Action Tracer with Android API

An investigation into real-time athelete feedback systems that help them improve and injury prevention.

University of Cape Town

Presented by:
Humphrey Chiramba

Supervisor
Dr Simon Winberg
Department of Electrical Engineering
University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town in fulfilment of a Master of Science in Engineering in Electrical Engineering.

Submitted on
December 8, 2023

The copyright of this thesis vests in the author. No quotation from it or information derived from it is to be published without full acknowledgement of the source. The thesis is to be used for private study or non-commercial research purposes only.

Published by the University of Cape Town (UCT) in terms of the non-exclusive license granted to UCT by the author.

University of Cape Town

Declaration

I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own. This thesis/dissertation has been submitted to the Turnitin module (or equivalent similarity and originality checking software) and I confirm that my supervisor has seen my report and any concerns revealed by such have been resolved with my supervisor.

Signature **H. T. Chiramba**

Date: December 8, 2023

Acknowledgements

I'd like to firstly thank my supervisor, Associate Professor Winberg for his guidance and help in working on this dissertation. I also thank my interviewees for their time and contribution to this dissertation. Alongside them, the many lecturers, staff, and lab mates that I have worked with and chatted to about this project are greatly appreciated.

This dissertation would not have been possible without the support from my parents, Baba naMai Chiramba. I appreciate your help in getting me to this point in my life, and aim to turn what once were sacrifices for me into joy for you! To Court and Hamilton, I think of you both all the time and wish the same joys and successes, and thank you for your prayers and input.

To the many friends who supported me, I am eternally grateful. From encouragement to meals to places to stay, I have felt at home in Cape Town and have not only loved the city by its people. This dissertation would be nowhere near complete without you all. A special thanks to Paul for the coffees and for telling me to "lug nut up" back in 2020. And Mikayla, I'd like to thank you for the late nights, lunches, rides and encouragement, may I also follow your example in future. Other amazing people that helped in this dissertation are Shingirai, Justin and Lukas, who gave me tools, ideas, and friendships.

Finally, I give glory to God for this dissertation and the time I've been at the university. He has allowed me a time to learn, explore and enjoy life. He has kept me, fed me and made me strong where I felt weak and is the knower of my deepest desires.

May all that I do, forever bring glory to You!

Abstract

This dissertation aims to present the design and testing of a low-cost motion capture device with feedback capabilities. In addition to this, an API to communicate with this system is developed that allows devices to connect to the motion capture system and execute commands on it.

This system, Action Tracer, would be useful in cases where a user may need to train or learn a repetitive motion found in an activity or sport. Such a task needs to be done under supervision and may need to be done away from a coach or therapist. In either situation, a system which can accommodate both cases while providing insights that are not visible to the eye is needed. In addition to this, the system would need to be configurable for the various sports that it may be used with, as well as be operated from an Android device.

Most of the work that will be presented in this report will relate to the development of marker based motion capture systems, with a particular interest in the use of Inertial Measurement Units (IMUs). In addition to this, the subject of Biofeedback is be covered, as well as how it relates to motion capture and how it can be applied in this field.

In order to build this system, first various figures of merit were obtained from multiple motion capture systems which are in use by way of exploring literature in the field. This is followed by expanding the obtained requirements into design specifications for the subsystems that will be identified as essential. These specifications formed the foundational building blocks for the system, and they will be put together to form the entire system.

In order to evaluate this system was compared to a markerless camera based motion capture system by observing joint rotations at the elbow and wrist. This lead to analysing the profiles obtained from multiple participants, and will be followed by analysing the variations in the motions performed for each system. The analysis will lead to obtaining an accuracy, precision, and range for the developed system.

Finally, the results of the comparisons between these two systems will be provided and showed that the developed system can keep, on average, within 5% of the markerless system with a precision of 10%.

Nomenclature

Acronyms

AC	Alternating Current
AM	Amplitude Modulation
API	Application Programming Interface
CAN	Controller Area Network bus
DC	Direct Current
dps	degrees per second
FM	Frequency Modulation
FSR	Full Scale Range
g	gravitational constant ($9.81 m/s^2$)
GUI	Graphical User Interface
IEEE	Institute of Electrical and Electronics Engineers
I2C	Inter-Integrated Circuit
IP	Internet Protocol
JSON	JavaScript Object Notation
LED	Light Emitting Diode
MSE	Mean Squared Error
OOD	Object Oriented Design
PCB	Printed Circuit Board
SDK	Software Development Kit
SPI	Serial Peripheral Interface
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver/Transmitter
UML	Universal Modelling Language
USART	Universal Synchronous/Asynchronous Receiver/Transmitter
Wi-Fi	Wireless Fidelity (IEEE802.11)

Terminology

client/user a person or entity utilising a specific resource, application, or system

design pattern design level solutions for common recurring problems during software development

developer/programmer a person or entity building upon a given resource, application, or system

drift small variations in readings when, under identical conditions, at different points in time

embedded platform a device with a microprocessor or microcontroller which is used to execute programs and connect to peripheral devices.

firmware software running on an embedded platform

high-level relating to an abstraction that is very high, leading to much more understandable concepts and syntax (more human-readable)

low-level relating to an abstraction that is very low, leading to more complex and less involved concepts and syntax (less human-readable)

marker An object/device used to locate and keep track of the area it is placed

off the shelf an item that is made by another party that is not made to fit a particular use case.

pitch (θ) rotation about an axis that is going from left to right

Pi a Raspberry Pi board

protobuf Protocol Buffer

roll (ϕ) rotation about an axis that is going from front to back

serialization the process of packaging a data structure into a logical order for uses in a program such as storage or transmission.

spacial relating to position

temporal relating to time

thread (thread of execution) a small sequence of programmed instructions that can be managed and executed by a scheduler on an operating system.

yaw (ψ) rotation about an axis that is going from top to bottom

Contents

List of Figures **i**

List of Tables **viii**

1 Introduction **1**

1.1 Background 1

1.2 Important Terminology 3

1.3 Research Problem 3

1.4 Research Focus and Questions 4

1.5 Significance of the project 4

1.6 Practical applications for the expected results 5

1.7 Scope and limitations 5

2 Literature Review **7**

2.1 Motion capture 8

2.1.1 Markerless Motion Capture 9

2.1.2 Marker Motion Capture 9

2.1.3 Miscellaneous Motion Capture Methods 12

2.1.4 Summary 12

2.2 Inertial Measurement Units 13

2.2.1 Principles of operation 13
2.2.2 IMU Element Combinations 15
2.2.3 Summary 21
2.3 Biofeedback 21
2.3.1 Principles of biofeedback 22
2.3.2 Biofeedback in use 24
2.3.3 Summary 24
2.4 IMU Based Motion Capture Systems 24
2.4.1 Hardware in IMU based motion capture systems 28
2.4.2 Software in IMU based motion capture systems 31
2.4.3 Summary 33
2.5 Solutions of interest 34
2.5.1 Summary 36
2.6 Summary 36

3 Methodology **38**

3.1 Planning and Overview 39
3.1.1 Initial Research 39
3.1.2 Initial Literature Review 40
3.1.3 Research Proposal 41
3.2 Design and Theory 41
3.2.1 Detailed Literature Review 42
3.2.2 Requirements Analysis and Subsystem Identification 42
3.2.3 Subsystem Design and Testing 44
3.3 Implementation 45

3.3.1 Informed design refinement 45
3.3.2 Experimentation 45
3.4 Documentation 46
3.4.1 Record and discuss results 47
3.4.2 Closing the project 47
3.5 Summary 47

4 System Design **48**

4.1 Requirements analysis 48
4.1.1 SR01: The system must be low cost. 49
4.1.2 SR02: The system must be able to capture human motion in real time. 49
4.1.3 SR03: The system must be usable from a distance. 50
4.1.4 SR04: Achieve a configurable system 50
4.1.5 Hardware and Software Requirements 50
4.2 System Break Down 51
4.2.1 Hardware Subsystem 51
4.2.2 Software Subsystem 52
4.2.3 Full System 53
4.3 Summary 54

5 Hardware Design **55**

5.1 Requirements analysis 55
5.1.1 HW01: The device must be cheap to develop 55
5.1.2 HW02: The design must be easy to reproduce 56
5.1.3 HW03: The device must be able to obtain rotation data 56

5.1.4 HW04: The device must be capable of being placed anywhere on the human body . . . 56
5.1.5 HW05: The device must be usable in sport and portable . . . 57
5.1.6 HW06: The device must be able to send data to another device . 57
5.1.7 HW07: The device must have a means to communicate to its user 57
5.1.8 HW08: The device must be capable of being used by multiple people 58
5.1.9 Hardware Specifications . . . 58
5.2 Component Selection . . . 58
5.2.1 IMU selection . . . 59
5.2.2 Embedded platform selection . . . 59
5.2.3 Feedback device selection . . . 60
5.2.4 Node design . . . 61
5.3 Proposed design . . . 61
5.4 Summary . . . 62

