the effects of low frequency gyroscopic drift.

4.4.3 Convergence Rate

The convergence rate of the algorithms outlined here is dependent on their respective value for gain. The effect of gain is no longer consistent between these algorithms, and therefore they become difficult to directly compare to one another. The method used here was to find the value for gain that gives the fastest convergence for a given rotation to occur for each algorithm. For the algorithm introduced in this section, the most efficient gain value is 1, since this will give a 1-step convergence. For the Gradient Descent-based algorithms, the gain is proportional to the step size. A smaller gain will lead to slow convergence, whereas a gain that is larger than optimum will lead to an oscillation that slows (or completely inhibits) convergence. The ideal gain was dependent on the required rotation, and the number of steps taken to converge is also dependent on the initial offset.

To evaluate the gain value that gave the fastest convergence for the GDA, the gain was varied, and the time taken to converge was observed. Initially a small gain was provided, and the number of steps taken for the algorithm to converge was counted. The gain was then increased, and the increased step size led to a decrease in the number of steps required. This was continued until the number of steps began to increase, since this was an indication that the step size was causing oscillations to occur in the convergence. At each minimum in the number of steps, the gain and steps required were recorded. In order to cover a wide variety of rotations, both simple and complex, this process was repeated 1,000,000 times. When using the optimum gain for a given orientation, the average steps taken to achieve convergence with an accuracy of 1˚ with a sample period of 0.001 seconds was approximately 65. As a result, it can be concluded that at this frequency, the new formulation converges 65 time faster than the gradient descent.

4.4.4 Efficiency – Base Algorithm

To evaluate efficiency, the time taken for the algorithm to run was observed. To perform this evaluation, both algorithms were set to an initial random orientation. A second random orientation was set as the target, and the algorithms calculated their corrective steps. These corrective steps were ignored, so convergence was never achieved. As a result, the time taken to complete a fixed number of steps could be measured. This was repeated for 1,000 initial position/target position pairs. It was found that on average, the time taken for the GDA to perform 1,000,000 steps was approximately 0.26 seconds. Conversely, it took 0.92 seconds for the implementation outlined here to perform the same number of convergences. As a result, it can be concluded that for this method to lead to an increase in efficiency, the implementation must call it for less than 28% of samples.

4.5 Chapter Summary

This chapter describes the process by which the orientation output of the IMUs was calculated. After an evaluation of the existing orientation estimation techniques, Madgwick's Gradient Descent Algorithm was chosen and implemented. Calibration techniques for both the gyroscopic sensors and the magnetometer sensors were implemented, resulting in a fast method of calibration that increases the likelihood of accurate orientation. The accuracy of the base algorithm was assessed by comparing against a known orientation provided by a camera-based system (Optotrak), which resulted in an average accuracy of 5.7 degrees.

The algorithm was implemented in the firmware of the IMUs to allow compatibility with low power host devices. In order to maintain the desired sample rate, several modifications were required including a type conversion from floating point numbers to integers, and custom normalization functions for three and four element vectors. This decreased the time required to execute the algorithm from ~3ms to ~1ms.

Several previous works have highlighted limits of the Madgwick algorithm that cause unexpected behaviour in experimental conditions. Euler angles are coupled, which leads to indirect convergence, and the magnetometer reference vector is dependent on local magnetic inclination, leading to inefficiencies in the algorithm which can lead to slow convergence. An additional step has been demonstrated that addresses both problems simultaneously, leading to fast, predictable convergence that is independent of factors such as inclination. The solution was tested in a simulated environment, which allows its performance to be compared to both the original formulation, and the improvements proposed by Admiraal.

Finally, a formulation of a new algorithm that calculates the precise orientation based on a single set of sensor data was presented. This new algorithm allows single step convergence, which removes the unpredictable convergence rate achieved through the gradient descent, allowing more directed filtering efforts to be implemented. The efficiency and convergence rate of this algorithm are discussed.

Chapter 5

Using MMG for Activity Monitoring

5.1 Activity Classification

The sensors system outlined in Chapter 3 was designed to allow hand gestures to be used as the basis for an HMI. The sensing modality does not directly require movement or dexterous manipulation, and therefore it is suitable for both healthy users and for amputees. The form of the interface is an armband that can be worn on the forearm and operates an 'always on' sensing protocol.

In order to function as an interface for Human-Machine interaction, a device must have the ability to detect volitional signals generated by the user for control. For a pervasive system such as the one described here, this involves both identifying when the user is attempting to interact with the device, and what the intent behind the interaction is. Both can be achieved using an activity classifier, where the default state is that the user is not attempting to interact with the interface, and additional states represent detected user intent. When this system is worn on the forearm, it can provide a number of categories of information to an activity classifier, including limb orientation, inertial gestures and muscle activity. The stream containing the most information will be the muscle activity, since the muscles in the forearm control the large number of degrees of freedom in the hand. Describing the required control signal in the form of hand gestures also provides a user-friendly interface, since users are required to perform actions that they are likely to be familiar with, reducing the amount of learning required to operate the system. For prosthetic control, capturing the muscle activity associated with performing hand gestures as a method of determining which grip to select has the potential of eliminating the cognitive burden associated with prosthetic control, and therefore the cognitive barrier between the wearer and their prosthetic. This requires a pattern recognition-based control system.

Regardless of the application, the methodology is consistent, and has the following steps:

1. The device must register that a volitional interaction is being made.
2. The signals representing that volitional interaction must be extracted from the data stream.
3. The extracted signals must be classified to determine the action that caused them to be generated.
4. The action and any available gesture context must be used to determine the intent behind the interaction.
5. The intent must be implemented by the end effector.

MMG has been not been studied as extensively as some of its HMI sensing counterparts, and as a result, there is not yet a consensus on the most effective methods of deriving control information from it. As discussed in Chapter 2, there have been several attempts to use MMG signals for prosthetic control, however factors such as the rejection of motion induced artefacts have imposed limits on the practical implementation of the system. As such, this chapter describes the creation of a gesture database, and the subsequent processes of gesture extraction and classification that were designed. It also describes the implementation of these methods in several real-time tests. This chapter can be summarized as the design and implementation of an activity classifier.

5.2 Database Generation

While previous studies have examined the topic, little is known about the individuality of MMG signals for subjects performing the same movement. In order to ensure that the methodology used by the activity classifier was not dependent on features specific to an individual, a database containing a number of examples of gestures from different individuals was created. This database would then function as an initial offline testing domain for various algorithms.

5.2.1 Participants

The volunteers selected to participate in the creation of this database gave their informed consent before the data collection began. The inclusion criteria were that volunteers must be above 18 and should have no physical impairment other than amputation. All experiments outlined in the following chapters were approved by the Imperial College Research Ethics Committee (ICREC reference: 15IC3068).

Six healthy subjects (3 male, 3 female, average age: 36.8yrs SD: 15.8yrs) and one transradial amputee (male, age: 36yrs) were involved in this work. The healthy subjects had no visible abnormalities to the skin. The amputee subject had a transradial amputation on their right arm, as well as amputations on the upper section of the other three limbs. The transradial amputee subject underwent his amputations after an infection that lead to Toxic Shock Syndrome (TSS), septicaemia and necrotising fasciitis. After both lower limbs and the left upper limb was amputated, the necrotizing flesh was removed from the right upper limb, which then had to be reconstructed. While this was initially successful at preserving the right limb, the infection persisted in the bone marrow, while eventually resulted in a transradial amputation in this limb. As a result of the extensive surgeries, reconstruction and subsequent amputation, the subject has a unique physiology in the radial section of their arm, and extensive scarring around that area.

5.2.2 Protocol

At the beginning of each recording session, subjects were invited to wear an armband that comprised of a single IMU and six MMG sensors on their right forearm. The armband was placed around the largest radius of the forearm, with the IMU uppermost on the arm when the arm was held horizontally in front of the user with the palm facing down. Subjects were seated in front of a table, so that their elbow could be rested comfortably, and they were not required to hold the weight of their arm. Additionally, the height was adjusted so that the forearm and hand could be held in a vertical position without additional effort. This was to limit the amount of background muscle activity required to maintain the position of the hand.

At intervals given by the data collection system, subjects were required to make instances of seven gestures. Data from each subject were stored and labelled according to which gesture the subject had been instructed to make. The gestures were selected based on six of the pre-programmed grip patterns that can be found on the Bebionic V2 (RSL Steeper), as well as an ‘open’ gesture. A number of these gestures were modified slightly to reduce ambiguity between similar gestures. An example of similar gestures is the Pinch (first finger to thumb) and Tripod (first and second finger to thumb) gestures, where the tripod gesture was modified to only include the second finger. In total, the gestures used for this period of data gathering were: Open the hand, Make a fist, Pinch with first finger, Pinch with second finger,

Raise Thumb, Point with first finger and Roll fingers. Each of the healthy subjects recorded one hundred instances of each gesture.

Due to their unique physiology, the amputee subject followed a slightly different protocol to the rest of the subjects. They reported that they could not feel any contraction associated with performing three of the gestures. As a result, the subject proposed three replacement gestures that were performed instead. Pointing with the first finger was replaced with tapping with the first finger and raising the thumb and pinching with the second finger were replaced by rotating the wrist clockwise and anticlockwise. The subject also reported a faster onset of fatigue than the other subjects, and as a result their session was reduced and fifty instances of each of the seven gestures were recorded.

In addition to this, healthy subjects were asked to perform five additional gestures to examine whether the sensor system was capable of distinguishing different contractions from the same muscle group. Subjects were asked to sit in a chair with their right arm relaxed and hanging to their side. They were then asked to twitch each of their digits in turn, starting with their thumb and moving to their fourth finger. This was to capture different contractions from the flexor digitorum muscle group (responsible for flexing the fingers), as well as the nearby flexor pollicis longus. Each of the healthy subjects also recorded one hundred instances of these gestures.

Finally, a list consisting of thirty-eight gestures was compiled describing the dexterous movements required for several ADLs. One subject completed a dataset comprising one hundred instances of each of these thirty-eight gestures. These gestures included large movements, such as opening or closing the hand and rotating the wrist, small movements, such as individual finger flexion and extensions, as well as more dexterous movements, such as moving a mouse with the fingers or pressing the different mouse buttons.

As a result, the final database consisted of 10150 examples of unsegmented gestures. These gestures were split into files, each of which contained an uninterrupted stream in which ten gestures occurred. The data consisted of MARG data and the mechanomyographic data recorded from six sites on the participants forearm. Each file was run through an activity monitor, and the presence of activity within the signal was highlighted. Each highlighted section of the signal was then manually examined and marked if it contained a gesture. This process of manually marking gestures was to allow an evaluation of automatic segmentation algorithms to take place. Additionally, each segmented gesture file contained the gesture the subject had been instructed to make, how far through the dataset the subject was at that point, and which subject had generated the signals. This database therefore contained enough information to provide a test environment for both gesture segmentation and gesture classification algorithms to be evaluated.

5.3 Gesture Segmentation

To perform event detection on a continuous data stream, the properties of that stream need to change. When monitoring the MMG signal from the forearm, one indicator that the muscle activity required to actuate the hand may be occurring is an increase in the energy of that signal. The signal from each MMG

sensor can be defined by the vector m. The signal was band-pass filtered between 1Hz and 50Hz to produce m^*. The energy e_{sig} in the signal at any given point can be defined as:

$$e_{sig} = \sum_{j=1}^{N} m_j^{*2} \tag{51}$$

Where N represents the total number of sensors and j is a specific sensor.

Analysis of the database allowed the energies which were most representative of the different gestures to be determined, as well as providing an indication of the background energy level. The value for the energy threshold (T_{MMG}) which identified gestures with the highest f-measure was determined. F-measure provides a statistical measure of a classifiers accuracy by combining both precision and recall as follows:

$$Fmeasure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{52}$$

In order to determine the most appropriate value for T_{MMG}, the following protocol was used. T_{MMG} was evaluated as the maximum value of e_{sig} present in the database, and the f-measure was calculated. T_{MMG} was then incrementally reduced, and the f-measure was observed to increase as the number of true positives were identified, and then decrease as the number of false positives also increased. The value of T_{MMG} which provided the highest accuracy according to the f-measure was taken as the value for T_{MMG} in the experiments going forward.

It has been noted previously that the energy of the MMG signal also increases in the presence of motion induced artefacts. Since the sensor suite contains an IMU, the motion of the limb could be examined. A motion threshold ($T_{g_{xyz}}$) was determined by observing the energy of the gyroscopic data ($\|G\|$) present during the segmented gestures in the database and selecting a value for $T_{g_{xyz}}$ where $T_{g_{xyz}} = \max(\|G\|)$.

Using these two thresholds, the gesture detection algorithm can therefore be summarized as:

$$Gesture\ Detected = \begin{cases} true & if \quad \|G\| < T_{g_{xyz}}\ AND\ e_{sig} > T_{MMG} \\ false & otherwise \end{cases} \tag{53}$$

Many of the gestures recorded had different durations, however the gesture duration was not sufficient for classification since it was not consistent. When choosing the length for the sample window, consistency throughout the gesture and application were both considered. As gestures were performed, differences in strength and speed of movement were observed to lead to inconsistencies in signal as the gesture progressed. Additionally, it was desirable for some form of feedback to be provided to the user

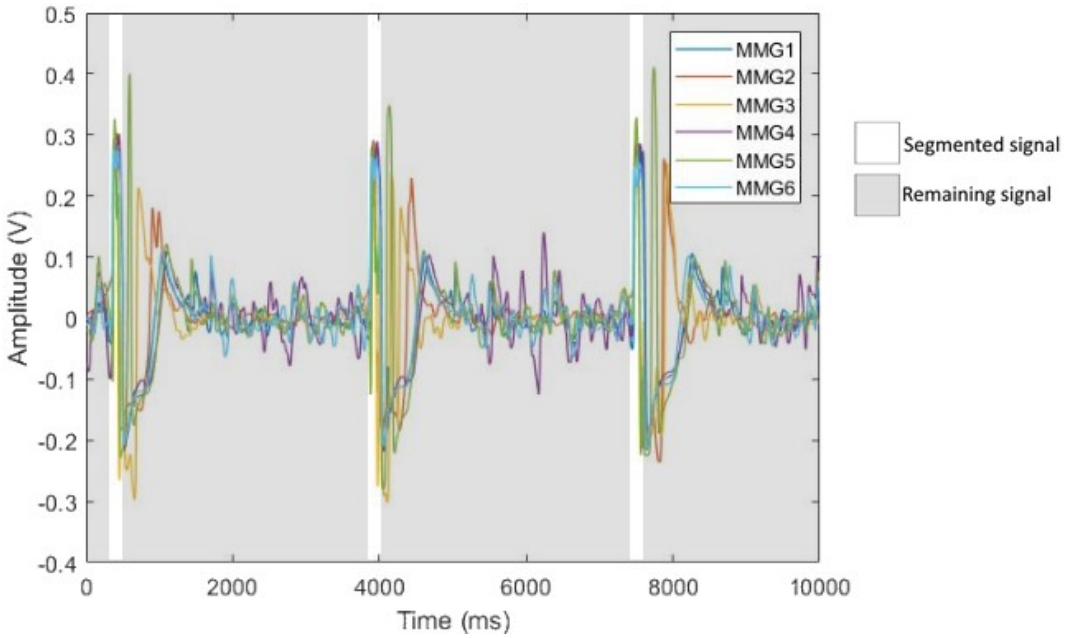

Figure 23 - Three 'Open' gestures identified and segmented from non-amputee subject

within 0.3 seconds, since this has previously been described as the maximum acceptable latency of HMI applications. As a result, a sample period of 0.2 seconds was selected for classification. At 1kHz, this corresponded to 200 samples. On average, the energy of the signal took 0.05 seconds to rise from the relaxed level to the T_{MMG} threshold, and so the algorithm extracted 0.05 seconds prior to the signal being detected, and 0.15 seconds after detection. If the point where the gesture is detected is defined as time i, then the extracted signal s_N from N MMGs therefore is given as:

$$s_N = [m_{N_{i-b}}, \dots, m_{N_{i+(a-b)}}] \tag{54}$$

where a is the length of the recording (200 samples), and b is the number of samples taken to reach the activation energy. The a by N matrix (s) can then be classified.

Before the classification took place, the signals were examined and compared to previous literature. Orizio [120] provided a description of MMG signal origins, and attributes the signals to a combination of factors, including a gross dimensional changes at the onset and offset of the effort, and the rate of muscle fibre recruitment during sustained contractions. The dominant frequency of these factors is dependent on the rate of movement and effort required, however, the largest frequencies present in the recorded signals tended to fall between 2Hz and 15Hz, indicating that the primary contributor to the generated signals is likely to have been the gross dimensional changes of the muscle during initial contraction.

5.4 Gesture Classification

5.4.1 Template-based Classification

The MMG signals were found to be consistent between different instances of the same gesture. As a result, a template-based pattern recognition system provided a computationally inexpensive method of classifying the extracted data segments. The database instances of the gesture were randomly assigned to either a training set or a test set. The initial ratio of training data to test data was 70%:30%, however it was found that the after approximately thirty gestures, accuracy no longer improved significantly.

Templates for each of the gestures were constructed by finding the average of the gestures in the training set. The method of segmentation of the data containing the gesture removed the need for alignment. For a training set of size g, an N dimensional template $\tilde{S}$ (from N MMGs) is expressed as:

$$\tilde{S}_N = \frac{1}{g}\sum_{k=1}^{g} s_{N_k} \tag{55}$$

After the templates had been created, they could be used to classify new data. This was achieved by correlating each channel of the new data to the corresponding channel in every template. The correlation

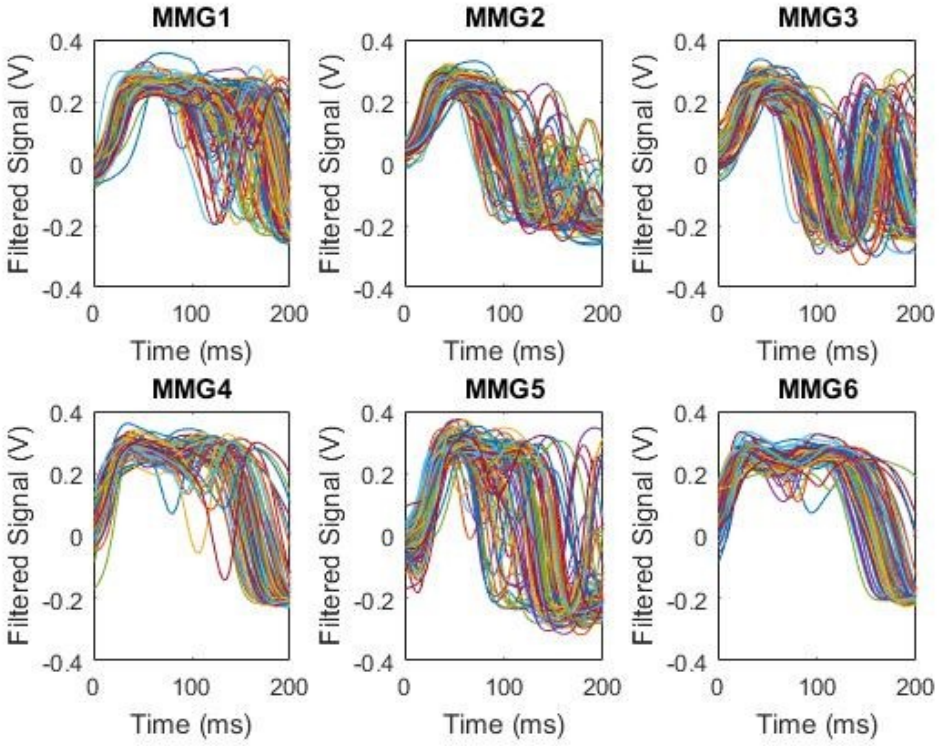

Figure 24 - One hundred instances of the 'Open' gesture from one non-amputee subject

was conducted using an implementation of the Pearson Product-Moment Correlation Coefficient (denoted by ρ). The correlation between s_N and $\tilde{S}_N$ is given as:

$$\rho_{s_N \tilde{s}_N} = \frac{\sum_{j=1}^{b} \left[s_{N_j} - \overline{s_N} \right] \left[\tilde{S}_{N_j} - \overline{\tilde{S}_N} \right]}{\sigma_{s_N} \sigma_{\tilde{S}_N}} \quad (56)$$

Where $\overline{s_N}$ and $\overline{\tilde{S}_N}$ are the average and σ is the standard deviation of each vector. The Pearson Product-Moment Correlation Coefficient will give a value for correlation ranging from -1 to 1. As a result, the coefficient can be normalized for N MMGs using:

$$\rho_{s\tilde{s}} = \frac{1}{N} \sum_{n=1}^{N} \rho_{s_N \tilde{s}_N} \quad (57)$$

A vector containing the correlation values ($\rho_{s\tilde{S}}$) for all templates can be constructed. The value that has the highest value in the vector is given as the systems prediction for the gesture.

5.4.1.1 Offline Accuracy

To evaluate the accuracy of this method of classification, gestures were randomly split 30:70 into a training set and a test set. The training set was used to create templates for each of the gestures. Each gesture in the test set was then correlated to each of their templates, and the gesture with the highest correlation value was given as the systems prediction (Gesture Classified). This process of randomly assigning the data to training/test sets and calculating the accuracy was repeated 100 times, and the accuracies averaged, to reduce the likelihood that the accuracies found were not representative of the data. Data from each subject were tested independently from data from the rest of the subjects. On average this led to an average accuracy of 82.9% (SD: 8.4%) across all non-amputee subjects. An example confusion matrix generated by compiling classification accuracies from each subject can be seen in Table 5.