6 Software Design **63**

6.1 Requirements analysis . . . 63
6.1.1 SW01: Collected data must be retrievable from any device . . . 63
6.1.2 SW02: Data transmission should be seamless . . . 64
6.1.3 SW03: Developers must be able to build upon this system . . . 64
6.1.4 SW04: The system must allow for configurability . . . 65
6.1.5 Software Specifications . . . 65
6.2 Detailed software design . . . 65
6.2.1 Design Pattern . . . 66
6.2.2 Configurability . . . 67

6.2.3 Data Collection 69
6.2.4 Data Transmission 70
6.2.5 Additional Processing and Feedback 73
6.2.6 Detailed Software Design Summary 74
6.3 API design 74
6.3.1 Tracer Point API 75
6.3.2 Action Tracer Server API 77
6.4 Summary 79

7 Experimentation **81**

7.1 Initial Testing 82
7.1.1 Hardware 82
7.1.2 Software 84
7.2 System Integration Testing 87
7.2.1 API Testing 90
7.2.2 Angle Testing 90
7.2.3 Motion Capture Tests 90
7.3 Motion Capture Benchmarking 91
7.3.1 Wearability 94
7.3.2 Data Resolution (Output Data Rate) 95
7.3.3 Accuracy and Precision 95
7.3.4 Range and Network 96
7.3.5 Profile identification 96
7.3.6 Motion classification 97
7.4 Summary 97

8 Results **98**

8.1 Initial Testing 99

8.1.1 Hardware 99

8.1.2 Software 102

8.2 System integration tests 106

8.2.1 API Testing 106

8.2.2 Angle Testing 108

8.2.3 Motion Capture Tests 110

8.3 Motion Capture Benchmarking 111

8.3.1 Wearability 112

8.3.2 Resolution 113

8.3.3 Accuracy and Precision 114

8.3.4 Range and Occlusion 120

8.3.5 Profile Identification 124

8.3.6 Motion Classification 132

8.4 Summary 135

9 Conclusion **137**

9.1 Summary of the Dissertation 137

9.2 Evaluation of requirements 138

9.2.1 Review of requirements 138

9.2.2 Review of objectives 139

9.3 Future Work 141

9.3.1 True wireless motion capture 141

9.3.2 Further benchmarking 141

9.3.3 Integration of AI 141

Bibliography **143**

A Additional documents **150**

B Design and Implementation **154**

B.1 Software: Accessing Action Tracer's source code 154

B.2 Hardware 155

B.2.1 Schematics 155

B.3 Test circuitry 158

B.4 Machine Specifications 158

B.4.1 Raspberry Pi Zero W 158

B.4.2 Machine 1 158

B.4.3 Machine 2 158

B.4.4 Machine 3 158

B.5 Software: Accessing Action Tracer's source code 159

B.5.1 Sample Python code to find angles between two vectors 159

B.5.2 UML Class Diagram 159

List of Figures

1.1 A diagram depicting the various systems involved in capturing and processing data for an activity. 2

2.1 The key topics which will be covered in this literature review. 7

2.2 A brief overview of the different kinds of motion capture systems currently in use. 8

2.3 Optical markers used in motion capture over various stages. 10

2.4 A visual illustration to help define yaw, pitch and roll. 16

2.5 An illustration of the data obtained when using an IMU with an accelerometer and gyroscope. 16

2.6 An illustration of the data obtained when using an IMU with an accelerometer, gyroscope and magnetometer. 17

2.7 A flow diagram showing how the quaternion rotation data is obtained from raw accelerometer and gyroscope readings at time t 20

2.8 The 3 stages of the biofeedback loop in the context of a patient. 22

2.9 A simplified model of a basic motion capture system using a single controller and multiple sensors. 28

2.10 A simple model of a multi-controller motion capture system using multiple devices to capture motion data. 29

2.11 Sama et al.'s data packet structure used for a motion capture glove. . . . 31

2.12 An image summarising the architecture and operational principles of Li et al.'s system, ISWIM. 34

2.13 An image summarising the architecture and operational principles of Slade et al.'s system. 35

3.1 A short diagrammatic representation of the phases used in designing, building, and testing the Action Tracer system. 39

3.2 Boehm's spiral model, which was followed in the design and evaluation of Action Tracer's hardware and software. 40

3.3 The literature review funnel used to define the project's objectives. . . . 41

3.4 The core foundational principles of Object-Oriented Design in software. . 44

4.1 An overview of the key topics that will be covered in the system design chapter. 48

4.2 A UML use case diagram detailing the key features and functionalities of the Action Tracer system. 49

4.3 The four components that make up the hardware subsystem. 52

4.4 The three components that make up the software subsystem. 52

4.5 A UML deployment diagram of the entire Action Tracer system, showing the main subsystems and their components. 53

5.1 An overview of the key topics that will be covered in the hardware design chapter. 55

5.2 The MPU-6050 selected for the design of the Action Tracer system's node. 59

5.3 The Raspberry Pi Zero W selected for the Action Tracer system's processing node. 60

5.4 A complete overview of the proposed Action Tracer system shown as a block diagram with all the key components. 61

5.5 The complete Action Tracer development hardware system displayed. . . 62

5.6 A picture of 3 nodes and the Raspberry Pi being worn by a user to track the upper limb movement. 62

6.1 An overview of the key topics that will be covered in this design chapter. 63

6.2 A UML activity diagram showing how the software should function in order to successfully be considered as a valid solution. 66

6.3 A diagram showing the different objects that will be put together in the firmware, allowing Action Tracer to be functional. ** Denotes a class that will not be created by the author.* 67

6.4 UML class diagrams of some methods and variables offered by the i2cdevlib's MPU-6050 library. 68

6.5 An illustration of the running program thread and the tasks it will be associated with when running the system 69

6.6 A UML sequence diagram showing how the communication with n IMU nodes over I2C. 69

6.7 The packet structure of the data collected from the MPU-6050's DMP by way of the IMUs FIFO. 70

6.8 An illustration of the Action Tracer system functionality relative to the collection thread. 70

6.9 UML class diagrams showing the communication component's structure and interactions. 71

6.10 The Action Tracer Protobuf specification illustrated to show the different structures that make up the packet. 72

6.11 An illustration of the Action Tracer system functionality relative to the transmission thread. 73

6.12 An illustration of the Action Tracer system functionality relative to the feedback and processing that may be additionally used. 74

6.13 An illustration of the Action Tracer system as a whole, showcasing all the threads and memory space. 74

6.14 A tree diagram of the commands available in the Tracer Point class, grouped by functionality. 76

6.15 A tree diagram of the commands available in the Action Tracer Server class, grouped by functionality. 78

6.16 A diagram showing the commands used in addressing different parts of the body and Raspberry Pi pins based on the main node of the Action Tracer system. 79

7.1 An overview of the key topics that will be covered in this experimentation chapter alongside an index to the sections where each of the tests will be found. 81

7.2 The IMU validation experimental set up, which is the main HAT for the Action Tracer system. 82

7.3 Pictures showing the two experimental set-ups used for assessing the hardware. 7.3a showcases the exact setup used with regard to the turntable, while 7.3b showcases the use of the adjustable platform in these experiments. 83

7.4 The experimental set-up for the software used to run the Action Tracer system. 85

7.5 The basic data packet structure chosen to transmit data from the Action Tracer system. 85

7.6 Three screenshots from an application built using the Flutter framework used to test the functionality of the Action Tracer Server's API. 88

7.7 A screenshot of the main screen showing a model that will mirror a user's motion. 89

7.8 An illustration of how angles will be obtained from the Motion Mirror application. The angle α is obtainable using both 2D and 3D coordinate data. 89

7.9 The general layout of the lab tests conducted by the researcher on a secure network. 91

7.10 The primary motion to be done by participants for this study. This motion requires that participants move straight to pose 2 without pausing. . . . 92

7.11 The second motion participants are to do for this study. This motion requires that users first obtain pose 2 before proceeding to pose 3. 93

7.12 The general layout of the participant tests conducted by the researcher with cameras placed in various positions along a 2-meter arc. 94

7.13 A graph showing the required actions during an interview with a participant relative to time. The time's highlighted are those to be used for analysis. 94

8.1 An overview of how this results section will be laid out. This overview also shows which sections of this chapter will house specific results. 98

8.2 The results of the stationary IMU tests converted to yaw, pitch and roll values with $\alpha = 0.05$. 100

8.3 The results from the parallelism tests using a single IMU to capture and send 500 data packets over 16 runs. 102

8.4 A bar chart showcasing the average times taken by each of the filters to process data before transmission, with a filter window size of 20. 103

8.5 Results from the serialisation tests run to see if there were any benefits to the serialisation.8.5a compares the time taken to serialize the data, while 8.5b showcases the variations in the packet sizes over the same time period. 104

8.6 The results from the thread idle time tests. Figure 8.6a shows results from a section of the data from the collection thread, while Figure 8.6b shows the same instance in time for the transmission thread. In both cases, the average from the session is shown. 105

8.7 A graph showing the maximum frequencies that can be achieved with a given number of nodes. 106

8.8 Results from the initial angle tests alongside their respective errors. 8.8a showcases the first part of the test with an angular rate of, $+10°/s$ and 8.8b showcases the second part of the test with an angular rate of $-10°/s$. 109

8.9 Images captured showcasing the motion Action Tracer system being used to track upper left limb motions in the Motion Mirror application. 111

8.10 A compilation of images showing (from left) MediaPipe estimation, video recording and Action Tracer estimation. 112

8.11 A side by side view of the amount of data MediaPipe can capture when in frame. There are 23 landmarks in total, with the face having 11 of the landmarks. 114

8.12 A graph showcasing the rotations at the elbow joint as estimated by both the Action Tracer system and MediaPipe using both 2D and 3D coordinate data. 115

8.13 A graph showcasing the rotations at the elbow joint as estimated by both the Action Tracer system and MediaPipe using only the x and y coordinate data. 116

8.14 The error distribution between the Action Tracer and MediaPipe estimations over Participant 1's first motion. 117

8.15 A graph showing the variations in motion between Action Tracer (corrected in time) and MediaPipe over the course of a single repetition of Motion 1 for Participant 1 at the wrist. 119

8.16 A graph showing the variations in motion between Action Tracer (corrected in time) and MediaPipe over the course of a smaller portion of Motion 1 for Participant 1 presented in Figure 8.15. 119

8.17 The variations between distance and the number of packets obtained in the various tests done to assess the range of the Action Tracer system. These are shown in two sets, the first, unobstructed, then the second with at least a single obstruction. 121

8.18 Graphs showing the elbow angle obtained from Action Tracer, as well as from various cameras, individually used with MediaPipe. 123

8.19 Graphs showing the wrist angle obtained from Action Tracer, as well as from various cameras, individually used with MediaPipe. 124

8.20 A graph showcasing all three repetitions of Motion 1 as well a profile based on their average elbow rotations as estimated by Action Tracer. 125

8.21 A view of Participant 1's motion profile based on the elbow rotation obtained from the 2D data estimated by Action Tracer (8.21a) and MediaPipe (8.21b). 126

8.22 A graph showcasing two profiles being compared to each other. Both profiles were obtained from Participant 1's elbow(8.22a) and wrist(8.22b) by Action Tracer and MediaPipe while they were conducting Motion 1. . 128

8.23 A graph showcasing all participant's Motion 1 profiles according to elbow estimations collected from the Action Tracer system. 129

8.24 A graph showcasing all participant's Motion 1 profiles according to elbow estimations collected from MediaPipe. 130

8.25 Four separate MSE matrices for the various MSEs run to view the similarities between participant profiles. The scale ranges from red (highest MSE) to green (lowest MSE). 131

8.26 A graph showcasing the three motions and their profiles according to the MediaPipe estimations obtained at the elbow. 132

8.27 A graph showcasing the three motions and their profiles according to the Action Tracer estimations obtained at the elbow. 133

8.28 Six separate correlation matrices for the various correlations run to view the similarities between motion profiles. The scale ranges from red (highest MSE) to green (lowest MSE). 134

8.29 Two profile error analysis graphs obtained by comparing Motions 2 and 3 to Motion 1. Figure 8.29a shows this for Action Tracer, and Figure 8.29b shows the same for MediaPipe. 135

List of Tables

2.1 A brief comparison of some motion capture systems that are have been produced by various researchers. 27

2.2 The system requirements necessary to develop and build Action Tracer. . 37

3.1 The hardware requirements (HW-R) necessary to develop and build Action Tracer. 43

3.2 The software requirements (SW-R) necessary to develop and build Action Tracer. 43

4.1 The hardware and software requirements necessary to develop and build Action Tracer. 51

5.1 The hardware specifications (HW-SP) necessary to develop and build Action Tracer. 58

6.1 The software specifications (SR-SW-SP) necessary to develop and build Action Tracer. 65

8.1 A brief analysis of the errors obtained from the drift experiments, presented in degrees between the filter and the internal calculation's data. 101

8.2 The variations in time to serialise data, as well as in the size of the packets alongside the percentage increase relative to the non-serialised data. . . . 104

8.3 A brief analysis of the variations in idle and busy times for the transmission and collection threads that were used in this test. 105

8.4 Results from the various API commands for the Tracer Point API, grouped into their categories as is presented in Section 6.3.1.2. 107

8.5 Results from the various API commands for the Action Tracer Server API, grouped into their categories as is presented in Section 6.3.2.2. 107

8.6 The results from the angle tests between two IMUs, where one was stationary and the second was rotating, going upwards in steps of 45°. . . . 108

8.7 The results from the angle tests between two IMUs, where one was stationary and the second was rotating, going downwards in steps of 50°. . . 108

8.8 Particulars of the 3 participants that volunteered to assist with data collection for tests regarding profiling and the systems robustness 112

8.9 A table containing 10 records from the data packets received from the Action Tracer system while recording a participant's session. 113

8.10 An error analysis of the data collected from the three participants, recoded and analysed sessions based on Figure 8.14. 117

8.11 A table that presents all the cumulative packet delays on the data collected by Action Tracer. 122

9.1 The bill of materials required to reproduce the Action Tracer system. . . 138

B.1 An overview of the accelerometer FSRs from all the systems, presented in Table 2.1 154

B.2 An overview of the gyroscope FSRs from all the systems, presented in Table 2.1 154

Chapter 1

Introduction

1.1 Background

In the current era of technology, there has been a rise in the implementation of new technologies in all spheres of human activity. Since the industrial revolution, changes have been made to how humans conducted their work, leisure, transport and everything in between, in order to reduce the amount of effort, risk and strain needed to conduct these activities. In light of this, it is certain that the way sport is conducted is set to continue improving to benefit safety and accessibility.

Far and wide, sport has been used as a form of recreation for individuals looking to get physically and/or mentally active[1]. This is true of games that are played on the ground, in water and in the skies, there are no limits to the location of activity. In general, a participant needs to both be trained physically, to condition their bodies through or for the activity, and mentally to aid in preparation and strategy planning which will assist in giving an upper hand at winning for those who partake in competitive sport and safety and enjoyment for those not seeking competitive environments.

Alongside recreation, sport is used for rehabilitation. Within this sphere, it has been noted that certain movements within some activities may be vital to individuals regaining some of their finer mobility or relieving their pain. Examples of sports associated with this are swimming [2] or running [3]. It is therefore crucial that such treatment can be done and monitored sufficiently to allow the patient to recover effectively and correctly and not introduce further complications. This combination of rehabilitation and sport brings out the vital role of sport in society as a career, teacher and healing platform.

The commercialisation of sport has also been on the uptake, with greater strides being made by betting and gambling companies to be included in sport through various forms of sponsorship. Obtaining data from sports would be advantageous to allow their patrons

to make more informed decisions about the teams they support with studies such as that of Hing et al.[4]. Other places of need for this would be within the team's management to know how to value an athlete when it comes to costly business transactions such as player transfers and loans. Sport could therefore be considered a lucrative industry should it be approached from a financial standpoint for investors, shareholders, supporters and participants, with the latter being the key assets to this industry thriving.

Finally, with the larger reaches of technology, sport has become a form of enjoyment for viewers both in an arena and within their homes. With the various forms of data that can be collected, the viewing experience can be improved for fans, as they would feel more immersed and more included within an event. There are certain metrics that are available to high ranking officials in teams that are now passed on to opponents and observers, which will increase the intensity and enjoyment, bringing viewers closer to the event. For a sportsman, regular check-ups and assessments such as Functional Movement Screening, which was used by Teyhen et al. in injury prevention[5], can be used to help direct training blocks and schedule different activities.

With this in mind, a system is needed to monitor metrics such as the range of motion, joint angle, fitness, muscle contraction and adequately provide feedback to a coach or therapist, viewer and participant, where needed. This data must be accessible and able to be processed to allow the different parties involved to take it in accordingly. While many systems such as this are easy to construct, not many have the ability to view this data in real time. Furthermore, an even smaller number of these are able to do the above and allow instant feedback to the athlete being examined. Over and above these criteria, this system should be open source, such that it may be improved by other parties that may want to implement it or to use it. Figure 1.1 shows the different systems and parties involved as to the above-mentioned system.