In each case, the hand was relaxed before the gesture, so that no volitional muscle activity was present before the gesture occurred. Subjects were also instructed to perform the gestures as a swift, single movement. The gesture ID numbers were assigned as follows:

(With elbow on table)

1. Hand fully opened
2. Hand fully closed
3. Pinch first finger and thumb

Table 5 - Confusion matrix showing classification accuracies of template-based classification for all non-amputee subjects

		Gesture Classified											
	ID	1	2	3	4	5	6	7	8	9	10	11	12
Actual Gesture	1	84.5%	2.9%	0.6%	0.6%	0.0%	0.6%	2.3%	0.6%	4.0%	1.1%	2.9%	0.0%
	2	0.6%	94.3%	0.0%	0.0%	0.6%	1.1%	0.6%	1.7%	0.0%	0.0%	1.1%	0.0%
	3	1.1%	1.7%	84.7%	5.1%	2.3%	0.6%	0.6%	1.1%	1.7%	0.6%	0.6%	0.0%
	4	4.5%	0.6%	6.8%	80.2%	0.6%	1.7%	1.1%	0.6%	0.6%	1.7%	1.1%	0.6%
	5	1.1%	0.0%	2.3%	1.7%	84.5%	2.3%	0.0%	0.6%	0.6%	0.6%	1.7%	4.6%
	6	0.0%	3.5%	0.6%	2.3%	4.6%	86.1%	0.0%	0.6%	0.6%	1.2%	0.6%	0.0%
	7	4.5%	3.2%	0.6%	3.9%	1.9%	1.9%	68.8%	1.9%	1.9%	3.2%	3.2%	4.5%
	8	0.0%	1.3%	0.0%	0.0%	1.9%	3.1%	5.0%	75.5%	3.1%	6.3%	3.8%	0.0%
	9	1.8%	0.6%	3.5%	0.0%	1.2%	0.6%	0.6%	0.0%	85.4%	3.5%	1.2%	1.8%
	10	1.8%	1.2%	0.0%	0.6%	0.0%	0.6%	1.8%	0.0%	3.0%	86.4%	4.7%	0.0%
	11	0.0%	0.6%	0.0%	0.0%	1.1%	0.0%	0.6%	3.4%	0.6%	2.3%	87.4%	4.0%
	12	2.3%	0.0%	0.0%	0.6%	5.8%	2.3%	0.6%	0.6%	3.5%	0.6%	6.4%	77.3%
										Average Accuracy			82.9%

Table 6 - Confusion matrix showing classification accuracies of template-based classification for amputee subject

		Gesture Classified						
	ID	1	2	3	4	5	6	7
Actual Gesture	1	74.0%	6.0%	0.0%	14.0%	0.0%	4.0%	2.0%
	2	11.8%	64.7%	5.9%	0.0%	7.8%	7.8%	2.0%
	3	3.8%	7.7%	86.5%	0.0%	0.0%	1.9%	0.0%
	4	29.6%	1.9%	0.0%	64.8%	0.0%	3.7%	0.0%
	5	1.9%	18.5%	3.7%	0.0%	40.7%	31.5%	3.7%
	6	2.0%	13.7%	3.9%	0.0%	15.7%	64.7%	0.0%
	7	1.9%	26.4%	0.0%	0.0%	11.3%	5.7%	54.7%
					Average Accuracy			64.11%

4. Pinch second finger and thumb
5. Raise thumb
6. Extend first finger
7. Extend all fingers sequentially (forth finger to first finger)

(With arm hanging by side)

8. Twitch thumb towards palm
9. Twitch first finger towards palm
10. Twitch second finger towards palm
11. Twitch third finger towards palm
12. Twitch forth finger towards palm

5.4.1.2 Real-time Implementation

Several modifications were required to allow the system to work efficiently in real-time.

The most fundamental change was that the band-pass filter previously applied to the entire dataset was replaced with two consecutive first order transfer functions. As a result, data could be filtered in real time, without requiring a large amount of historical data.

Saving all data from the each MMG channel during processing was deemed to put an unnecessary load on the available memory resource. To combat this, two buffers were created to store the data, one of length 200 (length a) referred to as A and one of length 50 (more generally, length b) referred to as B, and both with a width of 6 (N MMG channels). The B_b was implemented as a circular (or ring) buffer. Circular buffers are an implementation of a First In First Out (FIFO) data structure designed to allow a short amount of historical data to de retained without adding the computational expense. A pointer was used to determine where in the buffer to write the current data. When the pointer reaches the end of the buffer and is subsequently incremented, it will return to the beginning of the buffer and overwrite the information there. The pointer p is incremented using a modulo operator (the notation for which is %), as follows:

$$p_c = (p_{c-1} + 1)\%L_b \tag{58}$$

Where c denoted the current time step, and L is the buffer length (b in this case).

Once an energy threshold was reached, data from B was copied over to A, such that:

$$A_{0:b} = [B_{(p+1):b}, B_{0:p}] \tag{59}$$

The remaining 150 elements of data were then written to A in real time.

5.4.1.3 Real-time Accuracy

The protocol for real-time validation of this segmentation/classification strategy was similar to the protocol for the data collection. Subjects wore a sensor system consisting of one IMU and three MMG modules over the flexor and extensor muscle groups on their right forearm.

Four healthy subjects and one amputee subject were recruited for this study. The accuracies of the segmentation and classification protocols were examined over the course of three short tests. For each of these tests, a subset of the gestures types used in the database generation was selected.

The subsets selected for each of the tests were as follows:

Experiment 1 – Gestures 1 and 2 (appropriate for simple prosthetic control)

Experiment 2 – Gestures 1, 2, 3, 4 and 6 (appropriate for more advanced prosthetic control)

Experiment 3 – Gestures 8, 9, 10, 11 and 12 (appropriate for a pervasive HMI application)

At the beginning of the session, subjects were asked to make thirty instances of each of the gestures used in the test. These were then used to create the templates for classification. Subjects were then asked to observe a screen, and to perform a gesture when its name appeared on the screen.

The accuracy of the segmentation protocol was established by the ability to detect a gesture within three seconds of the name being displayed. If the user was unable to make a gesture that conformed to the segmentation requirements, then this was counted as incorrect segmentation. A gesture that conformed to the requirements was said to be correctly segmented. The segmentation accuracies are shown in Table 7.

Table 7 - Real-time segmentation accuracies

Segmentation			
Subject	**Ex. 1**	**Ex. 2**	**Ex. 3**
1	98.3%	97.3%	98.7%
2	100.0%	99.3%	98.6%
3	98.3%	100.0%	100.0%
4	100.0%	100.0%	100.0%
5	100.0%	97.3%	-

The weighted average segmentation accuracy during the real-time experiments was 99.0%. The weighted segmentation accuracy within the recorded dataset was 94.8%. It is possible that discrepancy is caused by mental fatigue due to the extended nature of the data collection.

Classification accuracy was established according to whether the subject's gesture matched the required gesture. Incorrectly segmented gestures, or gestures after the first within the time period were not included in the classification accuracy. The classification accuracies are shown in Table 8.

The weighted average classification accuracy across these tests was 66.9%. Within the datasets, the weighted accuracy for these experiments was 89.7%. It is not uncommon for performance to decrease when testing is occurring within a real-time setting. The additional pressure of performing the gestures when instructed reportedly caused subjects to contract their muscles before performing the gesture in several cases. This contraction was taken as the gesture, and any subsequent gestures the subject performed were ignored. An additional source of error may be the misalignment of the gestures. The exact position of the features within the dataset is dependent on the point at which the amplitude of the

Table 8 - Real time Classification accuracies using template-based classification

Classification			
Subject	**Ex. 1**	**Ex. 2**	**Ex. 3**
1	93.2%	74.0%	74.3%
2	85.0%	61.0%	52.7%
3	91.5%	58.0%	64.7%
4	98.3%	79.3%	78.6%
5	95.0%	28.0%	-

signal exceeded the threshold. While this is normally consistent between similar gesture instances, a uniquely low or high amplitude gesture may significantly displace the gesture within the dataset. Finally, in the offline analysis, the training and test data are selected randomly. This means that both sets contain gestures from all periods of the collection phase. Conversely, with real-time testing, all training data was taken in the first part of the experiment, and all test data was taken later in the experiment. As a result, factors that may have an affect over time, such as muscle fatigue, may be present in the test set, but not in the training.

The classification methodology offered here provides an indication that the complex interaction of the mechanical signals within the muscles of the forearm leads to repeatable patterns of mechanical vibration on the surface of the skin.

5.4.2 Machine Learning-based Classification

The template-based classifier described in the previous section works by generating a model that is representative of the training data, and then predicting the class of the test data based on this model. The classifier cannot make predictions without examples of the data, and the model becomes more representative of the gestures as data is added. Since it exhibits these two behaviours, the classifier described above can be thought of as a rudimentary machine learning algorithm. There are several other algorithms that have been shown to be successful in pattern recognition for differentiating gestures observed through alternative methodologies, so in this section a number of these are implemented to improve the classification accuracy.

The algorithms described in this section are implemented using the Classification Learner App in the Statistics and Machine Learning Toolbox for MATLAB. These algorithms were used to classify the same data as the template-based classifier, allowing the methods to be compared.

5.4.2.1 Algorithms

Based on existing methodologies in both EMG and MMG activity classification, four types of machine learning algorithms were implemented and tested during this work. For the initial classification, every data point within the signals was taken to be a feature for classification. As a result, the data from the six channels was sequentially combined to produce a single 1206 element vector of features from the 6x201 array describing the signal from the six sensor elements. The four algorithms chosen for this work were

Table 9 - Confusion matrix of classification accuracies using SVM

		Gesture Classified											
	ID	1	2	3	4	5	6	7	8	9	10	11	12
Actual Gesture	1	95.29%	0.70%	0.18%	0.16%	0.17%	0.24%	0.40%	0.33%	1.16%	0.88%	0.42%	0.06%
	2	0.34%	94.91%	0.63%	0.12%	0.39%	1.04%	0.43%	0.52%	0.11%	0.35%	0.66%	0.51%
	3	0.01%	0.03%	95.11%	1.42%	1.02%	0.06%	0.32%	0.87%	0.13%	0.18%	0.07%	0.78%
	4	0.21%	0.02%	1.85%	94.33%	0.77%	0.53%	0.74%	0.36%	0.00%	0.78%	0.20%	0.22%
	5	0.17%	0.15%	1.04%	0.89%	93.80%	0.64%	0.42%	0.19%	0.00%	0.00%	0.21%	2.47%
	6	0.13%	0.08%	0.03%	0.40%	1.86%	95.69%	0.88%	0.56%	0.00%	0.01%	0.17%	0.18%
	7	0.05%	0.59%	0.65%	0.72%	0.77%	1.07%	92.49%	1.00%	0.07%	0.24%	1.03%	1.32%
	8	0.00%	0.26%	0.16%	0.34%	0.53%	0.17%	1.11%	95.52%	0.41%	0.62%	0.29%	0.60%
	9	0.00%	0.09%	0.98%	0.00%	0.89%	0.07%	0.35%	0.62%	94.56%	0.85%	0.40%	1.19%
	10	0.02%	0.54%	0.45%	0.18%	0.20%	0.00%	0.62%	0.96%	0.59%	95.92%	0.29%	0.23%
	11	0.00%	0.19%	0.31%	0.00%	0.90%	0.00%	0.04%	0.58%	0.09%	0.73%	95.28%	1.89%
	12	0.00%	0.16%	1.10%	0.02%	1.45%	0.07%	0.76%	0.77%	0.56%	0.01%	1.72%	93.38%
										Average Accuracy			94.69%

Table 10 - Confusion matrix of classification accuracy for amputee subject using SVM

		Gesture Classified						
	ID	1	2	3	4	5	6	7
Actual Gesture	1	68.2%	11.4%	2.3%	13.6%	0.0%	4.5%	0.0%
	2	2.1%	79.2%	0.0%	0.0%	12.5%	2.1%	4.2%
	3	2.0%	0.0%	96.0%	0.0%	0.0%	2.0%	0.0%
	4	19.1%	0.0%	0.0%	68.1%	2.1%	6.4%	4.3%
	5	0.0%	6.7%	0.0%	0.0%	80.0%	4.4%	8.9%
	6	0.0%	22.4%	0.0%	0.0%	6.1%	53.1%	18.4%
	7	0.0%	12.5%	0.0%	0.0%	29.2%	14.6%	43.8%
					Average Accuracy			69.79%

K-Nearest Neighbours (KNN), Decision Trees (DT), Linear Discriminant Analysis (LDA) and Support Vector Machines (SVM). These algorithms are described in detail in Appendix I.

5.4.2.2 Experimental Comparison

Each of the four algorithms listed above were used to classify the gestures from the gesture database. The classification used k-fold validation to reduce the likelihood of overfitting, as well as minimize the effects of mislabelled data. K-fold validation works by splitting the data into k subsets, and using k-1 to train the

Table 11 - Comparison of different machine learning techniques on gesture groups for non-amputees (amputee)

	DT	KNN	LDA	SVM
G1&G2	97.17% (81.5%)	98.45% (96.7%)	96.75% (81.5%)	99.08% (88.0%)
G1,G2,G3,G4&G6	90.31% (71.0%)	96.98% (90.8%)	79.03% (56.7%)	97.44% (80.3%)
G8,G9,G10,G11&G12	86.38%	95.14%	77.82%	96.43%
All Gestures	71.83%	93.65%	57.48%	94.69%

classifier, with the final set providing the test data. This is repeated so that the classifier is tested on all folds of data. A k value of 5 was used for these experiments.

As with the previous experiment, classifiers were trained and tested independently for each individual. This was also repeated 100 times to ensure that the accuracies were representative of the performance of the algorithm, and not a feature of any specific initial conditions.

Overall, the SVM performed best across all gestures with an accuracy of 94.69%, while the LDA performed the worst with an accuracy of 57.48%. The KNN achieved an accuracy of 93.65% and the Decision Tree achieved an accuracy of 71.83%. A confusion matrix generated by combining the SVM classification accuracies for each non-amputee subject is presented in Table 9. The confusion matrix generated from the SVM classification data of the amputee is presented in Table 10.

Additionally, the three sub-groups used for real time testing of the template recognition were tested with the four classifiers to show how problem complexity affected classification accuracy. A summary of these results is presented in Table 11.

The accuracy of the KNN and SVM were comparable, and the difference between the two is negligible for a dataset of this size, although the KNN performed significantly better for the data from the amputee. This is likely because the amputee subject was not able to generate the same amount of data as the other subjects. In each case, the LDA performed least well. On reflection, this may be due to the high dimensionality of the data, which is known to negatively affect the performance of LDAs. As a result, methods of dimensionality reduction were employed to attempt to improve the performance.

5.4.2.3 Dimensionality Reduction

Two methods to reduce the dimensionality of the data were implemented here.

Principal Component Analysis

Principal Component Analysis (PCA) is a method of reducing the number of features describing a problem to a smaller set that still contains most of the information of the larger set. It has the potential to be useful in this problem, because the raw data being classified is sequential time domain data. Since the sample frequency of the sensors are significantly higher than the primary frequency of the MMG signal, this will lead to a large amount of dependence between different features. Additionally, since an MMG signal can

be detected at different points around the arm, there is also a potential for two adjacent sensors to detect some of the same information, leading to further dependency. PCA uses these dependent features to create a new set of independent features that contains most of the same information. Exactly how much information to retain is specified by the user and is a method of tuning the PCA algorithm.

PCA is similar to the first step of an LDA, with the difference being that while the dimensionality reduction within an LDA is a supervised process, PCA is unsupervised, meaning that it occurs with no reference to the labels on the data. Rather than attempting to maximize separation between groups, PCA aims to maximize the variance of the entire data set. It then creates a new feature which the points in the dataset can be mapped to, which expresses the largest possible variance, known as the principal component. Perpendicular variables can then be constructed that describe the remaining variance, ordered by the amount of variance they describe. As these components are generated, the variance in the remaining dimensions decreases. All dimensions are required to explain 100% of the variance, however 99% of the variance can often be explained by significantly less, particularly in datasets such as this.

The effect of performing the PCA was quantified by training each of the four classifiers on the components generated through the PCA. The configuration of each of the classifiers was the same, and as before, the accuracies were generated by training each classifier one hundred times for each person, averaging across all values to generate an overall accuracy.

Two configurations of PCA were used. The first attempted to create enough features to explain 95% of the variance of the data. The specific output of the PCA was dependent on the input data, but for one subject, 23 components were enough to explain 95% of the variance. The variance explained per component in this example started at 38.7% for the principal component and reduced to 2.1% by the tenth component. The first ten components explained ~84% of the variance, while the remaining thirteen were required to explain the remaining 11%. The second configuration attempted to explain 99% of the variance. For the example subject discussed above, explaining the extra 4% of variance required an additional 19 components. The resulting accuracies of both these configurations on all classifiers are presented in Table 12, along with the original accuracies when PCA was not performed.

Table 12 - Effect of PCA on classification

	DT	KNN	LDA	SVM
PCA (95%)	71.64%	91.83%	93.06%	93.48%
PCA (99%)	71.50%	87.11%	93.61%	93.88%
No PCA	71.83%	93.65%	57.48%	94.69%

It can be observed that since some information is lost, the accuracy of the algorithms that make no assumptions about the data decreases, whereas the accuracy of the LDA is markedly improved.

Feature Extraction

The second method of reducing the dimensionality of the problem is to generate features based on the specific form of the data. This is technique is widely documented in classifying EMG signals, so potentially could be of use in this problem. Feature extraction can offer significant benefits over simply using raw data. The primary benefit is that large datasets can be compressed into small feature spaces by individually extracting the important features. This will be beneficial to classifiers that rely on maximum likelihood rules such as LDAs. While PCA achieves this as well, it is a blind process, meaning the compressed data does not necessarily retain the separability of the input data. By selecting features that have previously been shown to allow different classes of similar data to be distinguished, this information may be retained. It also means that the features that are chosen are less likely to be coincidental due to the small sample size. The set of features used were taken from similar papers classifying EMG gestures, and were as follows:

Table 13 – Accuracies of classification using extracted features

	DT	KNN	LDA	SVM
Features	71.34%	89.90%	94.51%	93.33%

Table 14 - Summary of best performing machine learning methods for classification

Algorithm	PCA (Y/N) (Variance Explained %)	Features (F) or Raw Data (RD)	Accuracy
LDA	Y (95%)	RD	93.06%
SVM	N	F	93.33%
SVM	Y (95%)	RD	93.48%
LDA	Y (99%)	RD	93.61%
KNN	N	RD	93.65%
SVM	Y (99%)	RD	93.88%
LDA	N	F	94.51%
SVM	N	RD	94.69%

- Root Mean Square (RMS)
- Integral of Absolute Value (IAV)
- Mean Absolute Value (MAV)
- Modified Mean Absolute Value 1 (MAV1)
- Modified Mean Absolute Value 2 (MAV2)
- Simple Square Integral (SSI)
- Variance (VAR)
- Absolute Value of the 3rd, 4th and 5th Temporal Moment (TM3, TM4, TM5)

- Difference Absolute Mean Value (DAMV)
- Difference Absolute Standard Deviation Value (DASDV)

Each set of features was extracted for each channel for the dataset. For the twelve features listed above, this resulted in a total of 72 features to use for classification. The resulting accuracies using these features for all classifiers is shown in Table 13.

5.4.2.4 Comparison of Algorithms for Real-time Implementation

For a dataset of this size, it is not possible to claim that one algorithm will consistently outperform the others when new data is introduced. This examination can provide an estimation for the expected accuracy of these algorithms on this type of problem, but it cannot provide a definitive description as to will achieve the highest accuracy across all gestures for all users. Eight algorithms achieved accuracies that were within 1.6% of the top accuracy achieved, these eight can be considered for further examination. Table 14 summarizes these eight algorithms.

The top two of these were considered from real-time implementation. These were the SVM classifying the full gesture data, and the LDA classifying the features.

For real-time implementation, two assumptions can be made based on the expected use case. These assumptions are as follows:

- The target device may have limited processing power.
- The classifier may need to be retrained regularly.

Based on these assumptions, there are three addition relevant metrics to compare the two algorithms:

- Time taken to classify an observation.
- Time taken to train the classifier with X samples.
- The number of samples needed to achieve accurate classification (defined as a classification rate of within 2% of the final accuracy)

The first two of these are processor dependent, however a comparative examination can be performed, and the results are in Table 15.

The number of samples (N) required to classify data to within a threshold percentage of the final accuracy was derived experimentally, ranging from 5 to 80 in increments of 5.

For each individual, two sets were extracted. The first set, referred to as the training set (TR), contained N samples of each gesture. The second set was the test set (TE) and contained 20 samples of each gesture. TR and TE were created so as to have no common data and using a method that ensured that the instances for each set were chosen from the pool in a non-repeatable approach. TR was used to train the classifier and provided an estimation for the accuracy of the classifier using 5-fold validation. TE was then used to

Table 15 - Relevant metrics for classifier comparison

	Number of observations classified per second	Time taken to train classifier(s)
SVM	~420	39.659
LDA	~154[a]	1.1881

[a] It should be noted that the classifier itself can perform ~14,000 classifications per second, however the time taken to extract the features from all six channels was also factored into this calculation, significantly increasing the time for each classification.

test the classifier, providing the final accuracy on unseen data. This was repeated for each individual, and the accuracies were combined. For each value of N, this process was repeated a total of 10 times, providing a range of different training and test sets for each individual. The overall accuracy for each value of N was computed by averaging the accuracies for all individuals during all repetitions. Using this method, the graph in Figure 25 was constructed.

Based on this data, the number of samples of each gesture required to achieve within 2% of the final accuracy is ~40. It can be noted from Figure 25 that after approximately 15 instances of each gesture, the two algorithms achieve similar accuracies. As a result, the comparison between the two algorithms is summarised in Table 16.

Both algorithms would make good choices for real time implementation based on the comparison presented here. The only place where they diverge meaningfully is in the training time, where the SVM takes significantly longer to create a trained model. Since the two algorithms are being compared based on these metrics, it can be concluded that the LDA performed on features extracted from the data is the better choice, and so this was chosen as a benchmark for real-time applications.

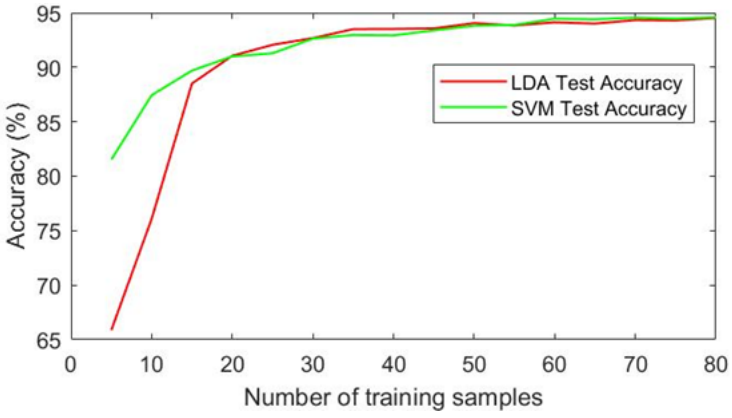

Figure 25 - How number of training samples affects accuracy

Table 16 – Summary of top performing classifier metrics

	Accuracy (5-Fold, Full dataset)	Number of observations classified per second	Time taken to train classifier (s)	Number of samples required for training
SVM	94.64%	~420	39.659	40
LDA	94.51%	~154	1.1881	40

5.4.2.5 Real-time Implementation

In order to implement this classifier in real-time, the C# environment described in Chapter 3 was adapted to make use of the Classification toolbox from MATLAB. This had the advantage of maintaining consistence between the offline and real-time implementation of the LDA without introducing a noticeable delay. In order to achieve this, the code required to construct the classifier was exported from the ClassificationLearnerApp. This code was then adapted to allow the number of classes and features to be configured by the C# environment based on its current requirements.

As with the previous real-time implementation, data was segmented and extracted into a storage buffer prior to classification. Once the buffer was full, the thirteen features described in the previous section were extracted from each of the six channels and compiled into a feature vector. During the training phase, the data label was appended to this feature vector, which was added to a training set array. At this point, the training set was passed to the classifier, and a validation accuracy was returned. When the program was closed, the trained classifier was saved to allow future classifications to be performed without the need to retrain the system.

5.4.2.6 Real-time Accuracy

The protocol for the assessment of the real-time LDA classification was consistent with the other experiments described in this chapter. Three healthy individuals (2 male, 1 female, average age: 47.3yrs SD: 14.5yrs) who gave their written informed consent participated in this assessment, and the sensor placement was kept consistent to previous data collection periods. The assessment was broken down into three experiments, and the gestures used were the same as in the real-time template-based classification experiment.

Each subject was seated to that they were able to comfortably keep their arm in both the upright and hanging position. The experiment was then discussed, and subjects were given the opportunity to familiarize themselves with both the protocol and the equipment. Subjects were then instructed to make forty instances of each of the gestures used in the experiment. These were used to train the classifier.