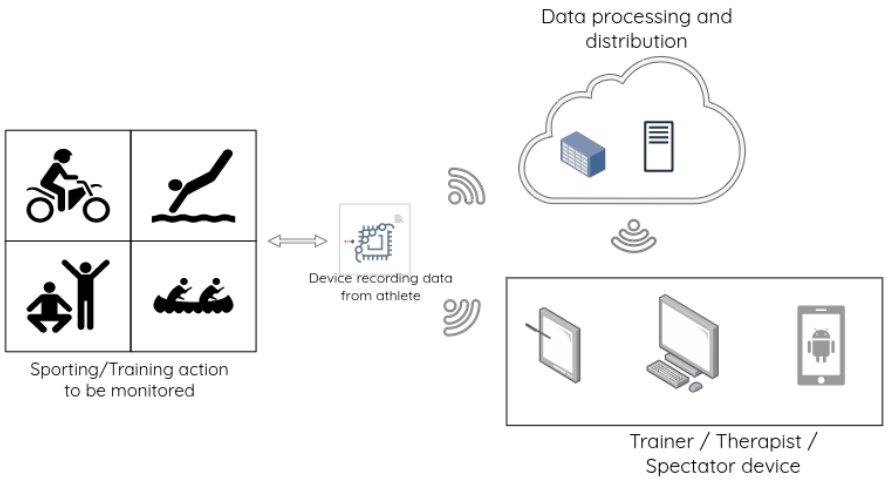


Figure 1.1: A diagram depicting the various systems involved in capturing and processing data for an activity.

1.2 Important Terminology

This section will present some key terms that will be used frequently in this dissertation. These terms will be central to the main aims and objectives of this project.

Application Programming Interface (API) This is often a set of commands that may be run on a host processor in order to simplify a larger set of commands to obtain data from a server or peripheral.

Biofeedback An amalgamation of the words biological and feedback. This term refers to the ability to provide feedback based on or relating to a human being.

Firmware Software that runs on a hardware device.

Inertial Measurement Unit (IMU) This is a device that, in most cases, contains an accelerometer, gyroscope, and magnetometer. It connects these three devices to a single communication channel that allows this unit to be a peripheral to a host device.

Motion Capture The act of recording the movement of objects or people. Often this is used to control a virtual object or person and is used in animation, sports science and various other fields.

1.3 Research Problem

With most major sporting teams and major rehabilitation centres having advanced methods of athlete and patient monitoring, there is a need for a similar intervention for smaller groups and practices. Currently, in the market are more expensive and proprietary alternatives that can not be purchased or easily used and maintained as there are costs in hiring or training personnel[6]. Most of these solutions rely on recorded data from sensors or video cameras that require post-processing on capable hardware, which may not always be available in some contexts such as schools or community clinics. There is valid reasoning to having such hardware in this context as it provides accuracy and reliability for users of such technology.

This dissertation aims to provide an architecture for a system that will be more affordable for both end users and clinics as well as reliable enough to extract valuable information for practitioners, trainers, and observers. It should be a versatile system allowing for customisation, such that it may be used for a wide array of activities.

1.4 Research Focus and Questions

The main areas of interest in this project are to do with collecting useful data from a network of sensors by way of a hybrid network and various endpoints. The data will be processed both in the sensor device and at the gateway that will deliver the information to a monitoring device to display this information in a digestible manner to a user. This process may either be direct or via a cloud server that will do some data processing to alleviate the end device used by a therapist or coach from any intense processing needed.

This project will look at the different technologies that could be used to achieve this and how they will affect the data availability, transmission, and processing. With the vast amount of devices available for this, the initial focus will be on data availability. Once this has been determined, focus will be placed on data transmission and processing.

The main points of improvement will revolve around the rate at which data may be transmitted and interpreted for a supervisor. There will also be emphasis on getting as much data as is needed to allow the cost of the system to be as cheap as possible. Finally, the customisability of the system will be investigated, addressing questions around different use cases in various sports and activities.

Some questions that will be answered will include:

1. Can a system be designed to help users learn a sport/activity?
2. Can the data collected by such a system be used in assessing an athlete's risk of injury?
3. Can this system interface with other commercial devices and protocols?
4. Will such a system be sufficient for all sports, or would it be better suited to specific categories of activities?

The notion of assessing an athlete's risk of injury using such a system was highlighted by Gruetzemacher et al. in the form of a mobile application[7]. This would be a key feature that would aid in training users in performing vital or sensitive motions that have a higher risk of injury safely.

1.5 Significance of the project

The main area of knowledge this project will contribute to is around the collection and delivery of data from sensor to consumer. The main focus area of research that has been

covered in this space has focused on protocols, post-processing and consultations with a user and their supervisor. This dissertation aims to help advance the field with regard to how quickly data can be received by a supervisor from multiple sensors without costly equipment. The effectiveness of different real time corrective filters and data processing will be briefly presented.

Currently, most of the desired tests can be done in various facilities but with costly interfaces such as motion capture cameras or high-end sensors, however these are not easily obtainable for many facilities that may require them. More importantly, users may not be able to self diagnose errors in their training or recovery without the aid of a supervisor. This project aims to support this by allowing users to be better trained and let trainers be more informed about their athletes health at a lower cost using similar principles of observation.

1.6 Practical applications for the expected results

Results from this study would allow 3rd parties and other researchers the ability to build upon an existing system with standardised protocols to extend the system's functionality. With this project being open source, the possibility of this system being commercialised exists, as the underlying intellectual property is covered by the MIT Licence. An example of where 3rd parties may benefit is shown in Gruetzemacher et al.'s IoT based health app, [7] when this system may be added on to aid with screening athletes.

The system being designed may find itself in use by trainers and medical practitioners, if so approved, to aid in the supervision of athletes and patients as they train or recover away from or within their facilities as was demonstrated by Kos and Umek[2]. In addition to this, the system could be used in the development of various motion capture software applications on various platforms, be it mobile or desktop based.

1.7 Scope and limitations

This project will consider journal articles, conference papers and dissertations authored between 2005 and 2021 and gathered from Google Scholar, meaning that the main sources were IEEE Xplore, ResearchGate, Academia and various other educational knowledge hubs.

While a majority of devices exist that can obtain the useful metrics for this project, only options that are available to the public will be examined and referenced. Comparisons will be made to some proprietary systems, however they will not form any part of the

system that is to be designed.

With regard to the intended audience, this device will be designed with human beings as the sole users and will not look to be used with animals or other species. All data used in this project will strictly relate to humans and human-safe technologies.

Finally, a budget of R10 000 will be set for the development of the device. This will affect the different components that will be used, however the final cost per system should end up below the average market cost of, R45000 at the time of writing this dissertation.

Chapter 2

Literature Review

This chapter gives a review of the state of the art relating to Motion Capture, with specific attention to how IMUs have been implemented in this field. The general outline of this chapter is presented in Figure 2.1.