Once the training was complete, the classifier was then tested. This was performed by asking subjects to repeat gestures based on a visual prompt.

The segmentation was assessed in the same way as with the previous experiment and had comparable results.

Table 17 - Segmentation accuracy for real time machine learning implementation

Segmentation			
Subject	**Ex. 1**	**Ex. 2**	**Ex. 3**
1	100.0%	100.0%	100.0%
2	100.0%	100.0%	100.0%
3	100.0%	100.0%	100.0%

The classification provided by the LDA was evaluated against the prompt, and the classification accuracy was determined. As before, any gestures made after the first within the cool down period were ignored.

Table 18 - Classification accuracy using LDA in real time application

Classification			
Subject	**Ex. 1**	**Ex. 2**	**Ex. 3**
1	100.0%	96.0%	100.0%
2	88.0%	88.0%	80.0%
3	96.0%	84.0%	96.7%

From these results, it was concluded that the Linear Discriminant Analysis classifier is suitable for real-time classification of gestures based on MMG signals. The steps for data acquisition to classification can be expressed as shown in Figure 26.

5.5 Guidelines for Practical Use of MMGs

The gesture database and the techniques described in the previous section allow important guidelines for the use of a system such as the one described here to be derived.

5.5.1 Ideal Number of MMGs

Using the classification algorithm described in the previous section, it is possible to evaluate the effects of the number of MMGs on classification accuracy. To achieve this, data from each subset of the MMGs was considered when training the classifier. For 6 MMGs, there are 73 possible combinations that make use of at least one MMG. The classifier was trained 100 times on the data from each combination, and the resulting accuracies were averaged to find the accuracy for each combination and for each person. These accuracies were then grouped by the number of MMGs they describe, and the average and standard deviation was calculated to describe the effect of introducing the additional MMGs. The graph describing these results is shown in Figure 27.

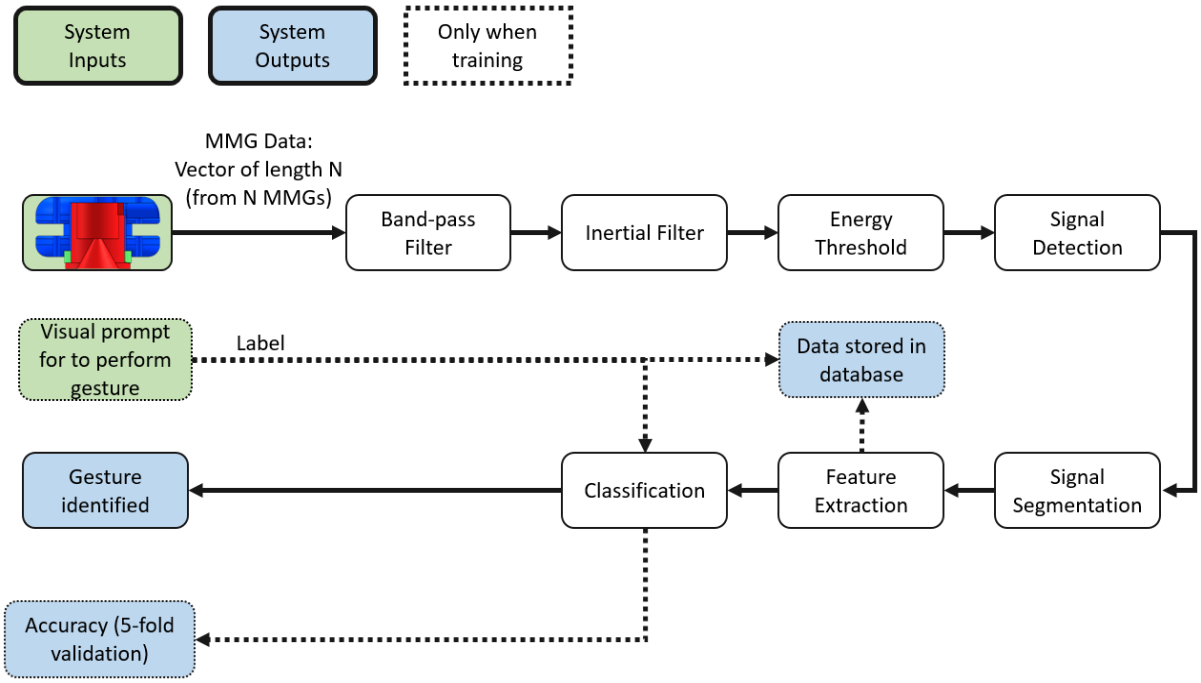

Figure 26 – Flowchart showing steps from data acquisition to gesture identification

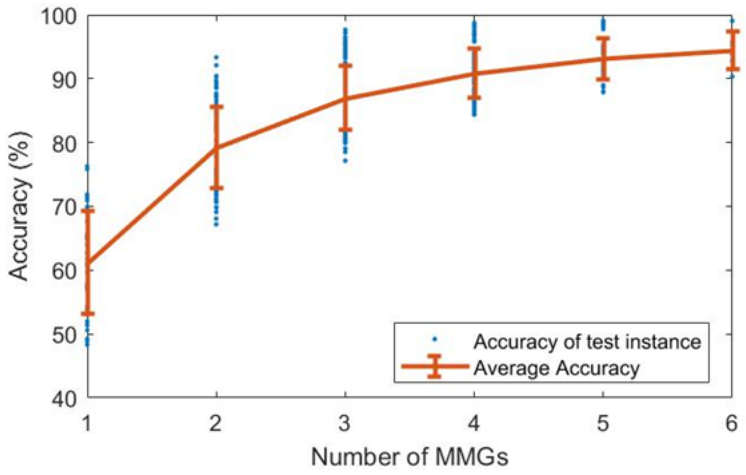

Figure 27 - Effect of number of MMGs on classification accuracy

This demonstrates that the accuracy that the system can achieve is dependent on the number of sensors in the system. The improvement that can be achieved by including additional sensors decreases for each one that is added. A single sensor achieved a mean accuracy of 61.20%, and the addition of a second

sensor provided an 18.02% improvement. The difference between the fifth sensor (93.11%) and sixth sensor (94.48%) was only 1.37%.

5.6 Chapter Summary

This chapter presented a detailed analysis of the MMG signals recorded from a six-element sensor placed on the forearm when performing a range of gestures. A database was created containing many recordings taken from seven subjects while performing twelve gestures. These recordings were labelled with their class within the dataset, so that segmentation could be simulated. A method of gesture segmentation based on thresholding was tested and found to provide successful gesture segmentation. This segmentation was then used to extract the individual gestures from their data files to allow several classifiers to be tested.

Classification methods included a template matching method. This method had several benefits; it was simple to implement and provided a visual representation of the trained classifier for each class. This method examined the consistency of the signals and concluded that similar gestures produced visually similar signal patterns, however the accuracy of this classifier was suboptimal for real-time applications at 66.7%. This indicates that while the signals are repeatable, there is some variation within individual subjects' datasets, and therefore this method of classification is not robust to the variance of human subjects.

Machine learning methods have the potential to provide a classifier that is more robust to intra-class variance. To explore this, several machine learning methods that have previously been utilised for EMG-based gesture recognition were examined, including K-nearest neighbour, decision trees, support vector machines and linear discriminant analysis. It was found that for raw datasets, support vector machines outperformed the other classifiers, potentially due to the high dimensionality of the data. The computational expense of training the classifier necessitated the exploration of other methods. Several dimensionality reduction techniques were explored along with these classifiers, including feature extraction and principal component analysis. The two best classifiers were linear discriminant analysis performed on extracted features, and support vector machines using the raw data from the database, both of which achieved an accuracy of ~95% in the offline analysis of 12 grip patterns.

Real-time considerations, such as training time, set the LDA on extracted features apart from the other classifiers, and so it was implemented for real-time testing. All real-time assessments in this chapter consisted of three experiments, each of which consisted of a training phase and a test phase on a subset of the previously examined gestures. The accuracies of these experiments were weighted by the number of gestures they described and combined to provide and overall accuracy. The overall, real-time accuracy for the template-based classification method was 66.9% for experiments consisting of 2, 5 and 5 gestures. The overall, real-time accuracy for the LDA across the same three gesture groups was 91.43%. The accuracy of this method was far more promising, and therefore machine learning-based methods were used for further experimentation in the following chapter.

Chapter 6

Applications: Prosthetic Control and Robot Teleoperation

6.1 Introduction

To fully validate the sensing system described in this document against its initial design brief, it must be implemented for the applications outlined in Chapter 1. This requires implementation for machine control, including prosthetic device control and robot teleoperation. The interface can be used for control of a ubiquitous computing system, but these also require levels of context awareness and artificial intelligence that is not found in current systems. As a result, it was necessary to design tasks that could plausibly represent the way these devices may be utilised in the future. This allows the utility of the system to be demonstrated in the context of the current system architectures. In some cases, this has required the tests to take place in a virtual environment, where the context can be provided to the system without requiring additional sensing.

6.2 Real-world implementation for Robot Teleoperation

The first implementation of the sensing system described here aimed to both test and demonstrate the full utility of the system. As the field of robotics evolves, it is likely that the role of human machine interfaces will be to provide more high-level control signals than direct manipulation. The experiment was designed around a situation where human expertise was required, but some functions of the robot were automated. This resulted in the creation of a semi-autonomous control system, which took control signals both from the NUIMU orientation estimation and the gesture recognition. A Baxter Robot was used as the target platform, with the Baxter's arms controlled by the orientation of the subject's arms. The subject could then make gestures to trigger several pre-programmed motions, ranging from manipulations of the end effector to complex full arm motions that required a greater level of accuracy than the user was able to provide. Since the Baxter Robot has a larger number of degrees of freedom than the previous manipulators, a larger number of individual NU modules were incorporated into the sensor suite.

6.2.1 Experimental Protocol

Subjects wore a NUIMU on their upper and lower arm segments, with six MMGs on their right forearm. The MMGs were arranged as per the previous experiments, with three located approximately over the extensor digitorum muscle group and three located approximately over the flexor digitorum muscle group. Prior to each experiment, each user recorded twenty instances of each of the required gestures.

Data was sent from the sensor suite to a computer via Bluetooth. The limits of Bluetooth networks required a secondary sensor implementation to be utilized, as described in Chapter 3. MMG gesture recognition and orientation estimation were performed on the computer and sent to a custom node in the Baxter robot's controller via a server located outside the local network. In this implementation, the simplest form of classification based on templates was used in order to test its efficacy in real world experimentation. As a result, there was no technical requirement that the user needed to be within the vicinity of the robot, provided the visual feedback was sufficient to successfully complete the required protocols.

The protocol itself was presented to the subjects as a story-based scenario, in which they had to complete a task requiring several steps. The scenario presented the subject with a large-scale circuit board, and a Baxter acting as a bomb disposal robot. They were given a list of instructions that required the manipulation of interactive components on the board to 'defuse a bomb'. The scenario was presented this way for two reasons: First, the inclusion of a storyline was intended to ensure that the subjects remained engaged with the tasks for the duration of the experiment. Second, this specific scenario was intended to emphasis to the subject that their task was time critical, and therefore encourage them to complete it as quickly as possible.

The movement of the robot was limited to a virtual 'box' within which it could operate. This protected the robot from damage, since it was being used within a crowded environment. A corresponding virtual box was also created in front of the user. When being directly controlled, the end position of the subject's hand inside the box was mapped directly to the robot, which aimed to move to the corresponding position within its constraints. One benefit of this method was that the subjects were able to stop control of the robot if they felt it necessary by removing their hands from the virtual box, without creating large unwanted movements.

A number of the tasks required the subjects to utilize 'tools' from a 'tool rack' located in front of the Baxter. Different tools were required for different tasks, however the tool rack was located outside of virtual area, to make best use of the space. When a tool was required, subjects selected it by performing the corresponding gesture. When the gesture was detected, a pre-programmed pattern of movement was executed, moving the arm out of the virtual constraints, selecting the relevant tool, and returning the arm with the tool to the position dictated by the subject's arm. The orientation of the end effectors was pre-defined and maintained within the robot controller.

The tasks used on the board fall into four categories:

- Unplug tasks – these tasks required the user to remove items from the board. The items used magnets to attach to the board and took the form of cubes with five-centimetre edges. The task required the subject to move the end effector of the robot so that the fingers of the gripper were on either side of the cube, and then trigger the robot to close the gripper. They were then required to pull the cube away from the board, position the end effector over a target box located on the tool rack, and trigger the gripper to open, dropping the cube into the box. If the subject caused the cube to drop elsewhere (either by knocking it from the magnetic contact points with the gripper, or triggering the open gesture too early), the cube was replaced on the magnetic point, and the subject was required to re-do the task.
    - This task required:
        - Two point to point motions
        - Two gestures (Close (close the gripper), Open (open the gripper))
- Switch tasks – these tasks required the subject to move a switch from one pole to the other. Each position of the switch was magnetized to ensure that a complete motion was required to move it. The task therefore required the subject to move the end effector to one side of the switch (one

motion), and then execute a lateral movement to physically actuate the switch (second motion). This is classified as two motions, since the first is a large point to point movement, whereas the second requires the subject to maintain motion along a specific path.
    - This task required:
        - Two point to point motions (One to approach the switch, and one to actuate it)
- Test Pad tasks – these tasks required the subject to simulate checking the voltage on the circuit board. To achieve this task, the subject had to retrieve the Voltmeter Tool from the tool rack, and either touch a defined pad, or two defined pads simultaneously depending on the protocol. When making contact with two pads simultaneously, the subjects were asked to ensure that as little time as possible was spent in contact with the pad, encouraging them to attempt to position both arms simultaneously as opposed to sequentially. Once contact had been achieved, the user was required to return the tool to the tool rack.
    - This task required:
        - One (or two simultaneous) point to point motions
        - Two gestures (Point (retrieve voltmeter tool), Open (replace the current tool))
- Key tasks – these tasks required the subject to retrieve a key tool from the tool rack, place it into a keyhole on the board, and rotate the wrist in the direction defined in the instructions. They were required to maintain the position of the key during rotation, and to return it to the tool rack once the required rotation had been completed.
    - This task required:
        - One point to point motion
        - Three gestures (Key grip (retrieve the key tool), Rotating the wrist (both clockwise and anticlockwise motions were required), Open (replace the current tool))

The experiment was split into two protocols and performed twice. The first recorded motion from the subject's left arm, while their right was solely responsible for creating the control gestures. This was to ensure that the gestures were as free from motion-induced artefacts as possible. As a result, six gestures were used, two to select tools, and four to manipulate the end effectors. The second protocol provided the user with simultaneous control of both arms and used three gestures for tool selection and end effector manipulation. Both protocols were conducted once with the NU Interface, and once with an Xbox controller.

An interface for this application was designed around an Xbox controller, to provide an indication of the task difficulty against which to assess the NU Interface. This control interface used the buttons of the interface to provide categorical data similar to the gestures and used a reference position in 3D space to provide the positional data. The joysticks on the controller were used to supply the Up/Down and Left/Right motions (referenced to the Baxter), and the triggers were used to provide the Forward/Back motions.

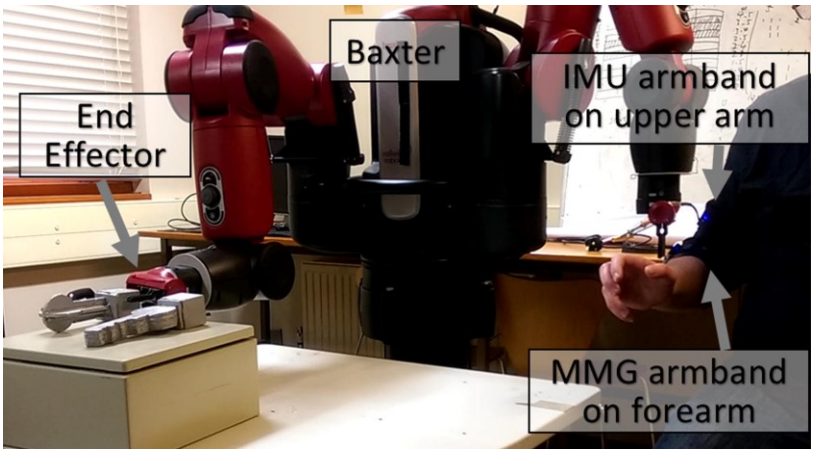

Figure 28 - Experimental Hardware

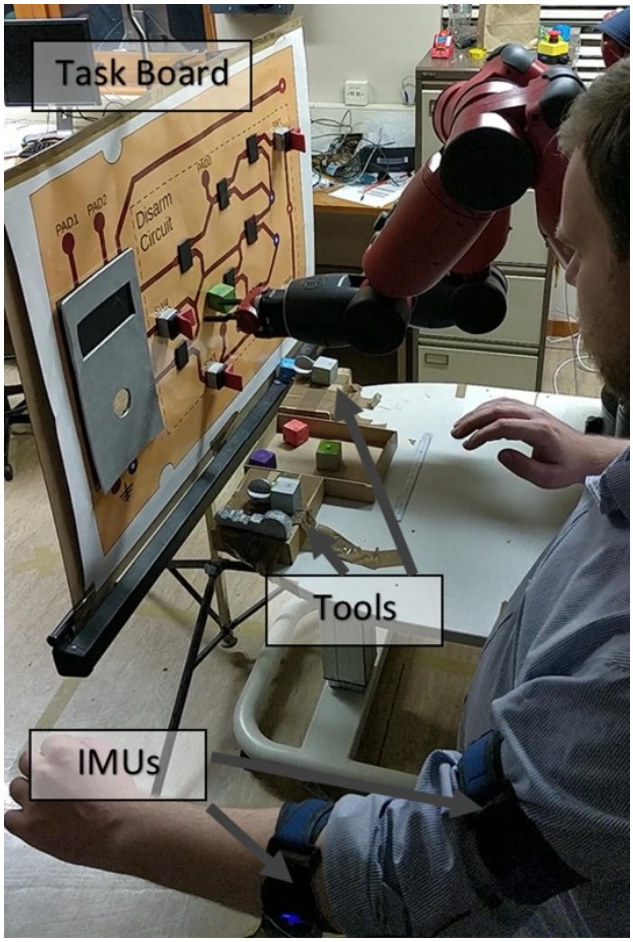

Figure 29 - Experimental setup

6.2.2 Results

The first protocol used one arm to generate positional data and six distinct gestures. The protocol consisted of; four unplug tasks, two key tasks, one (single arm) test pad task and two switch tasks. The second protocol consisted of; four unplug tasks, three (dual arm) test pad tasks and four switch tasks. Participants completed these tasks first using the controller, and then using the NU sensor suite. The time taken to complete both protocols using the NU sensor suite was longer than the corresponding time taken with the controller but did not require specific dexterous manipulation of the fingers. The average time taken is presented in Table 19.

Table 19 – Average accuracies of Baxter robot control

Time(minutes : seconds)			
Protocol 1		Protocol 2	
Controllers	NU Sensor Suite	Controllers	NU Sensor Suite
3:55	7:02	5:02	6:40

6.2.3 Discussion

When comparing the times taken to complete the experiment using the controllers against using the NU sensor suite, there is a noticeable difference between the two protocols. A possible explanation for this can be seen in the ratio of point to point movements and gestures. Protocol one required the subject to perform sixteen gestures (of six categories), and fifteen point to point motions, whereas Protocol two required twelve gestures (of three categories), and nineteen point to point motions. These averages can be used to generate difficulty ratings for these two tasks, by assuming that the speed of completion is proportional to the difficulty of each task according to:

$$T = p * ID_1 + g * ID_2 + n \quad (60)$$

Where T is the total time, p and g are the number of pointing tasks and gestures required respectively, and ID_1 and ID_2 are the index of difficulty for each task. n describes additional time spent by the user not actively trying to achieve a task, and therefore may not be consistent between tasks.

Assuming $n = 0$, it can be said that:

Table 20 – Index of difficulties for Baxter tasks

	$\boldsymbol{ID_1}$ $\boldsymbol{(pointing\ tasks)}$	$\boldsymbol{ID_2}$ $\boldsymbol{(gesture\ tasks)}$
Controllers	16.2	-0.5
NU Sensor Suite	10.8	16.3

While it is likely that ID_2 will be ≈ 0 while using the controllers, it cannot be negative and therefore $n \neq 0$, however the results appear to correlate to observations made during the experiment.

It can be seen that the time taken to make a correctly classified gesture was significantly longer than the time taken to press a button on the controller. This may be due to both the extended time needed to make a large gesture when compared with the small movement required to press a button, and the inaccuracies introduced in the classification of the gestures by recording at different arm orientations. It is possible that some of these errors in classification could have been avoided by increasing the quantity of training data. By contrast, when using the NU Sensor Suite, the point to point-based movements were completed in approximately 60% of the time when compared to the time taken when using the controllers.

In addition, the errors introduced through the use of a generic model of the arm when calculating the forward kinematics were not reported as noticeable by subjects when asked. This may be because the subjects were making movement based on visual feedback, provided by the robot motion, and not based on their proprioceptive sense.

In order to further examine factors that may have led to a lower than optimum classification accuracy, it was decided to continue this experimentation in virtual environments where hardware specific issues could be removed. Several experiments were conducted to identify potential factors and design compensation schemes. These are outlined in the remainder of the chapter.

6.3 Virtual Reality Environments

The desired virtual reality (VR) environment was created using the Unity game engine, since this offered inbuilt VR functionality and could run custom scripts. Each environment created within a unity project is known as a scene, and every scene included a C# script, which acted as a scene controller. The controller had two functions by default, one named Start which is called first and can be used to set up the environment, and one named Update, which is called every time the scene refreshes. Each scene was designed according to the specifications of each experiment, so as to provide subjects with the most appropriate feedback.

The VR environment was created as an executable software and required data from the NUIMUs in order to animate the movements of the user and the objects they were interacting with. While it would have been possible to implement the VR environment and NUIMU manager in the same application, it was decided that maintaining the VR executable as a separate application would provide forward compatibility for any changes in the NUIMU protocol. The chosen method of facilitating this inter-program communication was to use Pipes, an inbuilt function within Windows which can be accessed from both programs, and which can store data written from one program until it is read by the other. The Pipe was initialized in the C# host software, referred to as the pipe server, and the VR environment completed the connection, and is referred to as the pipe client. The pipe client created a dedicated thread to monitor this section of stored memory, and to change the corresponding Unity variable each time new data was

available. This variable was checked at the beginning of every Update function, and the objects within the scene were moved according to this new data.

The data are written to the pipe as bytes, regardless of their original type. As a result, packets were created to ensure both that the data was used in the correct way, and that it did not become desynchronized. The data format was:

$$0xDD \quad 0xA2 \quad Mode \quad d1 \quad \dots \quad d_N$$

The format and length of the data were dependent on the $Mode$ value and were usually application specific. A packet containing a single quaternion for example would have a length $N = 16$, since each of the four elements were saved as a 32-bit floating point number.

Three types of data were transmitted using this system for the following applications:

- Orientation – transmitted in the form of quaternions, the orientations could be applied to several elements in the scene. Each quaternion originally consisted of four floating-point numbers, each of which required four bytes to transmit.
- Gesture – transmitted as an integer, the presence of a gesture could trigger pre-programmed activities in the scene to occur. Since the total number of gestures in any one scene was less than 255, this could be encoded with a single byte.
- Position – transmitted as a three-dimensional vector, the position of objects within the scene could be used to place objects at known positions. Position was originally saved in a three-element vector of floating-point numbers, therefore this was encoded as twelve bytes.