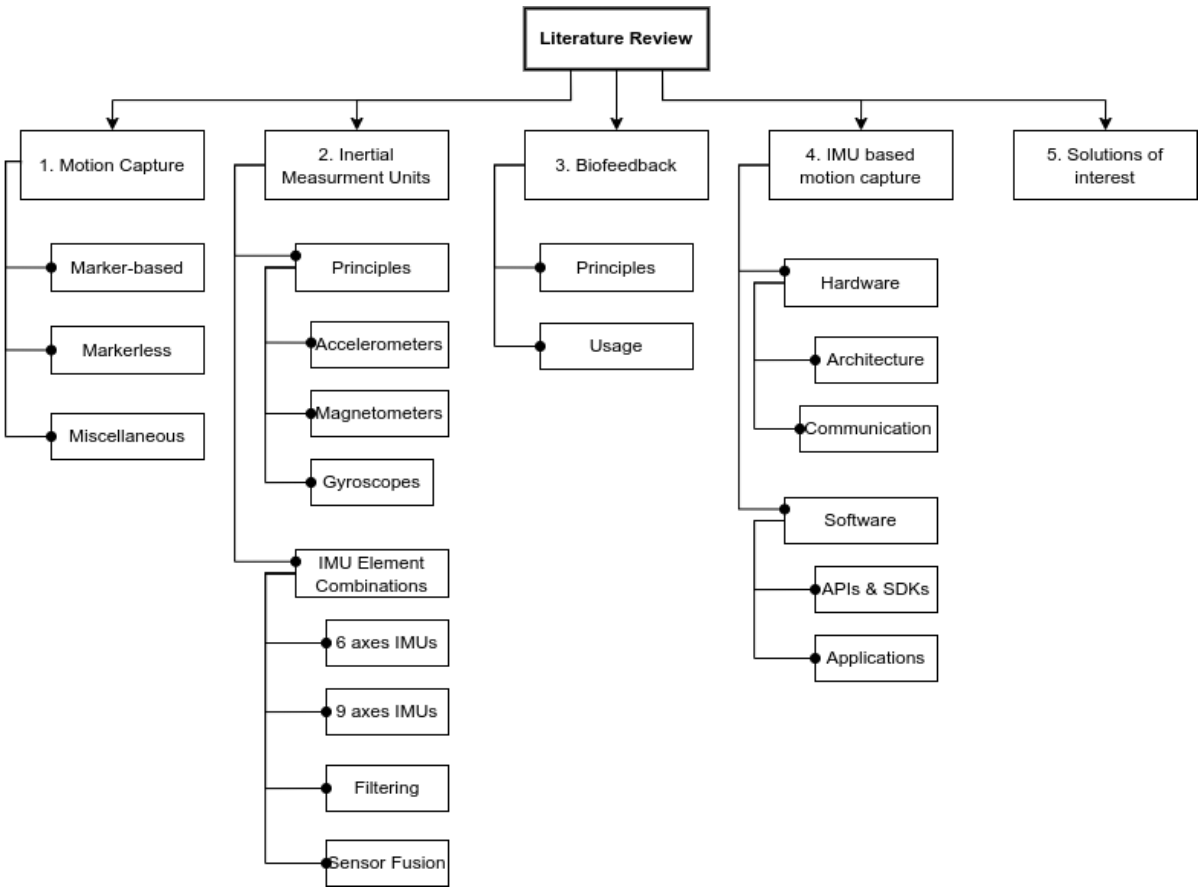


Figure 2.1: The key topics which will be covered in this literature review.

As is shown in Figure 2.1, this review is split into five different sections which will build

upon each other to culminate on the overall system requirements which Action Tracer will need to adhere to. The first of these sections will introduce motion capture and examine the various ways it can be done. The second section will introduce IMUs and discuss how they are can be used in obtaining position estimations in 3D space. The third section will introduce the concept of biofeedback and it's role in various forms of training and therapy. The fourth section combine the findings from the preceding sections to examine various IMU based motion capture systems from a hardware and software perspective. The final section will hone in on two solutions which would be used as a springboard for the development if Action Tracer. The findings of this review will lead to the formulation of the system requirements, presented at the end of this chapter.

2.1 Motion capture

Taking a deep dive into motion capture will open up many methods of dividing motion capture. For this dissertation, motion capture technologies will be divided into two broad categories, marker-based and markerless, as is presented by Nogueira[8]. The difference between the two is the use of 'markers' to gain an awareness of the movement of an object. These markers include sensors, mechanical devices or reflectors that allow a form of location feedback for processing. Over the years, there have been various leaps in the reliability, complexity and affordability motion capture systems, with some of their subgroups and sub-categories receiving little development while others are, to this day, being developed further. Today, there are many comparative studies between the two fields of motion capture, such as those done by Takeda[9], Ceseracciu et al.[10] to name a few.

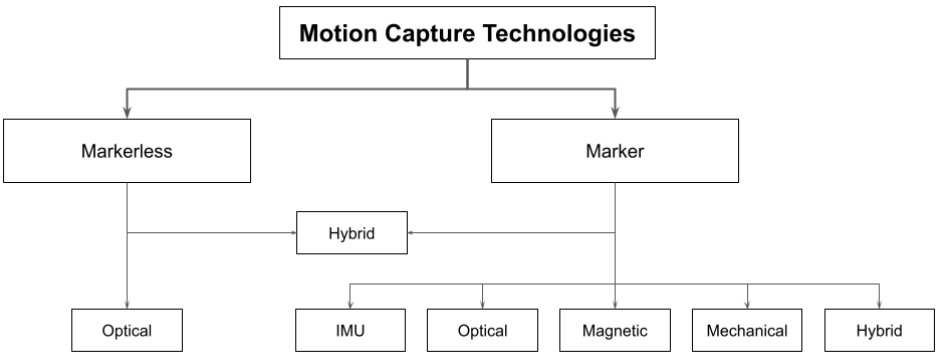


Figure 2.2: A brief overview of the different kinds of motion capture systems currently in use.

This review will cover the technologies shown in Figure 2.2 to remain brief. As stated, there are various other nomenclature and combinations of Motion Capture Methods available, with a few presented in Section 2.1.3.

2.1.1 Markerless Motion Capture

Markerless systems would be the most ideal intervention for motion capture in most industries. It is non-invasive on the subject to be captured and has minimal set-up required on the subject's part. An in-depth look at the state of this field was produced by Mündermann et al. in 2006 that showed the lack of development in the field at the time[11].

Most commonly used for markerless systems are cameras that capture images or record videos and combine this with computer vision algorithms to obtain object positions and rotations. Some of these methods are monocular, meaning that they rely on a single capture perspective (a single camera). A review of some systems developed between 2014 and 2022 is presented by Liu et al., showing the vast improvements over time in this field, but there is still need for improvement[12]. On the other end of the spectrum are multiple perspective systems using multiple cameras to obtain data. Each camera produces a 2D image, while multiple cameras enable 3D Motion Capture.

It is possible to construct a 3D perspective based on a single camera through various deep learning algorithms that will lift the various key frames out of the image[13]. Another method of obtaining a 3D perspective is to algorithmically combine multiple independent monocular views, as shown by Mohedano and Garcia, who used Bayesian Association[14].

In practice, markerless motion capture methods are utilised in various software packages such as SnakeStrike[15], OpenPose[16, 17, 18, 19] or MediaPipe[20] which are all Opensource.

2.1.2 Marker Motion Capture

There are many kinds of markers used in motion capture, ranging from sensors to physical points distributed on a subject to be used as references. The common thread to this type of motions capture is the need for a form of preparation on the subject, depending on the marker type being used. An example of motion capture markers is shown in Figure 2.3[21]. A brief overview of markers is given in the following sections, with criteria based on multiple topologies in various spaces such animation and sport[8, 22, 23].

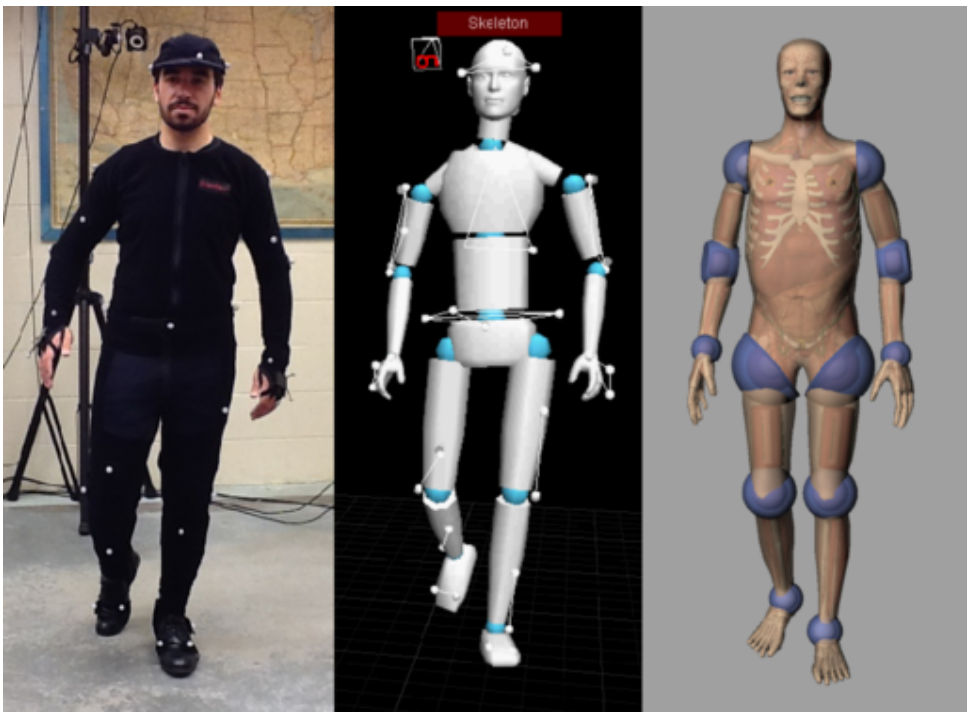


Figure 2.3: Optical markers used in motion capture over various stages.

2.1.2.1 Inertial Measurement Unit (IMU)

IMU markers are a very common marker in use because of their ability to avoid issues such as occlusion, dropout and, to a limited extent, environmental factors[24]. IMUs will be covered further later in the literature review, but to give a brief overview of where they have been used, the following paragraph details some use cases of these devices.