In applications where the subject was wearing multiple NU sensors, the data most commonly required for the Unity-based visual feedback was the location of the subject's hand. This was calculated in the NU host software, and so simply was calculated using matrix manipulation. Assuming:

$$WorldOrigin \rightarrow Shoulder\ (W2S) = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{61}$$

the default position of the arm is to be parallel to the X-axis so that:

$$UpperarmLength(UL) = \begin{pmatrix} 1 & 0 & 0 & UAL \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{62}$$

$$and$$

$$ForearmLength(FL) = \begin{pmatrix} 1 & 0 & 0 & FAL \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

And that the rotation of the arm segments is converted from quaternions to rotation matrices so that:

$$UpperarmOrientation(M) = \begin{pmatrix} m11 & m12 & m13 & 0 \\ m21 & m22 & m23 & 0 \\ m31 & m32 & m33 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\& \tag{63}$$

$$ForearmOrientation(N) = \begin{pmatrix} n11 & n12 & n13 & 0 \\ n21 & n22 & n23 & 0 \\ n31 & n32 & n33 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Then the position of the hand can be expressed by:

$$WorldOrigin \rightarrow Hand = W2S * M * UL * M^{-1} * N * FL \tag{64}$$

M^{-1} is required since both M and N are world reference rotations.

This can be simplified to give the matrix that describes the end effector:

$$EndEffector(EE) = \begin{pmatrix} n11 & n12 & n13 & x + UAL * m11 + FAL * n11 \\ n21 & n22 & n23 & y + UAL * m21 + FAL * n21 \\ n31 & n32 & n33 & z + UAL * m31 + FAL * n31 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The position can be taken from the fourth column of this matrix ($EE_{41}, EE_{42}, EE_{43}$), and was used for a number of the applications listed below.

The host software was receiving data from the NUIMUs at between 200Hz and 1,000Hz, depending on the nature of the data. The rate at which the Update function is called in unity varies between devices, but in this implementation is approximately 60Hz. To ensure both that the data transfer was as efficient as possible, and that rate of information was independent from the data rate of the NUIMUs, the code was refactored. The Pipe server was moved into its own class, referred to as PipeController. Timer objects were then implemented in this new class. When a new object of the PipeController class was created, it would initialize the pipe server and wait for a client device to connect to it. Once a connection occurred, this class would then begin a timer and wait for a timer event to occur. At this point, it would get the current state of every NUIMU connected, perform the necessary calculations, and then write the relevant data to the pipe. The frequency of these timers was approximately 30Hz, as this gave the appearance of smooth movement in the virtual environment but ensured that data was not written to the pipe more than once between Update functions.

6.4 Hands Free Control within Virtual Environments.

While the output from an individual IMU has been validated both against other sensing technologies and against commonly used estimation algorithms, their use as an interface technology has not. In order to provide this validation, a simple test was designed that placed users in a virtual environment. They were then tasked to move their hand from one designated position to another. Users were able to experience this environment in 3D through the use of a VR headset (an Oculus Rift). This allowed the subjects to observe the three-dimensional position of objects in the virtual environment, and therefore they were able to make the required movements without missing the target due to mistakes of depth perception, which could cause significant problems when performing the tasks using a two-dimensional form of visual feedback.

6.4.1 Participants

Five volunteers (4 male 1 female, average age: 28yrs SD: 2.3yrs) gave their written informed consent and participated in this study, all with no known cognitive or physical impairment. The participants had little experience operating in VR environments and were instructed to remove the headset if any negative side effects of VR were felt (such as motion sickness).

6.4.2 Experimental Protocol

Subjects were instructed to put on a wearable interface consisting of two NUIMUs, one worn on the forearm, and one worn on the upper arm. Both IMUs were calibrated before use and placed in a known orientation around the arm. Subjects were then instructed to put on the VR headset and make themselves comfortable. The position of the VR headset was then recorded, and the environment was assembled around the subject, so that a virtual humanoid model occupied the same position in virtual space as the subject. The position of the subject's shoulder was estimated using a transformation from the VR headset and used in future forward kinematic calculations.

Before the test began, the position of the subject's hand was calculated in virtual space, and a virtual ball ("Trial Ball") was placed there, giving the subject visual feedback as to where the system understood their hand to be. This position was updated in real-time. A second ball ("Target Ball") would appear within the subject's field of view, and they were instructed to bring the Trial Ball mapped to their hand into contact with the Target Ball, marking the beginning of the test. Each time the subject brought their hand into contact with the Target ball, it would instantly move to a new position. The initial position and target position pairs defined the beginning and end of the path that the user should follow and the shortest route between the two points represented the optimum movement. This was repeated fifty times, incorporating movements in a large variety of directions and distances from the shoulder. Subjects were instructed to perform the test as fast as they were able to.

Figure 30 -Using Rift Controller to monitor hand position

Figure 31 - Using IMUS to derive hand position

Each subject was asked to do the test twice, with different sensing modalities providing the position for the Trial Ball in each case. For the first set, subjects held an Oculus Touch Controller in their hand, which was tracked by multiple Oculus Sensors, as seen in Figure 30. These IR cameras have been shown to give

positions in their working environment accurate to within 0.01m. As such, the positions presented to the subject in 3D matched their proprioceptive sense of their hand position. For the second set, the Oculus controller was discarded, and the data from the NUIMUs was used to calculate position of the Trial Ball, according to the measured orientation of the limb segment, as shown in Figure 31.

The position of the Trial and Target ball, the calculated position of the hand and the current time was recorded every time the screen was updated. This allowed the time taken to move from the initial position to the target position to be calculated. It also allowed deviation from a straight path to be calculated, providing a second metric for assessment. For this experiment, the movement time was defined as the time taken between the Trial Ball leaving the radius of the initial position defined by the size of the Target Ball, and entering the radius of the new position, also defined by the Target Ball radius. It was defined this way remove any time taken for initial path planning, and the time taken for the new position of the ball to be observed by the subjects.

6.4.3 Results

With reaction and planning time removed, the average time taken to perform the required movements when using the Oculus controllers was 0.25 seconds. Due to the accuracy of the positions given by the Oculus system, this can be given as the base time taken to complete these movements when the subject's proprioceptive sense and visual feedback are providing complementary information. When the position of the hand was calculated using the NUIMUs, the average time taken to complete the movements was 0.54 seconds. It is hypothesized that this is due to a misalignment between the subject's proprioception, and the visual feedback they are receiving, which requires cognitive engagement to correct.

To examine the deviation from a direct path, the path lengths for each movement were standardized, and the average deviation at each point was calculated. The average deviation when using the Oculus Rift controllers was 0.045m, whereas the average deviation when using the NUIMUs was 0.069 meters. The average path taken using both systems can be seen in Figure 32.

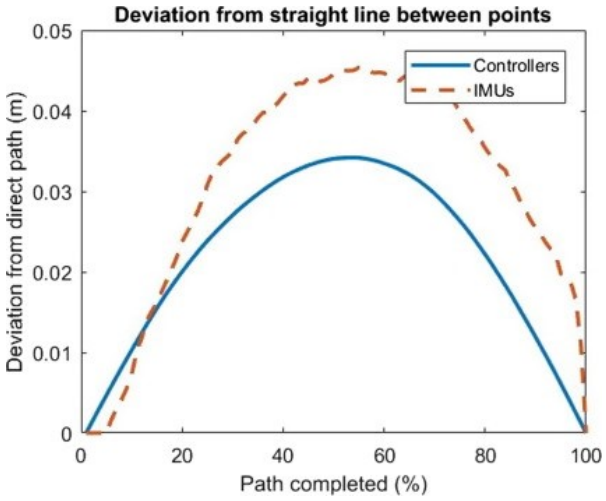

Figure 32 - Average deviation from a direct path

6.4.4 Discussion

The IMUs appeared to introduce 0.29 seconds of delay into the system when performing fast movements. An indication of the cause of this can be found when analysing the path and speed as the subjects progressed through the experiment. When using the Oculus Controllers, the deviation was described by a smooth path with no sharp changes in direction. The peak of the deviation occurs slightly after the halfway point, indicating that some correction is occurring, but it is impossible to say what the basis for this correction is. Conversely, the deviation from the path observed when using the NUIMUs is initially not smooth. These movements are categorized by two sharper changes of direction, one that occurs ~0.2-0.35s seconds into the movement, and one that occurs towards the end of the movement. It can be hypothesized that the initial correction occurs when the subject first detects the misalignment between the motion and the planned path, and the second occurs when the user slows to make fine corrections purely based on visual feedback towards the end of the path. Subjects did report that using the IMUs seemed to require more work than the controllers, although they could not explain why. They reported that they were unable to perceive any of the changes in direction indicating corrections with either system.

One factor that may have contributed to the misalignment between the user's VR position and proprioceptive sense was the approximations made in the forward kinematics. The position of the user's shoulder, and the length of both arm segments were kept consistent with the model, and not specified for each user. Despite this, subjects were able to complete the task, and since more information about the subject's kinematics were known, more realistic representations of the subject were available in the VR environment, for example, the VR 'floating hands' were replaced with arms that mirror the subject's movements.

As a result of this experiment, the use of NUIMUs in VR environments was deemed a suitable test platform for further studies, particularly in situations where environmental factors preclude the use of alternative sensing equipment, such as the Oculus Sensors.

6.5 Prosthetic Control – Classification in Uncontrolled Environments

To facilitate a form of prosthetic control that allows for natural motion to be observed, the MMG classification system must be robust to changes in arm position. Changes to the orientation of the arm will change the muscle activity required for supporting the weight of the hand. It was hypothesised that this along with changes to elbow angle would influence the recorded MMG signal during gesture induced contractions, reducing the classification accuracy. In order to observe the effect of these factors, and well as develop suitable compensation strategies, a new dataset was required.

6.5.1 Participants

The inclusion criteria for this secondary database generation were the same as for the first, and again participants gave their written informed consent. Five healthy subjects (4 male, 1 female, average age: 39.2yrs SD: 15.3yrs) were involved in this work.

6.5.2 Experimental Protocol

For this experiment, the subject wore two armbands, one placed on the upper segment of the right arm and one placed on the lower segment of the same arm. The armband on the upper segment consisted of a single IMU, and the armband on the lower arm segment consisted of one IMU and six MMGs. The armband was placed around the largest radius of the forearm, and oriented so that three MMG were over the flexor digitorum muscle group and three over the extensor digitorum muscle group. The IMUs on both bands were placed at a known orientation of the arm, allowing the kinematics of the arm to be observed.

Subjects were seated at a table and asked to move their arm into a specific orientation. They were then asked to make five instances of each of the five gestures used in this test. The gestures chosen were those used in Ex.2 in the previous real-time gesture experiments. Data collection was performed using the real-time segmentation technique implemented in chapter five, and therefore only gesture data were collected. Subjects then spent thirty seconds in a relaxed position, before moving back to the defined position, and making another five instances of the gesture. This process was repeated until forty instances of each gesture had been recorded. At this point, subjects were given time to rest, before being asked to repeat the experiment with their arm in a different position.

Data were recorded from a total of seven positions. Positions were identified to vary both the background contraction needed to maintain the orientation of the hand, and the shape of the forearm as it was changed by bending the elbow. The positions were as follows:

1. Arm straight, held vertically in an upwards direction
2. Arm straight, 45˚ above horizontal
3. Arm straight, held horizontally in front
4. Arm held parallel to the floor, elbow bent approximately 90 degrees
5. Arm held parallel to the floor, elbow bent to maximum
6. Arm straight, 45˚ below horizontal
7. Arm straight, held vertically in a downwards direction

These positions were chosen at suitable intervals to provide a maximum range of motion without prolonged data collection which may cause fatigue.

Data were labelled with both the gesture that each contraction represented, and the position in which it was recorded. Based on this, several different combinations of gesture and position could be compared. Three classifiers were used for the experiments in this section, an SVM using raw data from all MMG channels, and an LDA trained using features identified in the previous chapter, as well as a cubic SVM (CSVM), which was included due to the increased complexity of the classification due to the additional variables.

6.5.3 Classification – One vs One

To establish an indication of the effect of arm position on classification accuracy, each classifier was trained based on data from a single position. The classifier was then tested on unseen data from each of

Table 21 – LDA accuracy when trained and tested on different positions

		Tested						
		1	2	3	4	5	6	7
Trained	1	66.7%	58.4%	54.5%	41.3%	32.9%	39.8%	42.0%
	2	59.0%	65.2%	60.4%	50.3%	36.1%	44.1%	40.3%
	3	55.3%	63.3%	69.9%	58.1%	46.8%	57.9%	56.4%
	4	52.4%	48.2%	53.5%	72.7%	45.1%	57.8%	61.0%
	5	43.8%	45.3%	47.1%	54.1%	61.8%	48.1%	45.3%
	6	45.2%	49.1%	55.6%	55.6%	40.7%	71.6%	63.7%
	7	46.1%	43.8%	49.7%	54.0%	37.0%	62.9%	70.3%

Table 22 – Linear SVM accuracy when trained and tested on different positions

		Tested						
		1	2	3	4	5	6	7
Trained	1	70.2%	63.7%	54.8%	46.3%	28.0%	38.7%	39.4%
	2	66.6%	70.1%	67.0%	44.6%	29.3%	38.8%	38.0%
	3	65.3%	67.0%	72.6%	56.7%	43.2%	53.4%	54.5%
	4	50.8%	50.8%	66.4%	72.6%	52.9%	58.8%	60.4%
	5	34.9%	36.7%	49.7%	53.0%	69.3%	57.1%	52.6%
	6	35.4%	40.9%	57.5%	55.4%	46.3%	76.6%	70.7%
	7	31.7%	33.5%	51.2%	63.8%	46.6%	70.9%	74.2%

Table 23 – Cubic SVM accuracy when trained and tested on different positions

		Tested						
		1	2	3	4	5	6	7
Trained	1	86.7%	71.9%	67.1%	55.3%	45.2%	57.9%	55.3%
	2	72.9%	85.9%	79.4%	58.1%	42.2%	61.0%	52.2%
	3	67.1%	79.4%	90.4%	60.6%	44.8%	74.3%	62.4%
	4	52.6%	55.3%	61.3%	90.0%	55.1%	68.6%	72.8%
	5	45.0%	46.0%	52.5%	59.2%	88.3%	57.4%	57.1%
	6	53.9%	52.7%	69.0%	71.1%	51.3%	93.4%	75.5%
	7	54.9%	46.1%	53.8%	65.8%	48.4%	74.3%	92.2%

the seven positions. This process was repeated one hundred times for each classifier, and the accuracies attained on the unseen data were averaged. This was repeated so that a classifier was trained using data taken from each position in turn and tested on all. Each classifier was subject specific, and so all accuracies were also averaged across each subject. A summary of the accuracies achieved by the LDA are presented

in Table 21, a summary of the accuracies achieved by the linear SVM are presented in Table 22 and a summary of the accuracies achieved by the cubic SVM are presented in Table 23.

The highest accuracies are achieved when the training set and test set are recorded in the same position, across all three classifiers. The best accuracy achieved by a One vs One classifier when trained on the appropriate positional data was 89.6%, achieved by the Cubic SVM. This would suggest that it is desirable to train a classifier for each position in which it may need to function, however this would be both fatiguing for the subject and time consuming. It is also unlikely to be feasible to train for every position a user may require over the course of normal use. When the correct position is unknown, training on a single position may be necessary. In this case, the average accuracy achieved by the Cubic SVM was 64%. This is lower than desired for real-time use, and therefore the following sections propose several alternative methods.

6.5.4 Classification – Voting

Every classifier tested in the previous experiment achieved a higher accuracy than 40%, and therefore it was predicted that accuracy could be improved by using multiple classifiers. In this implementation, all seven classifiers were used simultaneously, and the most common prediction was taken as the overall system prediction.

The Cubic SVM classifier performed best overall, and so it was used for this test. An example of the classification method is presented in Table 24. The average accuracy achieved by this voting classifier was 77.8%, up from 64% when using a single classifier.

6.5.5 Classification – Many vs. One

In order to establish whether a subset of the positions contains the data required for successful classification, the positions were broken down into sequential groups of three. The middle set was used as the test set, and the classifier was trained on gestures from the positions on either side. The following groups of three were used: When the arm was straight 1-2-3, 2-3-6 and 3-6-7, and as the arm bent 3-4-5. The quantity of data used to train the classifiers was also varied. 40% of the available data represents the

Table 24 – Example of performance of voting classifier

Data Class	Classifier Output							Predicted Class
	1	2	3	4	5	6	7	
0	0	0	4	0	0	0	0	0
0	0	0	1	0	2	2	0	0
1	1	1	1	1	1	1	1	1
3	4	4	4	4	4	2	3	4
3	3	3	3	3	0	3	3	3
4	4	4	2	1	1	2	0	1
0	0	0	1	0	2	2	0	0
1	1	1	1	1	1	1	1	1
2	2	2	2	0	1	3	0	2

same number of instances of gestures as were used to train the classifiers in the previous section, and this was increased to 80%. The LDA, SVM and Cubic SVM were all used in the experiment to determine which of the three was most able to compensate using available data when the correct positional training data were unavailable. The results from this test are presented in Table 25.

Table 25 – Accuracies of classifiers trained on adjacent positions

Group	Classifier	Percentage of available training data used			Average of Adjacent	Average of Correct
		40.0%	60.0%	80.0%		
1-2-3	LDA	62.0%	67.1%	68.8%	60.8%	65.2%
	SVM	66.3%	67.1%	67.7%	65.3%	70.1%
	CSVM	80.3%	82.2%	83.4%	75.7%	85.9%
2-3-6	LDA	62.5%	67.8%	69.5%	58.0%	69.9%
	SVM	70.7%	71.7%	72.3%	62.2%	72.6%
	CSVM	80.4%	82.1%	83.1%	74.2%	90.4%
3-6-7	LDA	62.6%	67.9%	69.8%	60.4%	71.6%
	SVM	71.9%	72.2%	72.2%	62.2%	76.6%
	CSVM	81.8%	83.8%	84.6%	74.3%	93.4%
3-4-5	LDA	57.5%	62.6%	64.6%	56.1%	72.7%
	SVM	64.4%	64.6%	64.5%	54.9%	72.6%
	CSVM	67.4%	69.0%	69.6%	59.9%	90.0%

These classifiers achieved a higher accuracy when trained on data from multiple positions than when classification was performed using data from a single adjacent position. This indicated that there is consistency between the signals being obtained from each position.

Table 26 – Classifier accuracy when trained on representative dataset subgroups

Training Group	Average Accuracy
1-2-3-4-5-6-7	84.3%
1-3-5-7	79.5%
2-4-6	74.9%
2-3-4-6	77.0%

To test this further, a CSVM classifier was trained one data from several groups of gestures and tested against all seven positions. As with all other experiments in this document, no data points appeared in both the training set and the test set. It was decided to use fifty gestures from each position included in the training set, as this was deemed practical should this method be implemented in future real-time tests. The groups and resulting accuracies are presented in Table 26.

Training a single classifier with every position appeared to produce the highest accuracy, although the accuracy falls 5% short of the 89.6% achieved by the appropriate classifier.

6.5.6 Fused Mechanomyography and Inertial Measurement

The previous experiment has shown that the MMG signal observed during a gesture is partially dependent on the orientation of the arm and the elbow angle. From the results shown in Table 23 and Table 25, it appears that the factors affect the signal in a proportional manner, since adjacent positions provide better classifications than more distant positions. It also appears that distant positions may provide slightly contradictory information for the same gesture, since Table 26 shows that a dataset comprising of gestures taken from every position is less accurate than a classifier trained solely on data taken from the test position.

In order to achieve the highest possible level of accuracy, a new method of data fusion is introduced. The MMG signal is monitored for increased activity and low gyroscopic movement that has been shown to be associated with a volitional gesture. When this is detected, data segmentation occurs in the same manner as described in Chapter 5, and the forward kinematics of the limb are calculated. These kinematics do not require accurate limb lengths since they are only used to provide labels, therefore arbitrary values were used. When the classifier is being trained, MMG data are stored, along with the end position of the limb and the label for the gesture.

During testing, the detection of a gesture triggers several addition steps before classification. First, the current position of the end effector is calculated, and considered to be a point in 3D space. A variation of the K-Nearest-Neighbour algorithm is used to identify gestures in the training data that were observed in close proximity to the current test gesture. The subset of training data constructed using this method are then used to train a Cubic SVM specifically for the test instance, which is then classified. Although this is a computationally intensive method of classification, applications such as prosthetic control do not currently require multiple gestures in quick succession, and therefore training a classifier on a small, representative dataset is a viable option.

This method of data fusion allows for a higher accuracy to be achieved by ensuring that the factors such as arm orientation and elbow angle are consistent between both the training set and the test gesture. This was implemented in real time and tested for a simple two position classification. The training set was constructed first, and then several test gestures were made. The full training set, including end effector position was recorded, as well as a list of the gestures identified by the modified KNN for each test gesture. Figure 33 shows the end effector positions observed for both gestures made with the arm held over the head, and gestures made with the arm relaxed by the subject's side. In this implementation, the subject made forty instances of each gesture in each position, and the k value for the KNN was set to thirty-two. This provided an ideal scenario for offline testing, since the gestures selected by the algorithm were only taken from positionally relevant points, and therefore an accuracy of 89.56% was achieved. It is not expected that this accuracy will translate into the real-time experiments proposed in the following sections, however it demonstrates a protocol which can be used to achieve the highest possible accuracy.

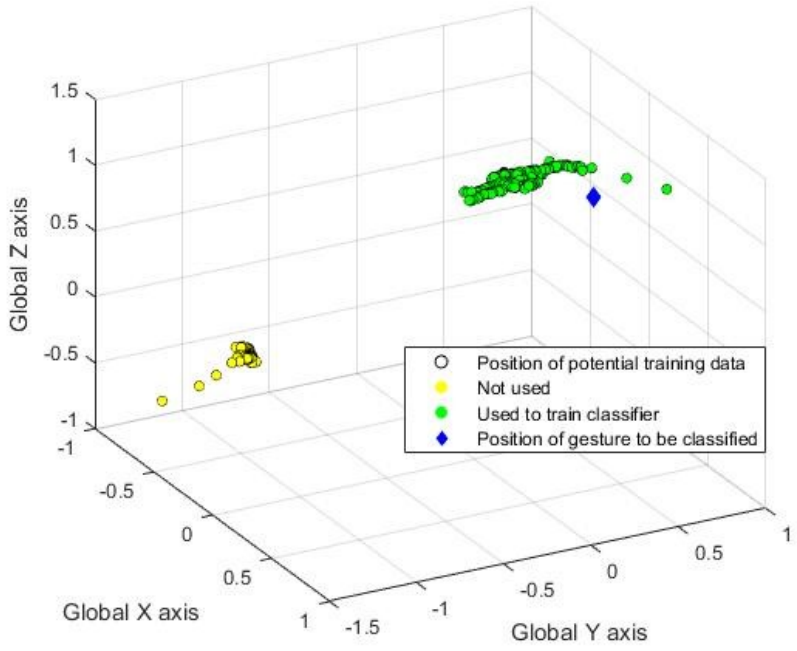

Figure 33 – Positions of gestures created in two arm configurations

6.5.7 Conclusion

Control of prosthetic devices must be robust to changes in orientation, since wearers are often required to move their arms into unnatural poses to compensate for a reduced number of degrees of freedom in their prosthetic. In the past, this has been shown to introduce errors in classification using both MMG and alternative technologies. The experiment in this section has confirmed that these errors can be easily observed in MMG-based classification systems, and a solution for prosthetic control has been proposed and implemented. The NUIMU is able to use orientation information to label each gesture in the training set with a 3D position representative of arm orientation in real time. For more complex implementations where end effector positions may not be representative of arm orientation, elbow angle and forearm orientation could be calculated and used instead. For each gesture to be classified, these positional labels can be used to ensure that appropriate training data can be selected, and a unique classifier can be trained and used. It has been shown that classifiers trained on appropriate data consistently outperform the alternatives. This system is implemented in the following section.

6.6 Robot Teleoperation in Virtual Environments

Robot teleoperation allows the information captured by the system to be demonstrated in an effective way, since the number of degrees of freedom being controlled can be determined through the selection of the robot. In order to test the practical applications of the system, a 7DoF LBR iiwa from Kuka was

controlled to perform several tasks in a virtual environment. The tasks involved a combination of both arm movement and muscle contraction. The virtual robot was designed to track the end position of the subject's hand, although virtual constraints were also included. The kinematics for the Kuka robot used in this experiment are available, however a lightweight implantation was derived for the virtual environment.

6.6.1 Kuka Kinematic Derivation

In order to provide control to the Kuka, a direct mapping from the hand of the subject to the end effector of the robot was used. The position of the subject's hand was calculated in the same way and the previous experiments and defined as the target position of the end effector of the robot. To provide this visualisation, the joint angles to move the end effector of the Kuka to the desired position were required. The constraints of the visualisation did not require a full kinematic solution to be implemented, so a lightweight controller was designed and implemented instead. The design of this controller is described in Appendix III.

This method of estimation allows the instantaneous joint angles for a given desired end effector position to be calculated, but this is not enough to realistically control the virtual robot. To ensure that the user would have to actively participate in controlling the robot, each joint was given a maximum angle of rotation per time period (i.e. a maximum speed). Including this required the subjects to factor the current position of the robot into their control strategies, increasing the realism of the simulation. Using this method, a lightweight implementation for end position driven Kuka control can be implemented.

6.6.2 Experimental Protocol

6.6.2.1 Participants

Subjects who met the inclusion criteria were selected and gave their written informed consent to participate in the experiment. Five healthy subjects (4 male, 1 female, average age: 38.4yrs SD: 15.8yrs) were involved in this work. Participants had a range of experience both in working in VR environments and with gesture-based interfaces.

6.6.2.2 Training Protocol

The training protocol was designed to provide data analogous to that generated during the robot teleoperation test phase, however several modifications were required to generate an appropriate quantity of data for the experiment without inducing fatigue in the subjects. In order to train the system, subjects wore two IMUs, one placed on the upper right arm, and one on the lower right arm. Subjects wore six MMG sensors, worn in a band around the right forearm, with three placed approximately over the flexor digitorum muscle group and three over the extensor digitorum muscle group (Placement can be seen in Figure 34). Subjects were seated in an upright position and had access to an armrest.

Figure 34 – Position of armband for gesture-based experiments

Based on the work performed in the previous section, five arm configurations were identified which provided a good representation of the workspace used in this experiment. These configurations varied in both elbow angle and forearm orientation, but in each position the subject's elbow was supported by the armrest, whose height was varied to ensure support in each arm configuration. Configurations were kept consistent between subjects.

Four control gestures were identified for the tasks in this experiment; Open hand, Close hand, Point index finger, and Pinch with index finger and thumb. Subjects were asked to make five instances of each gesture, and this was repeated three times for each of the five positions. Subjects were given thirty seconds between each recording to avoid fatigue. This resulted in a total of three hundred gestures recorded for each subject.

6.6.2.3 Testing Protocol

To test the inertial measurement and mechanomyography fusion for robot teleoperation in virtual environments, a series of tasks were created with the aim to allow the user to interact with the robot without their focus being on the interface. The tasks fell into three categories:

- Ball-to-Box (B2B) tasks – where subjects were required to register an Open gesture to select a gripper tool with the robot, move the gripper to a virtual ball, register a Close gesture to pick up the ball, move the ball above a virtual box, and register an open gesture to drop the ball into the box.
- Balloon-Popping (BP) tasks – where subjects were required to register a Point gesture to select a sharp tool on the robot's end effector, and then move that tool into contact with several virtual balloons to pop them.

- Lock-and-Key (L&K) tasks, where the user was required to register a Pinch gesture to select the key tool mounted below the robot's end effector, then place the key precisely into a virtual lock.

The three tasks were intended to utilise both the gesture-based control obtained through both inertial and mechanomyographic monitoring, as well as force the subjects to generate gestures in a diverse array of positions.

Once subjects had completed the training phase, they were invited to begin the test phase immediately. The tasks were explained, and the gesture required to complete them were described. Subjects were then asked to put on the virtual headset and were instructed to remove it if they felt any ill effects as a result

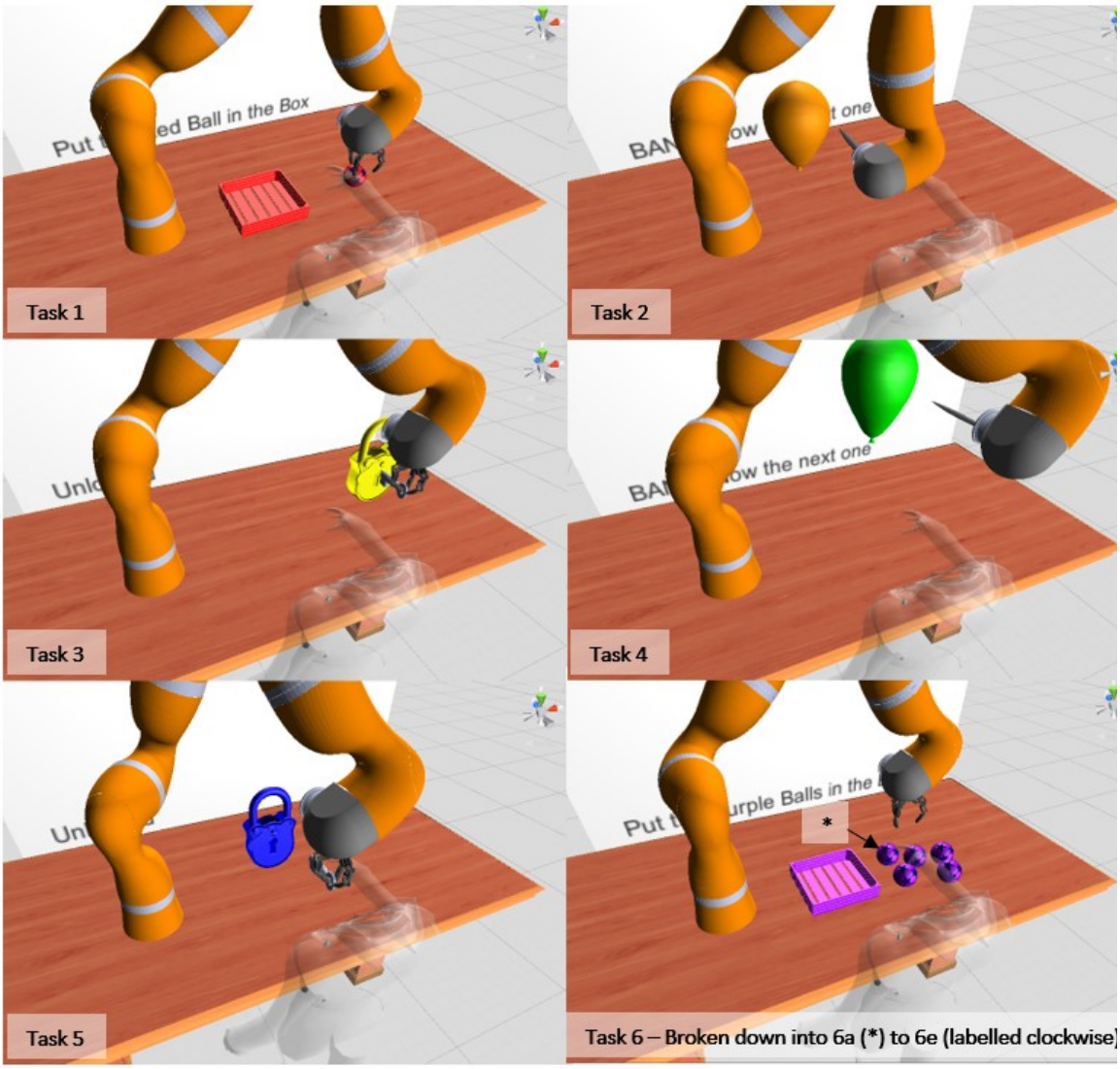

Figure 35 - The six tasks for VR Robot Control

of wearing it. There were invited to explore the test space and familiarise themselves with both the response of the robot, and the visual feedback triggered by performing the various control gestures. The limits to joint speed on the robot caused a delay between the user moving to the correct position, and the robot reaching that same position. Instantaneous user position feedback within the virtual environment was given by a 'ghost' figure that occupied the same position as the user and replicated their movement. After one minute, the subject was invited to begin the experiment.

The experimental tasks were as follows: one B2B task, four sequential BP tasks where completing one causes a new object to appear in a different location, two sequential L&K tasks, four sequential BP tasks, two sequential L&K tasks and five sequential B2B tasks which can be undertaken in any order. The time taken to complete each task was recorded, as well as the gestures subjects made during each one. An over-the-shoulder view of the tasks can be seen in Figure 35.

Once the final task had been completed, subjects were asked to remove the VR headset, and were given a pair of handheld controllers to be used with it. The controller's buttons that were programmed to incite the same behaviour as the gestures and the subjects were asked to familiarise themselves with the button's placement and function. Once they had done this, they were asked to wear the VR headset again, and complete the same series of tasks using the controllers. All aspects of the experiment were kept the same, with the exception of the instantaneous feedback, which was replaced by a ball that occupied the same space as the user's hand.

6.6.3 Results

There are two metrics that can be used to assess the effectiveness of the system displayed in this experiment. The first is the time taken to complete the overall experiment, summarised in Table 27.

Table 27 - Time taken to complete VR Robot Teleoperation experiment

Time(minutes : seconds)			
NUIMU		Oculus handheld controllers	
Mean	SD	Mean	SD
2:53	0:54	1:29	0:19

The second metric is the accuracy of the classification. In order to complete the experiment, subjects were required to make eighteen gestures. In addition to these eighteen volitional control commands that all subjects completed, several other gestures were recorded. These other gestures were categorised into three groups:

- Misclassifications – where the user had intended to perform a volitional command, but it was the incorrect command for their current point in the experiment (gestures that were incorrect due to human error were counted as misclassifications).
- False Positives – gestures that were the result of motion induced artefacts or muscular twitches when no gesture was intended.

- Corrective gestures – gestures that were correctly classified, but not an intended part of the experiment, corrective gestures were caused as users overcame any false positives that changed the current state of the robot from its intended state.

In this application, accuracy (ACC) is defined as:

$$ACC = \frac{G_c}{G_c + G_i} \tag{65}$$

where G_c is the number of correctly classifier gestures, and G_i is the number of misclassifications. The accuracies were worked out for each individual and the average was taken. Average accuracies can be seen in Table 28.

Table 28 – Accuracy of commands in VR Robot Teleoperation experiment

Accuracy of gesture/command classification			
NUIMU		Oculus handheld controllers	
Mean	SD	Mean	SD
71.38%	18.44%	93.89%	3.72%

Another useful metric when evaluating the usability of this system in practical applications is precision. Precision (PR) describes the false positives detected by the system, and is defined as:

$$PR = \frac{P_t}{P_t + P_f} \tag{66}$$

where P_t and P_f describe the number of true positives and false positives respectively. It should be noted here that the true positives are not required to be correctly classified, since accuracy is a separate metric. As a result, if corrective gestures are defined as GC, then:

$$P_t = G_c + G_i + GC \tag{67}$$

A description of the precision is summarized in Table 29.

Table 29 – Precision of commands in VR Robot Teleoperation experiment

Precision of gesture/command classification			
NUIMU		Oculus handheld controllers	
Mean	SD	Mean	SD
90.9%	5.8%	99.1%	1.9%

In order to examine these results in further detail, it was useful to examine the tasks that caused the greatest number of misclassifications and false positives. A summary of these breakdowns can be seen in Table 30.

Table 30 – Misclassifications and false positives separated by task

	Number of gestures required to complete task	Average number of Misclassifications	Average number of false positives
Task 1	3	0.2	0.2
Task 2	1	1.4	0.6
Task 3	1	0	0.8
Task 4	1	0.6	0.8
Task 5	1	0.2	0.4
Task 6a	3	1.4	0
Task 6b	2	0.4	0
Task 6c	2	0.6	0.2
Task 6d	2	3.4	0.4
Task 6e	2	1.4	0.2

6.6.4 Discussion

The completion of this experiment demonstrates that the sensor system is capable of interpreting user intent in dynamic conditions. Users were able to complete all tasks during a single run through and did not report fatigue from using the system.

Several contributory factors to the lower that optimal accuracy can be derived by examining the data. The most obvious is that while the training data was intended to be analogous to the required tasks, they were recorded using a different protocol. It was found in early testing that training in the VR environment by performing the tasks repeatedly led to much faster rates of fatigue, making the experiment uncomfortable, and in some cases impossible to complete. The training protocol observed here addressed that, however as a result subjects did not associate the gestures they were performing with the intended action until the test phase. It was observed that during the training phase, subjects were making large, deliberate gestures, however in the VR environment subjects appeared to initially make smaller, more precise gestures. Another possible contributory factor in the experimental protocol was the inclusion of the armrest. While the armrest was required to train the classifier in multiple arm configurations to avoid fatigue, it was not available to subjects during testing as it would have inhibited the subject's freedom of movement. The additional stress induced in the arm when held unsupported in position appeared to increase the background mechanomyographic noise, which may have led to misclassifications.

The tasks that recorded the greatest number of false positives were the L&K and the BP tasks. These are both tasks that required the user to perform large arm motions, and therefore it is likely that some motion induced artefacts may have led to false positives being detected. Both tasks required the user to make an initial 'tool selecting' gesture, and the number of misclassifications shows that on average, users found the point gesture to be more difficult to register. It was observed that the point gesture was the least

consistent during the test, with users switching between several different forms of the same gesture within a single test.

A data point that appears to be an outlier in Table 30 is the number of misclassifications that occurred during Task 6d. Task 6d was a B2B task, but it induced significantly higher errors than the other B2B tasks that were performed in Task 6. It was observed during the experiment that since the ball was close to the body, reaching it with the Kuka required the subjects to adopt a unique arm pose. This pose required an elbow angle that was more extreme than any of the training positions, causing errors comparable to Position 5 in Table 21.

6.7 Chapter Summary

This chapter describes the application of a system based on fused mechanomyography and inertial measurement for human-robot interface. The chapter examined the use of hand position as an intuitive method of providing part of this robot control. NUIMUs were worn on the arm segments, and a simplified form of forward kinematics was used to translate these three-dimensional data points into command signals. The accuracy and the effect of a reduced accuracy were tested using a virtual environment. Information indicative of subject's path planning was observed, and it was concluded that while users could complete movements quickly, providing feedback that was not aligned with the user's proprioceptive sense led to patterns of motion that could be fatiguing.

A Baxter robot was used as a real-world demonstration of the first implementation of the system. End position and gestures were used as control signals. In this case, subjects were manipulating physical objects using a combination of four NUIMUs and six MMGs to provide dual arm control of the 14DoF robotic platform.

Since end-point position was used as a channel for control signals, a requirement that gestures could be performed at any location in the effective workspace was observed. To ensure that the system described here was capable of that, gestures were recorded at a range of different positions defined so as to vary both elbow angle and the weight supported by the muscles when in the relaxed state. It was found that both variables had a noticeable effect on the signal, as both the LDA and linear SVM had large errors when classifying between positions. In order to combat this increase in complexity, a cubic SVM was introduced and implemented.

While the cubic SVM performed well when trained and tested on data taken from the same position, accuracy decreased when data from other positions was included in the training set. To ensure that data in the training set was representative of the test data, the forward kinematics of the arm were used to label each instance in the training data. In order to classify test data, the end position was calculated, and a modified K-Nearest Neighbour algorithm was used to select training data from other close positions. This training data contained data more representative of the test instance, and therefore when used to train the cubic SVM, a higher accuracy was possible.

Since gesture-based HMI applications require the user to make individual gestures, the overall information throughput is not high. As a result, training a classifier for every instance of data in the test set, as described by the KNN-cSVM fusion is a feasible option, even though it would not be appropriate for many machine learning applications. This method provides a true fusion of mechanomyography and inertial measurement data, allowing a higher accuracy to be achieved than through a single sensor alone.

This updated method of gesture classification and end point mapping was tested in a virtual environment using a 7-DoF Kuka robot. A lightweight inverse kinematic solution for calculating joint angled from end position was derived and implemented. A maximum joint rotation speed was implemented, both to add a level of realism to the simulation, but also to introduce a disconnect between the user and the robot end effector. This ensured that the user was more focused on the visual feedback provided and lessened the effect of errors introduced by the user's proprioceptive sense. Users were able to complete the series of tasks designed to test their control of the virtual robot.

In both experiments, the sensor suite was tested against industry standard interfaces. While the interface demonstrated here did not allow subjects to perform the experiments as fast as the commercial interfaces, its pervasive form means no requirement for cameras in the environment and favourable lighting conditions, or dexterous manipulation of a physical controller. As such it demonstrates an alternative interface platform that can be built upon in the future.

Chapter 7

Conclusions and Future Work

7.1 Thesis Summary

This thesis details the design and development of a new form of Human-Machine Interface which utilises a novel fusion of mechanomyography and inertial measurement to derive volitional control commands. The fusion of these two sensing modalities provide significant advantages over existing methods of interface which rely on alternative technologies. MMG sensors have previously been used to detect physiological features such as fatigue but have also been reported to detect motion induced artefacts. The inclusion of the inertial sensors in the system allows these artefacts to be identified and compensation strategies to be implemented. Additionally, the effect of arm orientation on pattern recognition accuracy has been observed, and new strategies of training have been developed to create a system which is robust to those changes. As a result, the first MMG-based gesture recognition system which is robust to both environmental interference and arm position has been developed. This new interface has been proved effective in a variety of different applications, including robot control and interaction in VR environments, demonstrating robust classification in practical applications. Several shortcomings were identified that have led to poor adoption of gesture control in the HMI market, and high levels of dissatisfaction and rejection in the prosthetics market. The work presented here demonstrates the utility of an alternative interface technology which, although less mature, has the potential to overcome a number of the issues raised by existing techniques.

MMG belongs to a small group of technologies that allow for biomechanical information (such as hand gestures) to be recorded through inference, as opposed to direct monitoring. This allows for the creation of interfaces that do not inhibit the generation of the control signals as they are recorded. Other technologies, such as vision or physical interfaces, often either constrain the users physically or by requiring them to stay within the cameras field of view. Pervasive interfaces build on MMG or EMG do not, and therefore are preferable for situations where the user may not be in clinical conditions. Additionally, MMG has a number of benefits over EMG when it comes to signal capture, it is not reliant on the electrical condition of the skin, which means that it can be more robust over extended periods of time. This is particularly true in the conditions inside a prosthetic socket, where non-breathable materials quickly lead to perspiration, which affects the skin impedance, a problem when examining the electrical signal but not the mechanical signal.

After an examination of existing HMI technologies both from a technological perspective and an end-user perspective, hardware to allow the collection of MMG and IMU data was designed and built. This process started with an identification of desired hardware and functional specifications, which allowed a small, high data rate package to be constructed. Particular care was taken to ensure that the hardware formed a platform that could be expanded to incorporate external hardware and further software modifications. Additionally, several software packages were constructed to allow the IMUs to be controlled from a number of host devices, including android mobiles and Windows PCs. The software system was capable of acting as both a local input for 3rd party software, such as Unity, or network applications through TCP/IP, enabling it to act as a ROS node.

The host software for Windows computers was structured in two projects, one to control the IMUs and one to control the Graphical User Interface. This provided a demonstration of the capabilities of the systems without being constrained to a specific interface. Additionally, the firmware on the IMUs was reconfigurable through this demonstration project, allowing the data rate to be tuned as per the requirements of the specific experimental

protocols. As a result, there are many projects for which this hardware package could be considered suitable. The design process and software functionality are detailed in Chapter 3.

Data from the MARG sensors can be used to approximate a world reference orientation, which is important for understanding motion, as well as providing contextual information for gesture recognition. There are three commonly used methods of determining orientation based on MARG data, complementary filters, Kalman filters and optimization methods. Of these, optimization methods such as Madgwick's Gradient Descent Algorithm are typically the easiest to implement, while still providing high levels of accuracy, however the GDA in particular has a number of shortcomings that become apparent during practical implementation. This includes both a slow convergence rate at high levels of magnetic inclination, as well as coupling between perpendicular Euler rotations. This work demonstrated modifications to this algorithm that addressed these shortcomings, leading to a faster convergence. Additionally, a second algorithm was proposed to act as an alternative to gradient descent-based algorithms. This new algorithm calculated the exact rotation of the IMU based on instantaneous sensor data. This allows for an implementation that is more similar to the complementary filter, but which also uses magnetometer data. The primary benefit of this implementation is that it separates sensor filtering and sensor fusion, allowing sensor trust to play a greater role. Details of this implementation, as well as the proposed improvements to the gradient descent orientation estimation algorithm are found in chapter 4.

Once the MMG data has been recorded, it needs to be interpreted. This interpretation was broken down into three steps, segmentation, classification and interpretation. The interpretation of the classified gestures is application specific, and therefore is dependent on the implementation. The segmentation was performed by energy thresholding on both the MMG data and the inertial data. This allowed motion induced artefacts to be detected and dismissed, ensuring that only volitional control signals were detected. A large dataset was taken, and a human assisted segmentation was performed. This allowed gestures to be identified and marked within the set, which facilitated testing and tuning of real-time detection techniques. Classification was performed using a number of techniques. First, a template-based method was designed. This was intended to operate purely as a method of determining the repeatability and distinguishability of the signals recorded, however this purely statistical technique proved successful in as a lightweight, real-time classification technique. Further methods from the field of machine learning were also tested, including decision trees, KNNs, LDAs and SVMs. Additionally, PCA and feature extraction were explored as dimensionality reduction techniques. A suitable real-time classification technique was identified and implemented for further testing. This series of experimental work is described in Chapter 5.

Finally, several experiments were designed to demonstrate the practical use of the systems as a Human Machine Interface. A first-generation implementation of the NUIMU/MMG system was used in the real-world experiments involving the teleoperation of robotic platforms. The level of abstraction in this application made for an intuitive form of interface, however error in classification introduced difficulties. In order to determine the source of the errors, experiments were performed in a virtual environment to provide ideal experimental conditions in a controlled environment. This allowed the performance of the system to be assessed against alternative commercial interfaces. The experiments showed that the IMUs could be used to bring more degrees of freedom into a virtual world, however when joints were directly mapped this introduced a conflict between the subject's proprioceptive sense and the visual feedback that was being provided. Arm orientation and elbow angle were identified as potential factors that may have

led to the misclassification observed in the teleoperation experiment, and evidence that supported this was found. A compensation strategy which used a more in-depth form of sensor fusion was found and tested in VR robot teleoperation tasks. Details of these experiments can be found in Chapter 6.