Slade et al. implement a system called OpenSenseRT that is evaluated as making it possible to implement a low-cost motion capture suit[25]. In their paper, they also present multiple solutions that use IMU markers, showing the variety of combinations of number of sensors, portability and various other metrics. What is common among IMU markers however is their potential to lose accuracy as a result of their placement on a body because of soft tissue artefacts[24]. A further implementation of these markers has been in sporting equipment such as a rowing paddle as is shown by Gravenhorst et al.[26]. In their paper, they use a phone to exploit its capabilities and sensors to capture the rowing motion and show the importance of sensor placement and the accuracy of mobile phone sensors.

2.1.2.2 Optical Methods

Optical markers are markers that rely on a camera to capture them in order to be used. Most commonly known are motion capture suits that use multiple obvious or reflective points all over the body to allow software to follow these points and translate the motion into virtual models. One common issue experienced with these systems is marker occlusion and obstruction[27, 28]. While Holden presents a solution that uses machine learning to address this issue, there is a great deal of time taken to help train and implement such interventions[27].

The different types of markers are also key to the success of systems that use this solution. The two broader groups are active and passive markers. The former, include markers such as LEDs that are turned on one at a time and localised in software and mapped to the appropriate segment in software[29]. As for the latter, passive markers, these rely on keeping track of multiple reflective points and mapping them accordingly.

2.1.2.3 Magnetic Methods

Through using some of the principles of magnetism and resonance, it is possible to obtain position data as was proposed and tested by Raab et al.[30] in 1979. This principle and technique has since been developed to allow full bodysuits to be commercially viable and used, as was done by Hashi et al.[31] and O'Brien et al.[32]. At the time of writing, this method of motion capture is not as common due to how prone it is to interference from metal[30].

2.1.2.4 Mechanical Methods

This is one of the cheapest methods developed to obtain motion data, often involving the use of rods, and simulated joints. A good amount of work has been done with regard to reading sign language as was done by Oz et al.[33]. The other option is to use resistive materials such as smart fibres and strain gauges to obtain motion data[34, 35].

One popular option is the Cyberglove™which was used by Pérez-Duarte et al.[36] and Oz et al.[33]. A very high precision can be obtained from such a device in the field of medical procedures and in sign language interpretation. Rajendran et al. utilises pressure sensors and strain gauges to obtain finger flex and object detection[34]. They were aiming to implement the system for sign language interpretation through letter diction and hand rehabilitation. A more focused attempt with just strain gauges is shown by Das et al. with 85% accuracy in basic hand movement detection[35].

2.1.3 Miscellaneous Motion Capture Methods

While the broader group of motion capture methods is sufficient for most options, some methods of motion capture do not fit into the category of marker or markerless capture. Over and above these, are more localised methods that are not so common or not commonly used due to their individual limitations.

Electromyography (EMG) uses electrodes that detect muscle contractions, obtaining a signal to show different muscles in motion. More often used with other motion capture methods to improve data as shown by Pradhan et al.[37] and Pérez-Duarte et al.[36].

Force Myography (FMG) uses Force Sensitive Resistors (FSRs) on the surface of the user to detect the volumetric changes in the underlying musculotendinous complex[38] This is more often used for patients who have lost a limb. Testing is often done on patients with limbs[39] and those without. Other uses have been in operating exoskeletons, as shown by Xing et al.[40].

Light As opposed to using a markerless technology, this involves the use of Light Dependant Resistors and light channels to estimate motion. One such attempt was made by Praveen and Karanth to interpret sign language based on hand and finger motion[41].

Beyond such methods, it must be noted that a mixed approach of techniques can be used in motion capture to attempt to improve motion capture data. Examples include IMUs and EMG[42], strain gauge and pressure sensor[34], EMG and strain[36] or magnetic sensors and IMUs[43].

2.1.4 Summary

This section presented a general overview of the two main classifications of motion capture systems, marker-based and markerless. In this exploration, the various ways in which both types of systems can be realised was presented with markerless systems vastly improving over the last decade with advances in software and camera technology. With regard to marker based systems, this sector has had some if it's implementation methods develop more than others. It was also found that a mixture of marker-based and markerless methods have can be used to try obtain better estimations in motion tracking.

2.2 Inertial Measurement Units

The term 'IMU' is a more succinct name for a device that often consists of two or more Micro Electromechanical Systems (MEMS) elements. A MEMS element consists of a small system containing one or more electrical and mechanical device that are interacting to either acquire, control or display data[44]. In the case of an IMU, the elements used often are Accelerometers, Gyroscopes and Magnetometers. In many cases, these are manufactured with three axes of sensitivity, simply described as X, Y and Z or Yaw, Pitch and Roll.

2.2.1 Principles of operation

The method by which each of these elements obtains data needs to be understood to better find how suited each of the methods are to motion capture. A brief explanation of how data is often acquired for the most common IMU elements is presented in the following paragraphs.

2.2.1.1 Accelerometer

An accelerometer is used to measure acceleration, better described as the rate of change of velocity measured as velocity per unit time (m/s^2), where velocity is displacement per unit time (m/s). There are two kinds of these devices in production,

AC Coupled These are linked with piezoelectric transducers in a charge or voltage mode[45]. These often operate by converting a piezoelectric transducer's output into an electric voltage or charge which is a representation of the acceleration event. These are considered to be much more accurate and have a higher signal-to-noise ratio, often leading to a higher price. Often, these are not good for measuring events where the tramsducer remains stationary, as they rely on the transducer's membrane vibrating.

DC Coupled Often made around capacitive or piezo-resistive components. With capacitive elements, a varying capacitance allows a measured signal to be acquired, while piezo-resistive accelerometers rely on strain gauges to produce a variable resistance[45]. While these are much cheaper and easier to produce, they suffer from high levels of noise and have a very low band-

width, with piezo-resistive accelerometers having a higher bandwidth than capacitive accelerometers.

A review of various devices is offered by Plasqui et al.[46]. In the article, some advice on choosing a type of accelerometer based on activity are given as a result of reviewing multiple other studies and the data obtained from them. This article recommends the use of piezo-resistive or capacitive accelerometers when there needs to be a distinction between various postures or activity types.

2.2.1.2 Gyroscope

Gyroscopes are used mainly to obtain the rate of change of angle, often measured in degrees changed in a unit time (°/s). Liu et al. provides a brief look at some gyroscopes in use and their different categories based on their principles of operation[47]. A brief introduction to each category is presented in the following paragraphs:

Coriolis Effect Gyroscopes of this nature use principles relating to the Coriolis effect in order to obtain a representation of the angular rate of change. A more in depth explanation of this phenomenon is given by Xia et al.[48]. There are various ways that this effect can be used with a common method involving a constantly oscillating capacitive mass being disturbed. A change in capacitance will be detected that is proportional to the Coriolis force which results from this disturbance.

Levitation Effect This method is based on the classic gyroscope model that involves a rotating mass that is affected by any twists and turns. While rotating, the rotating mass will want to return to its stable position and for the case of MEMS gyroscopes, this may be detected as a capacitance change[47].

The final two types of gyroscopes are not commonly used in low-cost systems, and will therefore only be mentioned and not explored further in this dissertation. These are: Sagnac Effect gyroscopes which are based on light[47], and Nuclear Magnetic Resonance gyroscopes, which are based on magnetics[49].

2.2.1.3 Magnetometer

A basic classification of magnetometers based on the physical effects they rely on will be used, as is given by You in their book, "Space Microsystems and Micro/Nano Satellites"[50]. You also notes that magnetometers can be classified based on the detection technique used, but for the sake of brevity, only the physical effects will be considered.

Induction Based on Faraday's Electromagnetic Law. More on how magnetometers do this can be found in Coillot and Leroy's book chapter, however a brief summary of the principle will be outlined[51]. Based on Faraday's Law, a voltage will be induced when a conductor with current flowing through it is subjected to a magnetic field. This will vary depending on the strength of the field. This measured voltage can describe the strength of a magnetic field around the device. This is known as the Hall Effect.

Magnetic These are based on two modes of operation, frequency modulation (FM) or amplitude modulation AM, both of which rely on Lorentz force. This is a force resulting from an AC excitation current, which will interact with a magnetic field and be bent as a result of it[52]. The distinction between AM and FM pertains to the method of measurement, where AM looks at the amplitude of the resonator and FM measures the resonator's frequency.

Most of the magnetometers on the market are based on the principle of induction or magnetics. Other kinds of magnetometers are those based on Magnito-Optical observations[53], and Superconducting Quantum Interference Devices (SQUIDs)[54].