7.2 Summary of Contributions

The core contribution of this thesis is to demonstrate an improvement to the understanding of the behaviour of Mechanomyographic signals while performing different gestures, and the place of this understanding in the context of gesture-based Human Machine Interfaces. The sensor has previously been researched primarily as a diagnostic tool, and its potential as a practical alternative to EMG-based interfaces has been largely ignored. This thesis has demonstrated that the sensor can operate in this field and is worthy of further research and development.

The specific contributions of this thesis are as follows:

- Creation of sensor-suite
    - The creation of a small form, high frequency, multisensor interface comprising of a combination of IMU and MMG sensors is presented, allowing for a sensor fusion that addresses some of the shortcomings of each sensor individually.
- Improvements to commonly used orientation estimation algorithm
    - A simulated demonstration of the commonly used 'Madgwick' algorithm (Gradient Descent Algorithm) showed that convergence is dependent on geomagnetic inclination, and that this can lead to slow convergence in areas of high inclination. A method to overcome this is proposed, and it is demonstrated to provide consistent convergence regardless of geomagnetic inclination. At 60 degrees, this implementation converges six times faster than the original gradient descent algorithm. This solution is also shown to decouple pitch/roll and yaw during correction, leading to more efficient convergence.
- Introduction of a new orientation estimation algorithm
    - The derivation of a new algorithm for estimating the orientation of a MARG system with respect to the global reference frame is proposed. This solution provides a configurable step convergence, which allows it to be easily tuned both for convergence speed and for sensor reliability. This provides more control over the nature of the convergence and the filtering of the signals than other commonly used algorithms. In its base form, the algorithm provides one steps convergence.
- MMG gesture classification
    - Offline analysis has shown that the MMG signal is sufficiently repeatable to permit classification of movement using several classification techniques, which range from purely statistical methods to more advanced machine learning techniques. A labelled dataset of MMG data that can be used to test other classification (and segmentation) techniques in the future has also been generated. The methods implemented here achieve an offline accuracy of ~95% across 12 distinct gestures. Online implementations were trialled, and accuracies on 94.7% and 90.8% were achieved for 2 and 5 gesture classifications respectively.

- A new form of IMU/MMG fusion to provide a classification accuracy that is robust to changes in the signal caused by arm pose was implemented. It was demonstrated that changing arm pose between training and testing caused classification accuracy to drop ~30%. The algorithm proposed here ensured that training data from the most relevant positions were used, leading to an offline accuracy of 89.6% classifying four gestures from seven diverse arm poses.

- Demonstration of Applications
  - The real-time multi-model sensor system was implemented in several applications, demonstrating the first MMG/IMU-based robot teleoperation system that allows for hands free robot control in unstructured environments.

7.3 Suggested Future Directions

While the MMG/IMU combination has been demonstrated to provide enough data for reliable gesture classification, there are many extensions to the system demonstrated here, which could be implemented. The extensions outlined here could lead to increased robustness in real-world applications by removing some of the system constraints.

The approach shown here relies on the classification of segmented gestures. This was intended to demonstrate the repeatable nature of the signal generation for similar gestures, but also introduces a source of error based on incorrect segmentation. The energy of the MMG signal is proportional to the speed of motion of the inciting gesture, and therefore gesture speed must be kept consistent to ensure that the segmentation is selecting the desired portion of the signal. This could be solved through the introduction of a continuous classification algorithm that actively classifies periods where a gesture is not present. This would also allow inertial data to be used as additional inputs for classification, removing the need for manually set threshold values. Algorithms such as Convolutional Neural Networks may provide such a solution.

Further testing on the sensor longevity and consistency of signal between sensors must also be examined. While short term classification is independent of these factors, the commercialization of any such system must address these issues. The system must also be tested during a wider range of activities of daily living, both in its current form and inside a prosthetic socket. Like other muscle sensors, MMGs are known to be affected by the pressure applied to them, and therefore their response while inside a socket that is also supporting the weight of a prosthetic must be assessed.

Increasing the number of MMG sensors in the system would be desirable. Sensors placed lengthways along the muscles in the forearm could allow for muscle contractions to be further examined by tracking the progression of the contraction down the length of the muscle. This could provide another method of noise rejection for the signal. Similarly, simultaneous comparison of sensors that are placed over the muscle body compared with sensors placed over areas such as the wrist of the end of the residual limb could also provide a method of isolating the muscle activity.

Further sensor fusion should be examined. MMG sensors are unaffected by several the issues that cause calibration problems with other sensor types (such as EMG), but significantly more work has been performed using EMG sensors. Fusion of the two sensors could allow for both improved noise correction and classification, as well as on-the-fly recalibration of the EMG sensing elements. This could allow such a system to take advantage of the benefits of both sensors.

The work outlined here explores the use of hand gestures as a means of generating commands, however there are a wide variety of other applications for such an intent detection system. As a prosthetic interface, the system could provide a method of detecting intended gait, resulting in responsive lower limb prosthetics. As an assistive device, the system has applications ranging from detecting the progression and technique behind rehabilitation exercises to acting as an interface for ALS patients. The system provides information about the biomechanical processes of the human body in detail, without requiring a controlled and structured environment. As such there is a large body of work yet to be completed with respect to these sensing modalities, and the experiments outlined in this document are only the beginning.

7.4 Final Comments

This work has investigated the monitoring of mechanomyographic signals as a means of detecting volitional commands generated through hand gestures. It has demonstrated that the process of performing a gesture produces repeatable signals that provide the enough information to be categorized. It has tested several methods of classifying the signals and found that high accuracy can be achieved. New methods of orientation estimation have been implemented, and the fusion of MMG and IMU data has been extended. The system is capable of recording data that is usually only available in a controlled environment and utilizing it for the purpose of providing natural and intuitive gesture-based control. As the number of smart devices with which we interact daily continues to increase, and as technology becomes more ubiquitous, we will find more and more that we interact with it through virtual interfaces that rely on natural movements, including hand gestures, and it is possible that MMG may be the technology that facilitates this vision.

References

1. Alves, N. and T. Chau. *Classification of the mechanomyogram: Its potential as a multifunction access pathway*. in *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 2009.
2. Park, K.-H. and S.-W. Lee. *Movement intention decoding based on deep learning for multiuser myoelectric interfaces*. in *2016 4th International Winter Conference on Brain-Computer Interface (BCI)*. 2016. IEEE.
3. Alves, N. and T. Chau, *Uncovering patterns of forearm muscle activity using multi-channel mechanomyography.* J Electromyogr Kinesiol, 2010. **20**(5): p. 777-86.
4. Atzori, M., M. Cognolato, and H. Muller, *Deep Learning with Convolutional Neural Networks Applied to Electromyography Data: A Resource for the Classification of Movements for Prosthetic Hands.* Front Neurorobot, 2016. **10**(9): p. 9.
5. Cao, W., et al. *Identifying hand-motion patterns via kernel discriminant analysis based dimension reduction and quadratic classifier*. in *2011 International Conference on Wavelet Analysis and Pattern Recognition*. 2011.
6. Geng, W., et al., *Gesture recognition by instantaneous surface EMG images.* Sci Rep, 2016. **6**: p. 36571.
7. Ma, Y., et al. *Hand gesture recognition with convolutional neural networks for the multimodal UAV control*. in *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*. 2017.
8. Purushothaman, G. and R. Vikas, *Identification of a feature selection based pattern recognition scheme for finger movement recognition from multichannel EMG signals.* Australas Phys Eng Sci Med, 2018. **41**(2): p. 549-559.
9. Teh, Y., R.B. Woodward, and L.J. Hargrove. *Comparing the Effects of Signal Noise on Pattern Recognition and Linear Regression-Based Myoelectric Controllers*. in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2018.
10. Zhang, Z., et al., *Real-Time Surface EMG Pattern Recognition for Hand Gestures Based on an Artificial Neural Network.* Sensors (Basel), 2019. **19**(14): p. 3170.
11. Hancock, P.A.a.C., M. H., *Intelligent Interfaces: Theory, Research, and Design*. 1989, New York, NY, USA: Elsevier Science Inc.
12. Myers, B.A., *A brief history of human-computer interaction technology.* interactions, 1998. **5**(2): p. 44-54.
13. Rogers, E.M., *Diffusions of Innovation*. 6 ed. 1962, London: Collier Macmillan Publishers.

14. Ziegler-Graham, K., et al., *Estimating the Prevalence of Limb Loss in the United States: 2005 to 2050.* Archives of Physical Medicine and Rehabilitation, 2008. 89(3): p. 422-429.
15. Ostlie, K., et al., *Prosthesis use in adult acquired major upper-limb amputees: patterns of wear, prosthetic skills and the actual use of prostheses in activities of daily life.* Disabil Rehabil Assist Technol, 2012. 7(6): p. 479-93.
16. Biddiss, E.A. and T.T. Chau, *Upper limb prosthesis use and abandonment: a survey of the last 25 years.* Prosthet Orthot Int, 2007. 31(3): p. 236-57.
17. Wright, T.W., A.D. Hagen, and M.B. Wood, *Prosthetic usage in major upper extremity amputations.* J Hand Surg Am, 1995. 20(4): p. 619-22.
18. Biddiss, E. and T. Chau, *Upper-limb prosthetics: critical factors in device abandonment.* Am J Phys Med Rehabil, 2007. 86(12): p. 977-87.
19. Durmus, D., et al., *The relationship between prosthesis use, phantom pain and psychiatric symptoms in male traumatic limb amputees.* Compr Psychiatry, 2015. 59: p. 45-53.
20. Hanley, M.A., et al., *Chronic pain associated with upper-limb loss.* Am J Phys Med Rehabil, 2009. 88(9): p. 742-51; quiz 752, 779.
21. Gambrell, C.R., *Overuse Syndrome and the Unilateral Upper Limb Amputee: Consequences and Prevention.* JPO Journal of Prosthetics and Orthotics, 2008. 20(3): p. 126-132.
22. Biddiss, E., D. Beaton, and T. Chau, *Consumer design priorities for upper limb prosthetics.* Disabil Rehabil Assist Technol, 2007. 2(6): p. 346-57.
23. Cook, A.M. and J.M. Polgar, *Principles of Assistive Technology*, in *Assistive Technologies*, A.M. Cook and J.M. Polgar, Editors. 2015, Mosby: St. Louis (MO). p. 1-15.
24. Matos, A., et al., *A Myographic-based HCI Solution Proposal for Upper Limb Amputees.* Procedia Computer Science, 2016. 100: p. 2-13.
25. Plettenberg, D.H. *PROSTHETIC CONTROL: A CASE FOR EXTENDED PHYSIOLOGICAL PROPRIOCEPTION.* . in *Myoelectric Symposium*. 2002.
26. Doubler, J.A. and D.S. Childress, *An analysis of extended physiological proprioception as a prosthesis-control technique.* J Rehabil Res Dev, 1984. 21(1): p. 5-18.
27. Cui, J., et al. *A review of teleoperation system control*. in *Proceedings of the Florida Conference on Recent Advances in Robotics (FCRAR)*. 2003.
28. Vitiello, V., et al., *Emerging Robotic Platforms for Minimally Invasive Surgery.* IEEE Reviews in Biomedical Engineering, 2013. 6: p. 111-126.

29. Smith, T.A., et al. *A Novel Haptic Interface for Navigation in Large Volume Environments*. in *Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'07)*. 2007.
30. Sokho, C., et al. *KIST teleoperation system for humanoid robot*. in *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289)*. 1999.
31. Rocon, E., et al., *Design and Validation of a Rehabilitation Robotic Exoskeleton for Tremor Assessment and Suppression.* IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2007. **15**(3): p. 367-378.
32. Kim, H., et al., *Kinematic Data Analysis for Post-Stroke Patients Following Bilateral Versus Unilateral Rehabilitation With an Upper Limb Wearable Robotic System.* IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2013. **21**(2): p. 153-164.
33. Tognetti, A., et al., *Wearable Goniometer and Accelerometer Sensory Fusion for Knee Joint Angle Measurement in Daily Life.* Sensors (Basel, Switzerland), 2015. **15**(11): p. 28435-28455.
34. Sharma, N., et al., *Closed-Loop Neural Network-Based NMES Control for Human Limb Tracking.* IEEE Transactions on Control Systems Technology, 2012. **20**(3): p. 712-725.
35. Gibbs, P.T. and H. Asada, *Wearable Conductive Fiber Sensors for Multi-Axis Human Joint Angle Measurements.* Journal of NeuroEngineering and Rehabilitation, 2005. **2**(1): p. 7.
36. DeGol, J., et al. *Automatic grasp selection using a camera in a hand prosthesis*. in *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2016.
37. Gardner, M., et al. *An unobtrusive vision system to reduce the cognitive burden of hand prosthesis control*. in *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*. 2014.
38. Dosen, S., et al., *Cognitive vision system for control of dexterous prosthetic hands: experimental evaluation.* Journal of neuroengineering and rehabilitation, 2010. **7**: p. 42-42.
39. Markovic, M., et al., *Sensor fusion and computer vision for context-aware control of a multi degree-of-freedom prosthesis.* J Neural Eng, 2015. **12**(6): p. 066022.
40. Ghazaei, G., et al., *Grasp Type Estimation for Myoelectric Prostheses using Point Cloud Feature Learning.*

41. Muro-de-la-Herran, A., B. Garcia-Zapirain, and A. Mendez-Zorrilla, *Gait analysis methods: an overview of wearable and non-wearable systems, highlighting clinical applications.* Sensors (Basel, Switzerland), 2014. **14**(2): p. 3362-3394.
42. Collins, T.D., et al., *A six degrees-of-freedom marker set for gait analysis: Repeatability and comparison with a modified Helen Hayes set.* Gait & Posture, 2009. **30**(2): p. 173-180.
43. Kadaba, M.P., H.K. Ramakrishnan, and M.E. Wootten, *Measurement of lower extremity kinematics during level walking.* J Orthop Res, 1990. **8**(3): p. 383-92.
44. Cloete, T. and C. Scheffer. *Benchmarking of a full-body inertial motion capture system for clinical gait analysis*. in *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 2008.
45. Wren, C., et al. *Pfinder: real-time tracking of the human body*. in *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*. 1996.
46. Mikic, I., et al. *Articulated body posture estimation from multi-camera voxel data*. in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. 2001.
47. Luck, J., D. Small, and C.Q. Little. *Real-Time Tracking of Articulated Human Models Using a 3D Shape-from-Silhouette Method*. in *Robot Vision*. 2001. Berlin, Heidelberg: Springer Berlin Heidelberg.
48. Chi-Wei, C., O.C. Jenkins, and M.J. Mataric. *Markerless kinematic model and motion capture from volume sequences*. in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* 2003.
49. Song, W., et al. *Teleoperation Humanoid Robot Control System Based on Kinect Sensor*. in *2012 4th International Conference on Intelligent Human-Machine Systems and Cybernetics*. 2012.
50. Almetwally, I. and M. Mallem. *Real-time tele-operation and tele-walking of humanoid Robot Nao using Kinect Depth Camera*. in *2013 10th IEEE INTERNATIONAL CONFERENCE ON NETWORKING, SENSING AND CONTROL (ICNSC)*. 2013.
51. Du, G. and P. Zhang, *Markerless human–robot interface for dual robot manipulators using Kinect sensor.* Robotics and Computer-Integrated Manufacturing, 2014. **30**(2): p. 150-159.
52. Weichert, F., et al., *Analysis of the Accuracy and Robustness of the Leap Motion Controller.* Sensors, 2013. **13**(5): p. 6380.

53. Biswas, K.K. and S.K. Basu. *Gesture recognition using Microsoft Kinect®*. in *The 5th International Conference on Automation, Robotics and Applications*. 2011.
54. Zafrulla, Z., et al., *American sign language recognition with the kinect*, in *Proceedings of the 13th international conference on multimodal interfaces*. 2011, ACM: Alicante, Spain. p. 279-286.
55. Kolkur, S., et al. *Human Skin Detection Using RGB, HSV and YCbCr Color Models*. 2016. Atlantis Press.
56. Du, G., et al. *Robot teleoperation using a vision-based manipulation method*. in *2010 International Conference on Audio, Language and Image Processing*. 2010.
57. Wang, R.Y., J. Popovi, and #263, *Real-time hand-tracking with a color glove.* ACM Trans. Graph., 2009. **28**(3): p. 1-8.
58. Chang, Y.S., et al. *Evaluating gesture-based augmented reality annotation*. in *2017 IEEE Symposium on 3D User Interfaces (3DUI)*. 2017.
59. Li, Y.D. and E.T. Hsiao-Wecksler. *Gait mode recognition and control for a portable-powered ankle-foot orthosis*. in *2013 IEEE 13th International Conference on Rehabilitation Robotics (ICORR)*. 2013.
60. Langlois, D., M. Rittenhouse, and Y. Roy, *Prosthetic and orthotic devices and methods and systems for controlling the same*. 2018, Google Patents.
61. Woodward, R., S. Shefelbine, and R. Vaidyanathan. *Integrated grip switching and grasp control for prosthetic hands using fused inertial and mechanomyography measurement*. in *2015 Swarm/Human Blended Intelligence Workshop (SHBI)*. 2015.
62. Gardner, M., et al. *Motion-based grasp selection: Improving traditional control strategies of myoelectric hand prosthesis*. in *2015 IEEE International Conference on Rehabilitation Robotics (ICORR)*. 2015.
63. Kyranou, I., et al. *Real-time classification of multi-modal sensory data for prosthetic hand control*. in *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*. 2016.
64. Wilson, S. and R. Vaidyanathan. *Upper-limb prosthetic control using wearable multichannel mechanomyography*. in *2017 International Conference on Rehabilitation Robotics (ICORR)*. 2017.
65. Miller, N., et al. *Motion capture from inertial sensing for untethered humanoid teleoperation*. in *4th IEEE/RAS International Conference on Humanoid Robots, 2004.* 2004.

66. Kim, S., S. Hong, and D. Kim. *A walking motion imitation framework of a humanoid robot by human walking recognition from IMU motion data*. in *2009 9th IEEE-RAS International Conference on Humanoid Robots*. 2009.
67. Reiss, A. and D. Stricker. *Introducing a New Benchmarked Dataset for Activity Monitoring*. in *2012 16th International Symposium on Wearable Computers*. 2012.
68. Attal, F., et al., *Physical Human Activity Recognition Using Wearable Sensors.* Sensors, 2015. **15**(12): p. 29858.
69. Wittmann, F., et al., *Self-directed arm therapy at home after stroke with a sensor-based virtual reality training system.* Journal of NeuroEngineering and Rehabilitation, 2016. **13**(1): p. 75.
70. Cifuentes, C., et al. *Development of a wearable ZigBee sensor system for upper limb rehabilitation robotics*. in *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*. 2012.
71. Fenu, G. and G. Steri. *IMU based post-traumatic rehabilitation assessment*. in *2010 3rd International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL 2010)*. 2010.
72. Spicer, R., et al. *REINVENT: A low-cost, virtual reality brain-computer interface for severe stroke upper limb motor recovery*. in *2017 IEEE Virtual Reality (VR)*. 2017.
73. Geethanjali, P., *Myoelectric control of prosthetic hands: state-of-the-art review.* Med Devices (Auckl), 2016. **9**: p. 247-55.
74. Battye, C.K., A. Nightingale, and J. Whillis, *The use of myo-electric currents in the operation of prostheses.* J Bone Joint Surg Br, 1955. **37-B**(3): p. 506-10.
75. Fougner, A., et al., *Control of upper limb prostheses: terminology and proportional myoelectric control-a review.* IEEE Trans Neural Syst Rehabil Eng, 2012. **20**(5): p. 663-77.
76. Ziai, A. and C. Menon, *Comparison of regression models for estimation of isometric wrist joint torques using surface electromyography.* J Neuroeng Rehabil, 2011. **8**(1): p. 56.
77. Nielsen, J.L.G., et al. *Enhanced EMG signal processing for simultaneous and proportional myoelectric control*. in *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 2009.
78. Rehbaum, H., et al. *Real time simultaneous and proportional control of multiple degrees of freedom from surface EMG: Preliminary results on subjects with limb deficiency*. in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 2012.

79. Dhillon, G.S. and K.W. Horch, *Direct neural sensory feedback and control of a prosthetic arm.* IEEE Trans Neural Syst Rehabil Eng, 2005. **13**(4): p. 468-72.
80. Naik, G.R., et al. *Subtle Hand Gesture Identification for HCI Using Temporal Decorrelation Source Separation BSS of Surface EMG*. in *9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications (DICTA 2007)*. 2007.
81. Kyberd, P.J., et al., *MARCUS: a two degree of freedom hand prosthesis with hierarchical grip control.* IEEE Transactions on Rehabilitation Engineering, 1995. **3**(1): p. 70-76.
82. Kyberd, P.J. and P.H. Chappell, *The Southampton Hand: an intelligent myoelectric prosthesis.* J Rehabil Res Dev, 1994. **31**(4): p. 326-34.
83. Dalley, S.A., H.A. Varol, and M. Goldfarb, *A method for the control of multigrasp myoelectric prosthetic hands.* IEEE Trans Neural Syst Rehabil Eng, 2012. **20**(1): p. 58-67.
84. Segil, J.L. and R.F. Weir, *Design and validation of a morphing myoelectric hand posture controller based on principal component analysis of human grasping.* IEEE Trans Neural Syst Rehabil Eng, 2014. **22**(2): p. 249-57.
85. Zardoshti-Kermani, M., et al., *EMG feature evaluation for movement control of upper extremity prostheses.* IEEE Transactions on Rehabilitation Engineering, 1995. **3**(4): p. 324-333.
86. Tkach, D., H. Huang, and T.A. Kuiken, *Study of stability of time-domain features for electromyographic pattern recognition.* Journal of neuroengineering and rehabilitation, 2010. **7**: p. 21-21.
87. Negi, S., Y. Kumar, and V.M. Mishra. *Feature extraction and classification for EMG signals using linear discriminant analysis*. in *2016 2nd International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Fall)*. 2016.
88. Hannaford, B. and S. Lehman, *Short Time Fourier Analysis of the Electromyogram: Fast Movements and Constant Contraction.* IEEE Transactions on Biomedical Engineering, 1986. **BME-33**(12): p. 1173-1181.
89. Englehart, K., B. Hudgins, and P.A. Parker. *Time-frequency based classification of the myoelectric signal: static vs. dynamic contractions*. in *Proceedings of the 22nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (Cat. No.00CH37143)*. 2000.
90. Maitrot, A., et al., *Signal-dependent wavelets for electromyogram classification.* Med Biol Eng Comput, 2005. **43**(4): p. 487-92.