2.2.2 IMU Element Combinations

In order to qualify as an IMU, a device must be able to measure acceleration and angular velocity at the very least. It has been found that some authors would refer to an IMU with a magnetometer as an Inertial Magnetic Measurement Unit (IMMU), but for this in this dissertation, the terms IMU and IMMU may be used interchangeably. There are two types of IMU available, 6-Axis and 9-Axis. Given that each IMU element may have three axes of measurement, a 6-axis IMU often has an accelerometer and gyroscope while a 9-axis IMU contains the same two devices as well as a magnetometer. To better understand the nuances associated with each combination, multiple researchers have looked into the

effects of each design on the gathered data. A brief motivation for each configuration is given in Sections 2.2.2.1 and 2.2.2.2.

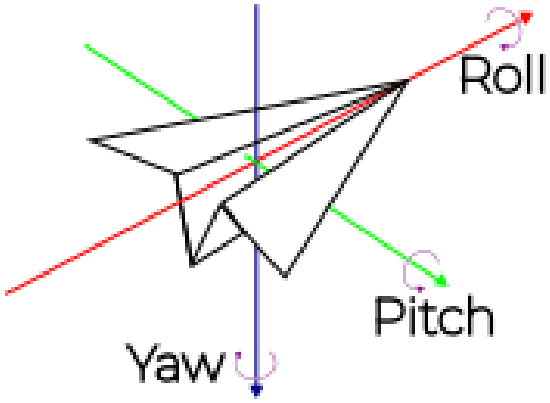


Figure 2.4: A visual illustration to help define yaw, pitch and roll.

2.2.2.1 6-Axis IMU

6-Axis IMUs mostly comprise a 3-axis accelerometer and a 3 axes gyroscope. This coupling is often used, as an accelerometer can only output data useful to calculate roll and pitch. In some cases, geomagnetic sensors (accelerometer and magnetometer sensors) may be used, such as Adafruit's LSM303AGR[55] or as is common in many smart mobile phones. This dissertation will focus on the former definition of a 6-Axis IMU.

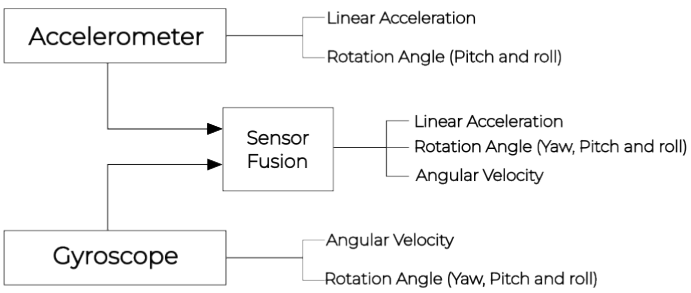


Figure 2.5: An illustration of the data obtained when using an IMU with an accelerometer and gyroscope.

Figure 2.5 shows the data obtained from both an accelerometer and a gyroscope. While the accelerometer can provide rotation angle data, it will not be able to produce yaw data. This is because it does not have a reference on that axis. By looking at Figure

2.4, it's observed that the reference to the yaw axis will always have gravity affecting it. This does not mean it is impossible to get accurate data that is not affected by gravity, as Nurhakim et al. from developing novel methods to obtain yaw data that came within 6.42% of the true value[56]. This shows the importance of also using the data from the gyroscope, which is not affected by gravity, meaning it will be able to obtain yaw, pitch and roll. How this is obtained will be discussed in Section 2.2.2.4, Sensor Fusion.

Both elements will also produce individually unique data, linear acceleration and angular velocity. For the former, this data may be used for activity classification even if collected from a smartphone, as was done by Bayat[57]. With the latter, such data could be used to identify swimmer pose and stroke, as was shown by Kos and Umek in their paper on biofeedback[2]. It must also be noted that these two metrics are the only real elements measured by the accelerometer and gyroscope, so in reality, all rotational data is sourced from linear acceleration and angular velocity.

2.2.2.2 9-Axis IMU

When all three sensors are used, accelerometer, gyroscope and magnetometer, the same principles as in the 6-axis IMU apply, with the benefits of a magnetometer. These mainly come from its ability to be referenced to magnetic north, allowing calibration against a constant reference[58].

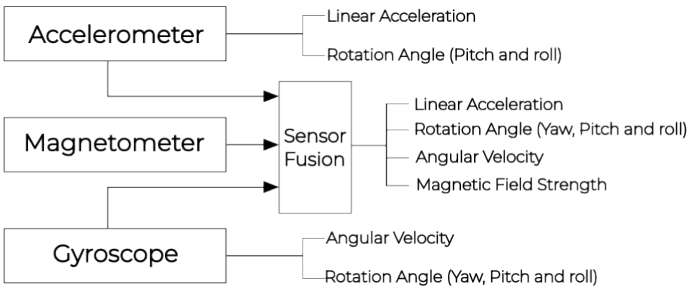


Figure 2.6: An illustration of the data obtained when using an IMU with an accelerometer, gyroscope and magnetometer.

It may be stated that in areas where a high degree of accuracy is crucial, a 9-axis IMU should be used due to its better performance. A study by Ahmad et al. analysed the different sectors where these IMUs are in use and confirmed that they are often used for better performance[59]. They were mainly used in robotics and industry, with a suggestion that most areas where the 6-axis IMU was being used will see them replaced. Therefore there is a preference to 9-axis IMUs as opposed to 6-Axis IMUs where such an option is available.

2.2.2.3 Filtering

A big area of research is IMU data filtering. As mentioned in Section 2.2.1, the various IMU elements have different compositions and manufacturing considerations. This leads to measurement limitations and inaccuracies that cause erroneous readings. In most cases, a MEMS IMU consists of a DC coupled accelerometer, Coriolis effect gyroscope and induction magnetometers. For these, often capacitive elements are used to measure the data by way of Analogue to Digital Converters (ADC). Following these paths, there are various areas where data may be incorrectly sampled.

A better classification of these errors and their impact is presented by Flenniken et al. in their paper on IMU error characterisation for a 6-axis IMU[60]. Using their simple model and generalising it,

$$s_r = r + c_r + b_r + w_{sens} \tag{2.1}$$

where s_r is sensor output, r being the true reading, c_r a constant offset, b_r as a moving/walking bias and w_{sens} representing wideband sensor noise[60]. The moving/walking bias is what is commonly referred to as drift and wideband noise is noise that covers and exceeds the principle sampling frequency.

In most cases, a low pass filter is sufficient to get rid of most of the noise generated as a result of the sensor readings and in some cases, the IMU is equipped with digital filters that will be used to process this data before it is delivered to a requesting device. Not all the wideband noise is removed, however, necessitating the use of further filters. In some cases, a complimentary filter may be implemented on the IMU, allowing the fusion of the different IMU element data before being delivered to the processing device. This will be covered further in Section 2.2.2.4. This is not to say however that sensor fusion and low pass filters can not be implemented on the processing device.

An alternative to on-board IMU filtering would be using a post-processing method where the data is filtered just before use. There is generally a lot more work in this section with regard to IMU data processing, as is shown by Madgwick's custom filter[61], various types of Kalman Filters as are used by Abbate et al.[62] or a moving average filter by Ghio et al.[63].

2.2.2.4 Sensor Fusion

While sensor fusion may be used as a method of filtering sensor data, it may be taken as its own part of the data processing chain. Sensor fusion is the integration of multiple sensor's data to create a single data signal and can be based on one of three techniques: probabilistic, least-squares or intelligent[64]. As was mentioned in Section 2.2.2.1, gyroscope and accelerometer data needs to be combined to obtain the three rotation angles.

With both the accelerometer and gyroscope being able to produce pitch and roll, a more in depth look will show the need for the fusion, which will be the only contributor to the yaw component of a data reading.