91. Nazmi, N., et al., *A Review of Classification Techniques of EMG Signals during Isotonic and Isometric Contractions.* Sensors (Basel), 2016. **16**(8): p. 1304.
92. Manimegalai, E., *Hand gesture recognition based on EMG signals using ANN.* Int J Comput Appl, 2013. **3**(2): p. 31-9.
93. Tenore, F., et al. *Towards the Control of Individual Fingers of a Prosthetic Hand Using Surface EMG Signals*. in *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 2007.
94. Zhai, X., et al., *Self-Recalibrating Surface EMG Pattern Recognition for Neuroprosthesis Control Based on Convolutional Neural Network.* Front Neurosci, 2017. **11**(379): p. 379.
95. Wei, W., et al., *A multi-stream convolutional neural network for sEMG-based gesture recognition in muscle-computer interface.* Pattern Recognition Letters, 2019. **119**: p. 131-138.
96. Elamvazuthi, I., et al., *Electromyography (EMG) based Classification of Neuromuscular Disorders using Multi-Layer Perceptron.* Procedia Computer Science, 2015. **76**: p. 223-228.
97. Subasi, A., *Classification of EMG signals using combined features and soft computing techniques.* Applied soft computing, 2012. **12**(8): p. 2188-2198.
98. Young, A.J., et al., *Classification of simultaneous movements using surface EMG pattern recognition.* IEEE Trans Biomed Eng, 2013. **60**(5): p. 1250-8.
99. Gokgoz, E. and A. Subasi, *Comparison of decision tree algorithms for EMG signal classification using DWT.* Biomedical Signal Processing and Control, 2015. **18**: p. 138-144.
100. Khezri, M. and M. Jahed, *A Neuro–Fuzzy Inference System for sEMG-Based Identification of Hand Motion Commands.* IEEE Transactions on Industrial Electronics, 2011. **58**(5): p. 1952-1960.
101. Rossi, M., et al. *Hybrid EMG classifier based on HMM and SVM for hand gesture recognition in prosthetics*. in *2015 IEEE International Conference on Industrial Technology (ICIT)*. 2015.
102. Kim, K.S., et al., *Comparison of k-nearest neighbor, quadratic discriminant and linear discriminant analysis in classification of electromyogram signals based on the wrist-motion directions.* Current applied physics, 2011. **11**(3): p. 740-745.
103. Chu, J.-U., et al., *A supervised feature-projection-based real-time EMG pattern recognition for multifunction myoelectric hand control.* IEEE/ASME Transactions on Mechatronics, 2007. **12**(3): p. 282-290.

104. Linderman, M., M.A. Lebedev, and J.S. Erlichman, *Recognition of handwriting from electromyography.* PLoS One, 2009. **4**(8): p. e6791.
105. Phinyomark, A., et al., *EMG feature evaluation for improving myoelectric pattern recognition robustness.* Expert Systems with applications, 2013. **40**(12): p. 4832-4840.
106. Li, Z., et al., *Boosting-based EMG patterns classification scheme for robustness enhancement.* IEEE J Biomed Health Inform, 2013. **17**(3): p. 545-52.
107. Yang, D., et al. *EMG pattern recognition and grasping force estimation: Improvement to the myocontrol of multi-DOF prosthetic hands*. in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009.
108. Kumar, D.K., S. Poosapadi Arjunan, and V.P. Singh, *Towards identification of finger flexions using single channel surface electromyography--able bodied and amputee subjects.* J Neuroeng Rehabil, 2013. **10**(1): p. 50.
109. Ameri, A., et al., *Support vector regression for improved real-time, simultaneous myoelectric control.* IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2014. **22**(6): p. 1198-1209.
110. Hargrove, L.J., K. Englehart, and B. Hudgins, *A Comparison of Surface and Intramuscular Myoelectric Signal Classification.* IEEE Transactions on Biomedical Engineering, 2007. **54**(5): p. 847-853.
111. Boyali, A., N. Hashimoto, and O. Matsumoto. *Hand posture and gesture recognition using MYO armband and spectral collaborative representation based classification*. in *2015 IEEE 4th Global Conference on Consumer Electronics (GCCE)*. 2015.
112. Abreu, J.G., et al. *Evaluating Sign Language Recognition Using the Myo Armband*. in *2016 XVIII Symposium on Virtual and Augmented Reality (SVR)*. 2016.
113. Xu, Y., et al. *Development of a hybrid motion capture method using MYO armband with application to teleoperation*. in *2016 IEEE International Conference on Mechatronics and Automation*. 2016.
114. Cattarello, P. and R. Merletti. *Characterization of dry and wet Electrode-Skin interfaces on different skin treatments for HDsEMG*. in *2016 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*. 2016.
115. Fukuda, O., et al., *A human-assisting manipulator teleoperated by EMG signals and arm motions.* IEEE Transactions on Robotics and Automation, 2003. **19**(2): p. 210-222.

116. Artemiadis, P.K. and K.J. Kyriakopoulos. *Teleoperation of a robot manipulator using EMG signals and a position tracker*. in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2005.
117. Artemiadis, P.K. and K.J. Kyriakopoulos. *EMG-based teleoperation of a robot arm using low-dimensional representation*. in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2007.
118. Barreto, A.B., S.D. Scargle, and M. Adjouadi, *A practical EMG-based human-computer interface for users with motor disabilities.* J Rehabil Res Dev, 2000. **37**(1): p. 53-63.
119. Smith, D.B., et al., *Mechanomyographic responses to maximal eccentric isokinetic muscle actions.* Journal of Applied Physiology, 1997. **82**(3): p. 1003-1007.
120. Orizio, C., *Muscle sound: bases for the introduction of a mechanomyographic signal in muscle studies.* Crit Rev Biomed Eng, 1993. **21**(3): p. 201-43.
121. Smith, D.B., et al., *Mechanomyographic responses to maximal eccentric isokinetic muscle actions.* J Appl Physiol (1985), 1997. **82**(3): p. 1003-7.
122. Harba, M.I.A. and C. Goh Eng. *Muscle mechanomyographic and electromyographic signals compared with reference to action potential average propagation velocity*. in *Proceedings of the 19th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. 'Magnificent Milestones and Emerging Opportunities in Medical Engineering' (Cat. No.97CH36136)*. 1997.
123. Esposito, F., C. Orizio, and A. Veicsteinas, *Electromyogram and mechanomyogram changes in fresh and fatigued muscle during sustained contraction in men.* Eur J Appl Physiol Occup Physiol, 1998. **78**(6): p. 494-501.
124. Woodward, R., S. Shefelbine, and R. Vaidyanathan. *Pervasive Motion Tracking and Muscle Activity Monitor*. in *Computer-Based Medical Systems (CBMS), 2014 IEEE 27th International Symposium on*. 2014.
125. Farina, D., X. Li, and P. Madeleine, *Motor unit acceleration maps and interference mechanomyographic distribution.* J Biomech, 2008. **41**(13): p. 2843-9.
126. Youn, W. and J. Kim, *Estimation of elbow flexion force during isometric muscle contraction from mechanomyography and electromyography.* Med Biol Eng Comput, 2010. **48**(11): p. 1149-57.
127. Alves, N., T.H. Falk, and T. Chau, *A novel integrated mechanomyogram-vocalization access solution.* Medical engineering & physics, 2010. **32**(8): p. 940-944.
128. Tanaka, M., T. Okuyama, and K. Saito. *Study on evaluation of muscle conditions using a mechanomyogram sensor*. in *2011 IEEE International Conference on Systems, Man, and Cybernetics*. 2011.

129. Orizio, C., et al., *Muscle-joint unit transfer function derived from torque and surface mechanomyogram in humans using different stimulation protocols.* Journal of Neuroscience Methods, 2008. **173**(1): p. 59-66.
130. Beck, T.W., et al., *Cross-correlation analysis of mechanomyographic signals detected in two axes.* Physiological measurement, 2009. **30**(12): p. 1465.
131. Alves, N., et al., *The effect of accelerometer location on the classification of single-site forearm mechanomyograms.* BioMedical Engineering OnLine, 2010. **9**(1): p. 23.
132. Silva, J., et al. *Optimization of the signal-to-noise ratio of silicon-embedded microphones for mechanomyography*. in *CCECE 2003 - Canadian Conference on Electrical and Computer Engineering. Toward a Caring and Humane Technology (Cat. No.03CH37436)*. 2003.
133. Posatskiy, A.O. and T. Chau, *Design and evaluation of a novel microphone-based mechanomyography sensor with cylindrical and conical acoustic chambers.* Med Eng Phys, 2012. **34**(8): p. 1184-90.
134. Cescon, C., et al., *Non-invasive characterization of single motor unit electromyographic and mechanomyographic activities in the biceps brachii muscle.* Journal of Electromyography and Kinesiology, 2006. **16**(1): p. 17-24.
135. Akataki, K., K. Mita, and M. Watakabe, *Electromyographic and mechanomyographic estimation of motor unit activation strategy in voluntary force production.* Electromyography and clinical neurophysiology, 2004. **44**(8): p. 489-496.
136. Carr, J.C., et al., *Mechanomyographic responses for the biceps brachii are associated with failure times during isometric force tasks.* Physiological Reports, 2018. **6**(4): p. e13590.
137. Shinohara, M. and K. Søgaard, *Mechanomyography for Studying Force Fluctuations and Muscle Fatigue.* Exercise and Sport Sciences Reviews, 2006. **34**(2): p. 59-64.
138. Tian, S.-L., et al., *Mechanomyography is more sensitive than EMG in detecting age-related sarcopenia.* Journal of Biomechanics, 2010. **43**(3): p. 551-556.
139. Angeles, P., et al. *Automated assessment of symptom severity changes during deep brain stimulation (DBS) therapy for Parkinson's disease*. in *2017 International Conference on Rehabilitation Robotics (ICORR)*. 2017.
140. Silva, J., W. Heim, and T. Chau. *MMG-based classification of muscle activity for prosthesis control*. in *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 2004.

141. Hallett, E., et al. *Rapid bicycle gear switching based on physiological cues*. in *Automation Science and Engineering (CASE), 2015 IEEE International Conference on*. 2015. IEEE.
142. Woodward, R.B., et al., *Segmenting Mechanomyography Measures of Muscle Activity Phases Using Inertial Data.* Scientific reports, 2019. **9**(1): p. 5569.
143. Woodward, R.B., S.J. Shefelbine, and R. Vaidyanathan, *Pervasive monitoring of motion and muscle activation: inertial and mechanomyography fusion.* IEEE/ASME Transactions on Mechatronics, 2017. **22**(5): p. 2022-2033.
144. Kenney, L.P.J., et al., *Dimensional change in muscle as a control signal for powered upper limb prostheses: a pilot study.* Medical Engineering and Physics, 1999. **21**(8): p. 589-597.
145. Heath, G.H., *Control of proportional grasping using a myokinemetric signal.* Technology and Disability, 2003. **15**(2): p. 73-83.
146. Heath, G. and P. Bowker. *Myokinemetric control of a prosthetic prehensor from residual forearm musculature*. in *Proceedings of International Conference on Informatics and Control, St. Petersburg, Russia*. 1997.
147. Zheng, Y.P., et al., *Sonomyography: Monitoring morphological changes of forearm muscles in actions with the feasibility for the control of powered prosthesis.* Medical Engineering and Physics, 2006. **28**(5): p. 405-415.
148. Shi, J., et al., *Assessment of muscle fatigue using sonomyography: Muscle thickness change detected from ultrasound images.* Medical Engineering & Physics, 2007. **29**(4): p. 472-479.
149. Shi, J., et al., *Continuous Monitoring of Sonomyography, Electromyography and Torque Generated by Normal Upper Arm Muscles During Isometric Contraction: Sonomyography Assessment for Arm Muscles.* IEEE Transactions on Biomedical Engineering, 2008. **55**(3): p. 1191-1198.
150. Guo, J.-Y., et al., *Continuous monitoring of electromyography (EMG), mechanomyography (MMG), sonomyography (SMG) and torque output during ramp and step isometric contractions.* Medical Engineering & Physics, 2010. **32**(9): p. 1032-1042.
151. Chen, X., et al. *A novel approach for detection of muscle boundary in ultrasound images*. in *2011 4th International Conference on Biomedical Engineering and Informatics (BMEI)*. 2011.
152. Chen, X., et al., *Sonomyography (SMG) Control for Powered Prosthetic Hand: A Study with Normal Subjects.* Ultrasound in Medicine & Biology, 2010. **36**(7): p. 1076-1088.

153. Qian, C., et al. *A research of SMG controlled prosthetic hand with SSE2 acceleration*. in *2008 9th International Conference on Signal Processing*. 2008.
154. Youjia, H. and H. Liu. *Performances of surface EMG and Ultrasound signals in recognizing finger motion*. in *2016 9th International Conference on Human System Interactions (HSI)*. 2016.
155. Castellini, C. and D.S. Gonzalez. *Ultrasound imaging as a human-machine interface in a realistic scenario*. in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013.
156. Hettiarachchi, N., Z. Ju, and H. Liu. *A New Wearable Ultrasound Muscle Activity Sensing System for Dexterous Prosthetic Control*. in *2015 IEEE International Conference on Systems, Man, and Cybernetics*. 2015.
157. Li, Y., et al. *Human-machine interface based on multi-channel single-element ultrasound transducers: A preliminary study*. in *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*. 2016.
158. Hutchinson, T.E., et al., *Human-computer interaction using eye-gaze input.* IEEE Transactions on Systems, Man, and Cybernetics, 1989. **19**(6): p. 1527-1534.
159. Duchowski, A.T., *A breadth-first survey of eye-tracking applications.* Behavior Research Methods, Instruments, & Computers, 2002. **34**(4): p. 455-470.
160. Kocejko, T. *Gaze controlled prosthetic arm with EMG and EEG input interface*. in *2017 21st European Microelectronics and Packaging Conference (EMPC) & Exhibition*. 2017.
161. Buckley, M., R. Vaidyanathan, and W. Mayol-Cuevas. *Sensor suites for assistive arm prosthetics*. in *2011 24th International Symposium on Computer-Based Medical Systems (CBMS)*. 2011.
162. Giordaniello, F., et al. *Megane Pro: Myo-electricity, visual and gaze tracking data acquisitions to improve hand prosthetics*. in *2017 International Conference on Rehabilitation Robotics (ICORR)*. 2017.
163. Cristanti, R.Y., et al. *Eye gaze tracking to operate android-based communication helper application*. in *2017 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC)*. 2017.
164. Awais, M., N. Badruddin, and M. Drieberg. *Automated eye blink detection and tracking using template matching*. in *2013 IEEE Student Conference on Research and Developement*. 2013.
165. Tunhua, W., et al. *Real-time non-intrusive eye tracking for human-computer interaction*. in *2010 5th International Conference on Computer Science & Education*. 2010.

166. Cheng, J., et al., *On the tip of my tongue: a non-invasive pressure-based tongue interface*, in *Proceedings of the 5th Augmented Human International Conference*. 2014, ACM: Kobe, Japan. p. 1-4.
167. Lund, M.E., et al. *Inductive tongue control of powered wheelchairs*. in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*. 2010.
168. Mace, M., et al., *Tongue in cheek: a novel concept in assistive human machine interface.* Journal of Assistive Technologies, 2009. **3**(3): p. 14-26.
169. Mamun, K.A., et al. *Multi-layer neural network classification of tongue movement ear pressure signal for human machine interface*. in *2010 13th International Conference on Computer and Information Technology (ICCIT)*. 2010.
170. Vaidyanathan, R., et al. *Human-machine interface for tele-robotic operation: mapping of tongue movements based on aural flow monitoring*. in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. 2004.
171. Owen, A.M., et al., *Detecting Awareness in the Vegetative State.* Science, 2006. **313**(5792): p. 1402.
172. Hochberg, L.R., et al., *Neuronal ensemble control of prosthetic devices by a human with tetraplegia.* Nature, 2006. **442**(7099): p. 164.
173. Hochberg, L.R., et al., *Reach and grasp by people with tetraplegia using a neurally controlled robotic arm.* Nature, 2012. **485**(7398): p. 372.
174. Ángel-López, J.P. and N. Arzola de la Peña. *Voice Controlled Prosthetic Hand with Predefined Grasps and Movements*. in *VII Latin American Congress on Biomedical Engineering CLAIB 2016, Bucaramanga, Santander, Colombia, October 26th -28th, 2016*. 2017. Singapore: Springer Singapore.
175. Oppus, C.M., et al. *Brain-computer interface and voice-controlled 3D printed prosthetic hand*. in *2016 IEEE Region 10 Conference (TENCON)*. 2016.
176. Asyali, M., et al., *Design and implementation of a voice-controlled prosthetic hand*. Vol. 19. 2011.
177. Huang, H., et al. *The Development on a New Biomechatronic Prosthetic Hand Based on Under-actuated Mechanism*. in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2006.
178. Angeles, P., et al., *Automated assessment of symptom severity changes during deep brain stimulation (DBS) therapy for Parkinson's disease.* IEEE Int Conf Rehabil Robot, 2017. **2017**: p. 1512-1517.
179. Zhou, H. and H. Hu, *Human motion tracking for rehabilitation—A survey.* Biomedical Signal Processing and Control, 2008. **3**(1): p. 1-18.

180. Abbasi-Kesbi, R., A. Nikfarjam, and H. Memarzadeh-Tehran, *A Patient-Centric Sensory System for In-Home Rehabilitation.* IEEE Sensors Journal, 2017. **17**(2): p. 524-533.
181. Heinz, E.A., et al. *Using Wearable Sensors for Real-Time Recognition Tasks in Games of Martial Arts - An Initial Experiment*. in *2006 IEEE Symposium on Computational Intelligence and Games*. 2006.
182. LaValle, S.M., et al. *Head tracking for the Oculus Rift*. in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014.
183. Wang, S., et al. *Keyframe based large-scale indoor localisation using geomagnetic field and motion pattern*. in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016.
184. Kallapur, A., I. Petersen, and S. Anavatti. *A robust gyroless attitude estimation scheme for a small fixed-wing unmanned aerial vehicle*. in *2009 7th Asian Control Conference*. 2009.
185. Liu, T. and C. Fan. *Visible-light wearable eye gaze tracking by gradients-based eye center location and head movement compensation with IMU*. in *2018 IEEE International Conference on Consumer Electronics (ICCE)*. 2018.
186. Meyer-Hilberg, J. and T. Jacob, *High accuracy navigation and landing system using GPS/IMU system integration.* IEEE Aerospace and Electronic Systems Magazine, 1994. **9**(7): p. 11-17.
187. Wendel, J., et al., *An integrated GPS/MEMS-IMU navigation system for an autonomous helicopter.* Aerospace Science and Technology, 2006. **10**(6): p. 527-533.
188. Liu, Y., et al., *An innovative information fusion method with adaptive Kalman filter for integrated INS/GPS navigation of autonomous vehicles.* Mechanical Systems and Signal Processing, 2018. **100**: p. 605-616.
189. Fauser, T., S. Bruder, and A. El-Osery. *A comparison of inertial-based navigation algorithms for a low-cost indoor mobile robot*. in *2017 12th International Conference on Computer Science and Education (ICCSE)*. 2017.
190. Odry, Á., et al., *Kalman filter for mobile-robot attitude estimation: Novel optimized and adaptive solutions.* Mechanical Systems and Signal Processing, 2018. **110**: p. 569-589.
191. Barshan, B. and H.F. Durrant-Whyte, *Inertial navigation systems for mobile robots.* IEEE Transactions on Robotics and Automation, 1995. **11**(3): p. 328-342.
192. Bortz, J., *A New Mathematical Formulation for Strapdown Inertial Navigation.* IEEE Transactions on Aerospace and Electronic Systems, 1971. **AES-7**(1): p. 61-66.

193. Yean, S., et al., *Smartphone Orientation Estimation Algorithm Combining Kalman Filter With Gradient Descent.* IEEE J Biomed Health Inform, 2018. **22**(5): p. 1421-1433.
194. Yoo, T.S., et al., *Gain-scheduled complementary filter design for a MEMS based attitude and heading reference system.* Sensors (Basel), 2011. **11**(4): p. 3816-30.
195. Kalman, R.E., *A New Approach to Linear Filtering and Prediction Problems.* Journal of Basic Engineering, 1960. **82**(1): p. 35-45.
196. Sabatini, A.M., *Quaternion-based extended Kalman filter for determining orientation by inertial and magnetic sensing.* IEEE Trans Biomed Eng, 2006. **53**(7): p. 1346-56.
197. Marins, J.L., et al. *An extended Kalman filter for quaternion-based orientation estimation using MARG sensors*. in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*. 2001.
198. *Product Manual for use with InertiaCube4 Serial and USB Interfaces*. 2012, InterSense: Billerica, MA.
199. *VN-100 User Manual*. 2017, VectorNav Technologies: Dallas, TX, USA.
200. *MTi User Manual*. 2018, Xsens Technologies B.V. : Pantheon 6a, The Netherlands.
201. *3DM®-GX5-35 Attitude and Heading Reference System (AHRS) with GNSS*. 2017, MicroStrain® Sensing Systems: Williston, VT 05495, United States of America.
202. *User Manual TRAX*. 2014, PNI Sensor Corporation: Santa Rosa, CA, USA.
203. Madgwick, S.O.H., A.J.L. Harrison, and R. Vaidyanathan. *Estimation of IMU and MARG orientation using a gradient descent algorithm*. in *Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on*. 2011.
204. Mahony, R., T. Hamel, and J. Pflimlin. *Complementary filter design on the special orthogonal group SO(3)*. in *Proceedings of the 44th IEEE Conference on Decision and Control*. 2005.
205. Valenti, R.G., I. Dryanovski, and J. Xiao, *Keeping a Good Attitude: A Quaternion-Based Orientation Filter for IMUs and MARGs.* Sensors (Basel), 2015. **15**(8): p. 19302-30.
206. Cavallo, A., et al., *Experimental Comparison of Sensor Fusion Algorithms for Attitude Estimation.* IFAC Proceedings Volumes, 2014. **47**(3): p. 7585-7591.
207. Mourcou, Q., et al., *Performance Evaluation of Smartphone Inertial Sensors Measurement for Range of Motion.* Sensors (Basel), 2015. **15**(9): p. 23168-87.
208. Admiraal, M., S. Wilson, and R. Vaidyanathan. *Improved formulation of the IMU and MARG orientation gradient descent algorithm for motion tracking in human-*

machine interfaces. in *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2017 IEEE International Conference on*. 2017. IEEE.
209. Yi, C., et al., *Estimating Three-Dimensional Body Orientation Based on an Improved Complementary Filter for Human Motion Tracking.* Sensors (Basel), 2018. **18**(11): p. 3765.
210. Fan, B., Q. Li, and T. Liu, *How magnetic disturbance influences the attitude and heading in magnetic and inertial sensor-based orientation estimation.* Sensors, 2018. **18**(1): p. 76.
211. Lorusso, A., D.W. Eggert, and R.B. Fisher, *A comparison of four algorithms for estimating 3-D rigid transformations*. 1995: University of Edinburgh, Department of Artificial Intelligence.
212. Hamilton, W.R., *Ii. on quaternions; or on a new system of imaginaries in algebra.* The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 1844. **25**(163): p. 10-13.
213. Diebel, J., *Representing attitude: Euler angles, unit quaternions, and rotation vectors.* Matrix, 2006. **58**(15-16): p. 1-35.
214. Shepperd, S.W., *Quaternion from rotation matrix.[four-parameter representation of coordinate transformation matrix].* 1978.

Appendix I

Description of Machine Learning Algorithms used

K-Nearest Neighbours (KNN)

KNNs are the simplest of the algorithms described here. In practical terms, they are also functionally the most similar to the template-based algorithm described in the previous section. KNNs work by assuming that each array of features that describe a signal can be represented by a single point in N-dimensional Euclidean space, where N is defined by the number of features being considered. The classifier can be trained by defining each member of the training set t as a point in N-D space, leading to M points for a training set of length M. Each point within that set can be addressed as t_m where $m = 1,2,\dots,M$. To classify a new data point (s), straight lines can be constructed between s and each instance of t_m, where the magnitude of the line is the Euclidean Distance (d) between the two points, given by:

$$d_m(\bar{s},\bar{t}) = \sqrt{\sum_{i=1}^{N}\left(s_i - t_{m_i}\right)^2} \tag{68}$$

The prediction of a KNN identifies the K values of d_m with the lowest magnitude, and the classification is performed by finding the modal class of the corresponding t_m values.

Since each member of the training class is preserved, this is less memory efficient than using the template-based method described above, however it has the potential to be more accurate since data is not lost through generalizing the templates.

The KNN used in this evaluation used the value $k = 10$, and each of the neighbours were weighted equally.

Decision Trees

Decision trees perform classification by performing queries on the data, where each subsequent query is dependent on the answer to the previous one. Each query can be thought of as a node, the algorithm contains a single-entry point, known as a root node. The result from a query will cause the algorithm to move down one of several possible branches to the next node, finally reaching a leaf node, which contains the classifier prediction. Each node in the tree (other than the leaf nodes) separate data by checking to see if a condition is met. As a result, each node leads to two branches.

The size of a decision tree is important when implementing for real time use. Larger trees will be more suited to classifying large, complex data sets, but they will also be more computationally expensive. The size can be controlled by limiting the growth of the tree, either by limiting the number of layers (nodes equidistant to the root node), or the number of branches within a layer. In this implementation, the number of branches within a layer was limited to 20.

The layout of the decision tree used in this implementation was constructed according to the Gini diversity index, where every possible split for the data is assessed to find the one that gives the best split for the data. This is performed at every branch node to construct the tree.

Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a two-stage method for classifying categorical data. The first stage is a process of dimensionality reduction, which is implemented to maintain class separation while reducing the number of dimensions in which the classes are expressed. This is achieved by projecting the data into a lower dimensional feature space that maximizes the distance between the mean values of the classes, while simultaneously minimizing the variance of the data within each class. As a result, this method of dimensionality reduction makes three assumptions regarding the input data:

1. The input data represents a Gaussian distribution.
2. The variance is the same for each attribute.
3. Each class has a unique mean value

In practice, this means that data should be prepared to ensure that it conforms to these assumptions. This includes removing outliers that could potentially affect these statistical characteristics.

The second stage of the LDA implementation is the classification. This is often performed by estimating the probability that the new data belongs in each class, with the most probably outcome being the systems prediction. The implementation used here is based on Bayes' Theorem. For a classification problem of K classes, the prior probability of an observation X belongs to k class is given as $P(k)$. Bayes' Theorem uses this, along with the density function describing the distribution of data within each k class (denoted as $f_k(x)$), to estimate the posterior probability $P_k(X)$ that the observation is a member of class k, according to:

$$P_k(X) = \frac{P(k)f_k(x)}{\sum_{j=1}^{K} P(j)f_j(x)} \tag{69}$$

LDA creates linear discriminant boundaries between the classes, hence the second assumption. To cope with data sets where this is not the case, LDA was extended by allowing quadratic decision boundaries. This extension is known as Quadratic Discriminant Analysis QDA, and it allows for the quadratic boundaries that are created when the variance is not consistent between classes. Due to the increased amount of data required for QDA, only LDA was tested.

Support Vector Machines

Support Vector Machines (SVM) work by creating hyperplanes to separate different classes of data. A hyperplane is defined as being a geometric object with a dimensionality that is one less than the dimensionality of the data space. By this definition, a hyperplane can be used to separate an N-dimensional space into two N-dimensional subspaces. SVMs are a generalization of an optimally separating hyperplane that can be used to separate two groups within the dataset. In practice, the SVM allows for some limited misclassification on the training data to construct a plane that can be described linearly (quadratic and cubic hyperplanes can also be used for complex problems). SVM are typically used to solve two class problems, and the hyperplane is constructed to leave the greatest margin between the

two classes. The data points closest to the hyperplane are referred to as support vectors, and the margin is calculated by evaluating the sum of the distance between these support vectors and the hyperplane.

Since SVMs are binary classifiers, a single SVM is not suited to a multiclass problem, such as that presented here. There are two commonly used methods for using multiple SVMs to perform classification, one vs all and one vs one. In the one vs all approach, one SVM is trained for each class in the problem (K classes will require K classifiers). This SVM will identify whether the data belongs in its class, giving a confidence score. The class with the highest confidence score is taken as the system's prediction for the class. The one vs one approach creates a classifier for each possible pair of classes, which is trained to distinguish between two of the possible classes. For K classes this requires $K(K-1)/2$ classifiers. The prediction for the system is the class that got the highest number of positive classifications. In this implementation, a one vs one approach was used.

SVMs have several benefits when compared to LDA, but the most significant is that it does not make any assumptions regarding the incoming data. SVMs are not susceptible to outliers, since they only make use of the points defined as support vectors. As a result, they do not require signal conditioning, which could be challenging in real time implementations.

Appendix II

Description of Features used when processing MMG data

Root Mean Square

For a dataset d of length (N), the Root Mean Square (RMS) is given by:

$$RMS = \sqrt{\frac{1}{N}\sum_{n=1}^{N} {d_n}^2} \tag{70}$$

Integral of Absolute Value

The Integral of Absolute Value (IAV) is given by:

$$IAV = \frac{1}{N}\sum_{n=1}^{N} d_n \tag{71}$$

Mean Absolute Value

The Mean Absolute Value (MAV) is given by:

$$MAV = \frac{1}{N}\sum_{n=1}^{N} |d_n| \tag{72}$$

Modified Mean Absolute Value 1

The Modified Mean Absolute Value 1 ($MAV1$) is a weighted version of the MAV. $MAV1$ attempts to increase the robustness of the signal by reducing the weighting at the beginning and the end of the signal.

$MAV1$ is implemented as follows:

$$MAV1 = \frac{1}{N}\sum_{n=1}^{N} w_n |d_n| \tag{73}$$

$$w_n = \begin{cases} 1, & if\ 0.25N \le n \le 0.75N \\ 0.5, & otherwise \end{cases}$$

Modified Mean Absolute Value 2

The Modified Mean Absolute Value 2 ($MAV2$) is an extension that smooths the weighting function. It is implemented as follows:

$$MAV2 = \frac{1}{N}\sum_{n=1}^{N} w_n |d_n|$$

$$w_n = \begin{cases} 1, & if\ 0.25N \leq n \leq 0.75N \\ 4n/N & if\ 0.25N > n \\ 4(N-n)/N & if\ 0.75N < n \end{cases} \quad (74)$$

Simple Square Integral

The Simple Square Integral (SSI) is given by:

$$SSI = \sum_{n=1}^{N} |d_n|^2 \quad (75)$$

Variance

The Variance (VAR) of the signal is the mean of the square of the difference from the MAV. In most EMG systems, mean of the signals are approximately zero, and so the definition is normally abbreviated. The MMG signal are low pass filtered, however the classification is being performed on a section of a low frequency gesture. As a result, the average cannot be assumed to be zero, so the term is defined as:

$$VAR = \frac{1}{N-1}\sum_{n=1}^{N} (|d_n| - IAV)^2 \quad (76)$$

Absolute Value of the 3rd, 4th and 5th Temporal Moment

The Absolute Value of the 3rd, 4th and 5th Temporal Moment ($TM3$, $TM4$ and $TM5$) have been used in the statistical analysis of EMG signals. The terms are defined as:

$$TM3 = \left| \frac{1}{N}\sum_{n=1}^{N} {d_n}^3 \right|$$

$$TM4 = \left| \frac{1}{N}\sum_{n=1}^{N} {d_n}^4 \right| \quad (77)$$

$$TM5 = \left| \frac{1}{N} \sum_{n=1}^{N} {d_n}^5 \right|$$

Difference Absolute Mean Value

The Difference Absolute Mean Value ($DAMV$) is given by:

$$DAMV = \frac{1}{N-1} \sum_{n=1}^{N-1} |d_{n+1} - d_n| \tag{78}$$

Difference Absolute Standard Deviation Value

The Difference Absolute Standard Deviation Value ($DASDV$) is given by:

$$DASDV = \sqrt{\frac{1}{N-1} \sum_{n=1}^{N-1} |d_{n+1} - d_n|^2} \tag{79}$$

Appendix III

Description of Kuka kinematics used in VR visualisation

The axes of rotation of the simulated robot are shown in Table 31.

Table 31 – Summary of Denavit Hartenberg parameters of virtual Kuka robot

$\boldsymbol{i}$	$\boldsymbol{a_{i-1}}$	$\boldsymbol{\alpha_{i-1}}$	$\boldsymbol{d_i}$	$\boldsymbol{\theta_i}$
1	0	0°	d_1	θ_1
2	0	-90°	d_2	θ_2
3	0	90°	d_3	θ_3
4	0	-90°	d_4	θ_4
5	0	90°	d_5	θ_5
6	0	-90°	d_6	θ_6
7	0	90°	d_7	θ_7

Using this, the kinematics have been defined using the Denavit Hartenberg method, allowing the relationships between joints to be summarized, as in Table 31.

The definition for these parameters is:

$a_i = the\ distance\ from\ \hat{Z}_i\ to\ \hat{Z}_{i+1}\ measured\ along\ \hat{X}_i$

$\alpha_i = the\ angle\ between\ \hat{Z}_i\ to\ \hat{Z}_{i+1}\ measured\ about\ \hat{X}_i$

$d_i = the\ distance\ from\ \hat{X}_{i-1}\ to\ \hat{X}_i\ measured\ along\ \hat{Z}_i$

$\theta_i = the\ angle\ between\ \hat{X}_{i-1}\ to\ \hat{X}_i\ measured\ about\ \hat{Z}_i$

The rotation matrix around each joint is given by:

$$T_i^{i-1} = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i \cos\alpha_{i-1} & \cos\theta_i \cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1}\, d_i \\ \sin\theta_i \sin\alpha_{i-1} & \cos\theta_i \sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1}\, d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (80)$$

The orientation and position of the end effector in the base frame is given by:

$$T_7^0 = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 T_7^6 \quad (81)$$

The angles required to move the end effector to a given position and orientation can be calculated through the inverse kinematics of the Kuka simulation, however that was not required for the experiments defined here. Instead, a number of specifications were defined based on the intended use during the experiment.

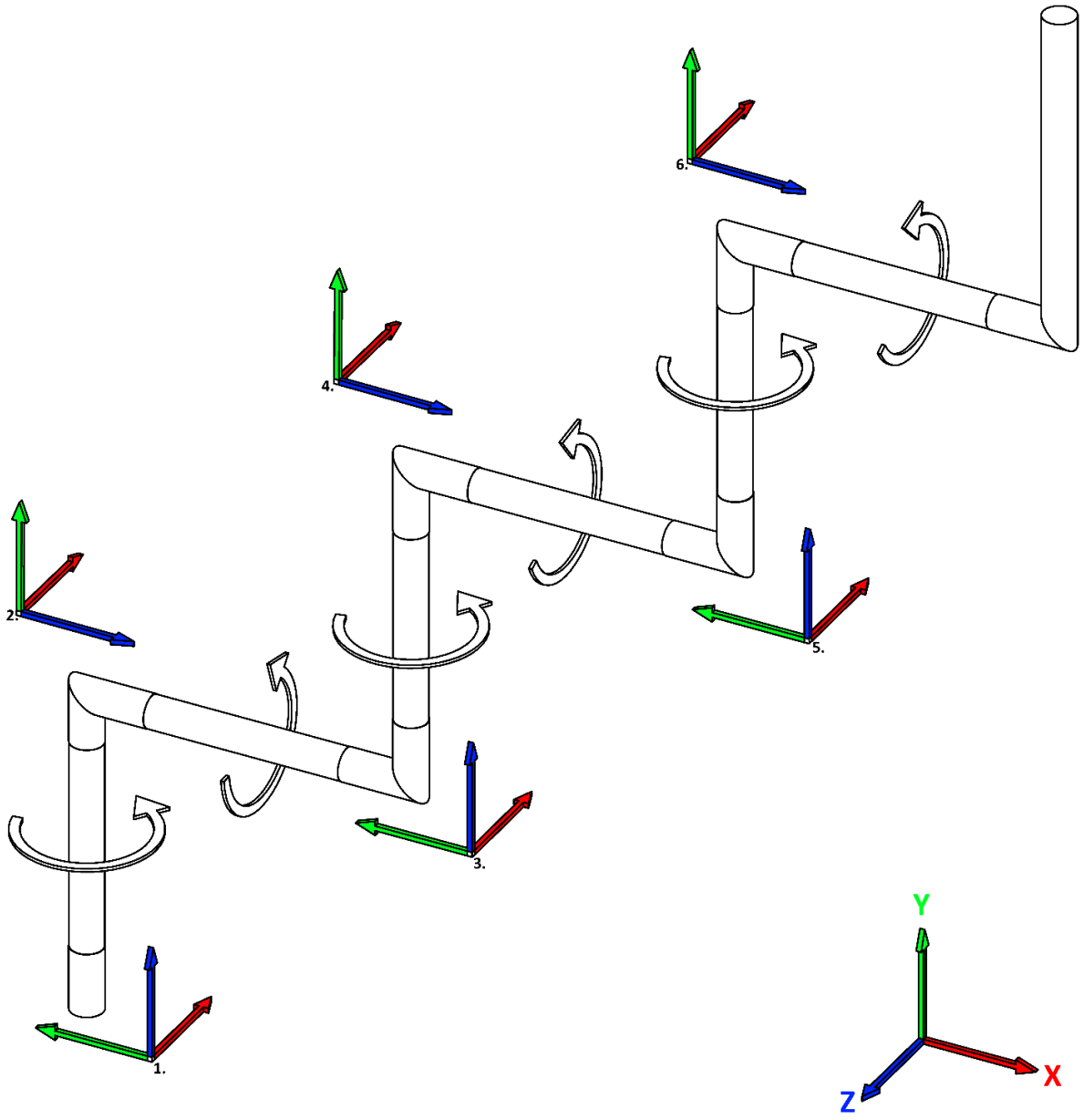

Figure 36 – Axis of rotations of virtual Kuka robot

Those specifications were as follows:

Specification_1 - The end effector was only required to be in one orientation, which is parallel to the Z axis.

Specification_2 - There are no obstacles in the way of the arm, thus an elbow up solution is appropriate in all situations.

Specification_3 - $l_2 = l_4 \; and \; l_3 = l_5$

Specification_4 - $\theta_3 = \pi$

To simplify descriptions, the end point of each joint (i) will be defined according as $P_{i_{xyz}}$. The target position can be defined at tar_{xyz}.

These specifications allowed the derivations of joint angles to be expressed as a minimization problem by defining the error (err) as the deference between tar_{xyz} and $P_{7_{xyz}}$, therefore the minimum value of err is given by:

$$min\left(tar_{xyz} - P_{7_{xyz}}\right) \tag{82}$$

$P_{7_{xyz}}$ can be calculated by taking the position vector from T_7^0, which is dependent on θ_{123456}.

Based on **Specification_1**, the point between $P_{6_{xyz}}$ can be expressed by:

$$P_{6_{xyz}} = \; [P_{7_x}, P_{7_y}, P_{7_z} - l_7] \tag{83}$$

The global orientation of joint 6 (θ_{J6}) is dependent on the previous five joints, and is not current known, As a result, the position $P_{5_{xyz}}$ cannot be determined at this point, however since $\alpha_6 = \frac{\pi}{2}$, the position of $P_{5_{xyz}}$ can be expressed in terms of θ_{J6}, as follows:

$$P_{5_{xyz}} = \; [P_{7_x} - l_6 * \sin(\theta_{J6}), P_{7_y} - l_6 * \cos(\theta_{J6}), P_{7_z} - l_7] \tag{84}$$

If the position $P_{5_{xyz}}$were known, then θ_1, θ_2 and θ_4 can be calculated. Based on **Specification_4**, l_2 and l_4 are parallel and in opposite directions, and therefore cancel each other out.

$$\theta_1 = atan2(P_{5_z}, P_{5_z}) \tag{85}$$

$P_{0_{xyz}}$ is located at the origin. Based on the configuration of the robot, $P_{1_{xyz}}$ is always located at $[0, l_1, 0]$.

A line (l_c) between $P_{1_{xyz}}$ and $P_{5_{xyz}}$ is defined as:

$$l_c = \left|P_{1_{xyz}} - P_{5_{xyz}}\right| \tag{86}$$

A second line (l_d) between $P_{0_{xyz}}$ and $P_{5_{xyz}}$ is defined as:

$$l_d = \left| P_{0_{xyz}} - P_{5_{xyz}} \right| \quad (87)$$

These lines can be seen in Figure 37.

Based on this model, θ_2 and θ_4 can be calculated according to:

$$\theta_2 = \pi - \theta_B - \theta_E \quad (88)$$

$$\theta_4 = \pi - \theta_A \quad (89)$$

Where, using the cosine rule:

$$\theta_A = \cos^{-1}\left(\frac{l_5^2 + l_3^2 - l_C^2}{2 * l_5 * l_3}\right) \quad (90)$$

$$\theta_B = \cos^{-1}\left(\frac{l_3^2 + l_C^2 - l_5^2}{2 * l_3 * l_C}\right) \quad (91)$$

$$\theta_E = \cos^{-1}\left(\frac{l_1^2 + l_C^2 - l_D^2}{2 * l_1 * l_C}\right) \quad (92)$$

The rotation of θ_5 must be set so that joint 6 lies on a plane parallel to the XY plane. As a result, if $l_1 = l_2 = l_3 = l_4 = l_5 = 0$, then this plane will be defined by Z = 0.

Solving the forward kinematics for T_5^0 with θ_5 as an unknown, and equating the Z component to 0 yields:

$$a \sin\theta_5 + b \cos\theta_5 = 0 \quad (93)$$

Where
$$a = -\cos(\theta_1)\cos(\theta_4)\sin(\theta_3) + \sin(\theta_1)\sin(\theta_2)\sin(\theta_4) - \cos(\theta_2)\cos(\theta_3)\cos(\theta_4)\sin(\theta_1) \quad (94)$$

And
$$b = \cos(\theta_1)\cos(\theta_3) - \cos(\theta_2)\,sin(\theta_1)\sin(\theta_3) \quad (95)$$

Rearranging for θ_5 gives:

$$\theta_5 = 2\tan^{-1}\left(\frac{a - \sqrt{a^2 + b^2}}{b} + \pi n\right) \quad (96)$$

The final variable θ_6 can be calculated by evaluating T_6^0. In this case, the Y axis can be set to 0, and the l_6 can be set to 0. Solving as before:

$$a \sin\theta_6 + b \cos\theta_6 = 0 \quad (97)$$

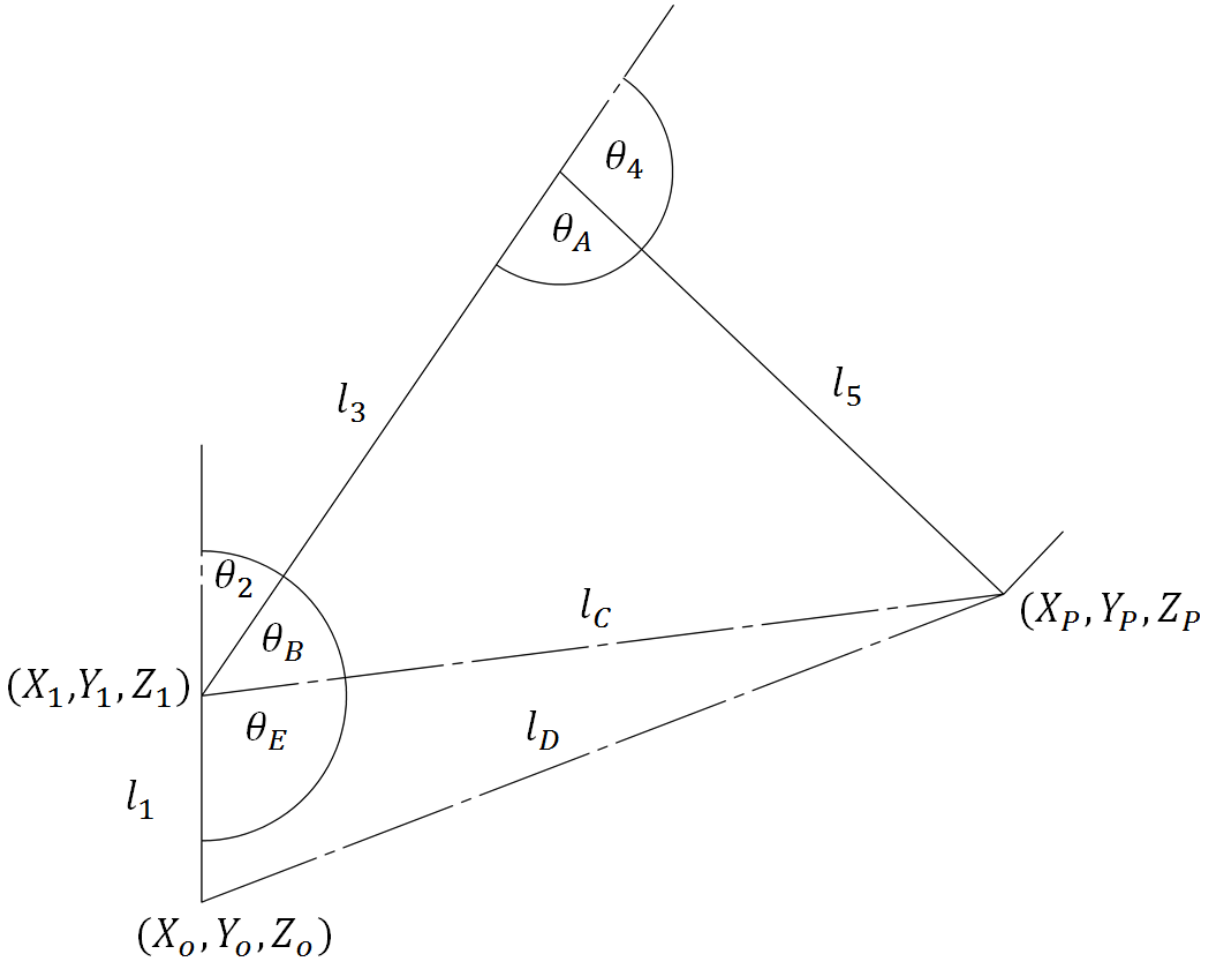

Figure 37 – Virtual kinematic model for solving elbow up inverse kinematics

Where a and b are functions of θ_{12345}.

The error can then be calculated by solving T_7^0 to find $P_{7_{xyz}}$, however this method requires an estimation for $P_{5_{xyz}}$. The implementation of this minimization requires an iterative process, if $P_{5_{xyz}}$ is set to tar_{xyz}, err_1 can be found. The second estimation for $P_{5_{xyz}}$ of $tar_{xyz} + err_1$ allows a new error value (err_2) to be generated, and a third estimation of $tar_{xyz} + err_1 + err_2$ can be made. The error values will tend to 0, and therefore the system will tend towards the correct position. A threshold for accuracy can be used to decrease the required iterations.