By taking advantage of each sensor's strengths, a bias may be set to obtain roll and pitch from both sensors and have a weighted output. The accelerometer is good for long term readings that may be used to diminish gyroscope drift while the gyroscope would be effective in short amounts of time, before the drift is an issue. To investigate this, the following definitions of the acceleration vector and the gyroscope vectors will be used:

$$\vec{A}(t) = [a_x(t), a_y(t), a_z(t)] \qquad \vec{G} = [g_x(t), g_y(t), g_z(t)]$$

where $\vec{A}$ is a vector composed of the acceleration components in the x, y and z directions made up of a_x, a_y and a_z respectively. In the same way, $\vec{A}$ is a vector composed of the angular rotation components in the x, y and z directions made up of g_x, g_y and g_z respectively. Using $\vec{A}$ and $\vec{G}$, Equation 2.2 will show how $\vec{A}$ can be used to obtain roll(ϕ) and pitch(θ) data while Equation 2.3 will show how roll, pitch and yaw can be obtained using $\vec{G}$.

$$\phi = \arctan\left(\frac{a_y(t)}{a_z(t)}\right) \qquad \theta = \arcsin\left(\frac{a_x(t)}{a_y(t)}\right) \tag{2.2}$$

It can be seen that Equation 2.2 is calculated instantaneously and does not rely on past readings in order to provide a reading. In addition to this, no method to obtain the yaw component is given as this is a more verbose process which is out of the scope of this research.

$$\phi_{x(n)} = \phi_{x(n-1)} + g_x(n)\Delta t \qquad \theta_{y(n)} = \theta_{y(n-1)} + g_y(n)\Delta t \qquad \psi_{z(n)} = \psi_{z(n-1)} + g_z(n)\Delta t \tag{2.3}$$

Equation 2.3 introduces n, which shows that the present reading is reliant on past readings, showing the cumulative nature of a reading obtained from a gyroscope. A complimentary filter can fuse the two readings together based on a bias term, α ranging between 0 and 1, as is shown in Equation 2.4.

$$\phi = (1-\alpha)\phi_a + (\alpha)\phi_g \qquad \theta = (1-\alpha)\theta_+(\alpha)\theta_g \qquad \psi = \psi_g \tag{2.4}$$

where ϕ_a and ϕ_g refer to the roll components of the accelerometer and gyroscope respectively, θ_a and θ_g refer to the pitch components of the accelerometer and gyroscope respectively and ψ_g refers to the yaw component of the gyroscope. Assuming the use of recent advances with regard to yaw readings, ψ may be dependent on ψ_a, the yaw component resulting from an accelerometer reading, found using novel techniques[56], and ψ_g, based on Equation 2.3.

It must be noted that yaw pitch and roll are subject to gimbal lock. This is a phenomenon where two of the axes of rotation become aligned, such that their individual rotations are indistinguishable. This causes a loss of axes, leaving what was once a 6-axis device as a 4-axis device until resolved. The yaw, pitch and roll are based on Euler angles, and there are multiple algorithms in development to either avoid this scenario, or to correct it when detected, but there is a separate solution which involves using a 4 dimensional coordinate system that involves sensor fusion.

By describing rotations in a 4 dimensional space, many of the pitfalls from a 3 dimensional space are solved and avoided, but there are computational drawbacks in doing so. Figure 2.7 is an adaptation of Hindle et al.'s method of obtaining quaternion data[24].

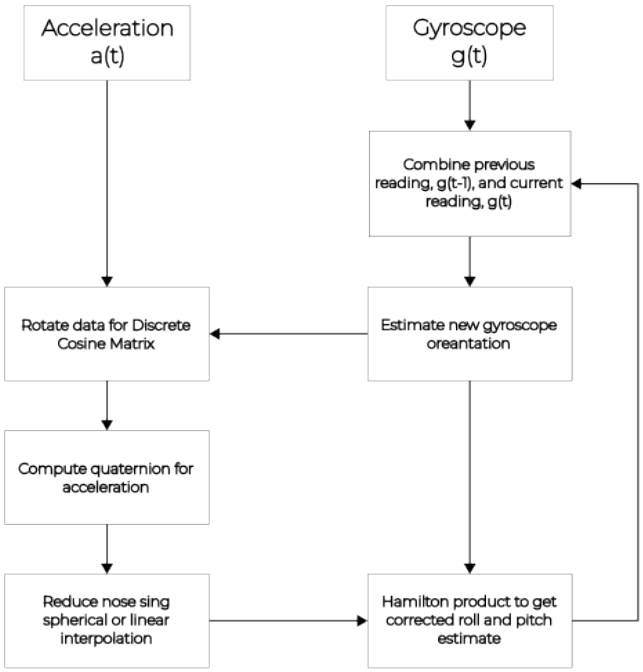


Figure 2.7: A flow diagram showing how the quaternion rotation data is obtained from raw accelerometer and gyroscope readings at time t

A quaternion is a four-dimensional complex number that can be used to represent the orientation of an object in three dimensional space. It is possible to convert quaternions

to Euler values by following the Equations 2.5 - 2.7, where $\vec{q} = [q_w, q_x, q_y, q_z]$ in which q_w, q_x, q_y and q_z are magnitudes of each of the components that make up a quaternion.

$$\psi = \arctan \frac{2q_x q_y - 2q_w q_z}{2q_w q_w + 2q_x q_x - 1} \tag{2.5}$$

$$\theta = -\arcsin 2q_x q_z + 2q_w q_y \tag{2.6}$$

$$\phi = \arctan \frac{2q_y q_z - 2q_w q_x}{2q_w^2 + 2q_z^2 - 1} \tag{2.7}$$

A final property of quaternions which is commonly used is that to obtain the angle between two quaternions, $\vec{p} = [p_w, p_x, p_y, p_z]$ and $\vec{q} = [q_w, q_x, q_y, q_z]$. This result may be obtained by following Equation 2.8 where χ is the angle between quaternions $\vec{p}$ and $\vec{q}$.

$$\vec{p} \cdot \vec{q} = |p||q| \cos \frac{\chi}{2} \tag{2.8}$$

2.2.3 Summary

This section presented a general overview of the various components that make up an IMU. These are an accelerometer, gyroscope and magnetometer. While not all three need to be present, a look into how the various data outputs from the accelerometer and gyroscope was could be transformed into useful angle data was also presented. The quaternion representation was also presented as a means of solving some of the issues that may be encountered when using Euler representations. This section also presented the the concepts of data filtering as well as sensor fusion, methods by which data from an IMU could be stabilised for more accurate readings from either a single IMU element or multiple elements.

2.3 Biofeedback

As was mentioned in Chapter 1, there is a need for devices which are able to give feedback to their users in the various interventions for rehabilitation and training. This is more broadly known as biofeedback and has been covered in various contexts by various authors. First, an overview of the importance of biofeedback will be presented as well as some of its core principles, followed by some work that has been conducted in the past

will be examined.

2.3.1 Principles of biofeedback

First a definition of biofeedback, as given by Kos and Umek, is the use of sensors to measure physiological and physical body parameters and activities which a person may be unaware of or unable to detect[65]. As opposed to monitoring, the field of biofeedback aims to allow subjects to have access to information given to them in a more digestible format which is often linked with one of their senses. A distinction is made between the physiological and physical monitoring due to the vast differences in the types of data involved in each. Physiological biofeedback would relate more so to monitoring internal bodily functions and systems such as the respiration or pulse, while the physical markers would include gait and arm rotations.

The given definition of biofeedback however does not take into account the fact that this field is much more broad than just sport and activities. It has been explored in other fields such as in treating muscular disorders[66], or in treating irritable bowl syndrome[67]. Frank et al. present a case for biofeedback not being considered a treatment and to be considered as training[68]. This is because this intervention is aimed towards educating a user and training them or their bodies to respond so specific stimulus in a desirable way.

Biofeedback works in a closed loop system much like that of most control systems, where the expected result is a change in response from the user being examined. This loop consists of three stages: sensing, processing and monitoring as is shown in Figure 2.8.

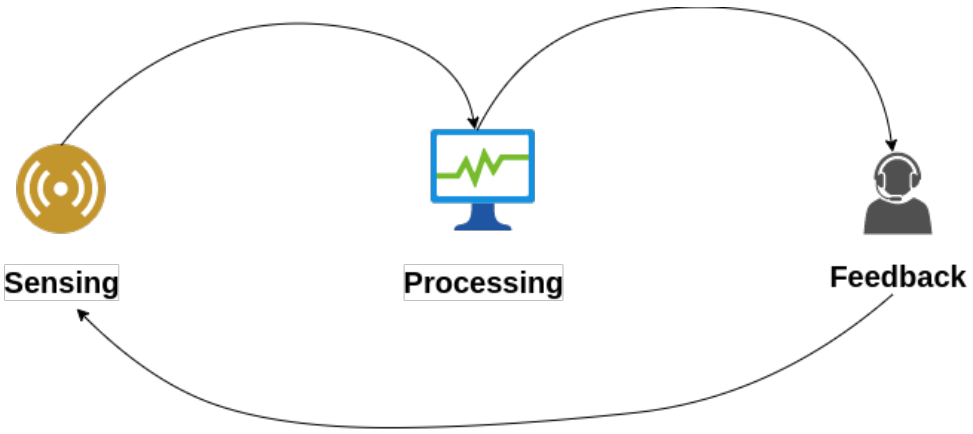


Figure 2.8: The 3 stages of the biofeedback loop in the context of a patient.

It is the open-ended nature of the processing step that allowed various forms of intervention that go across various spheres of medicine, sports, and psychology. To understand these steps further, they will be presented in the context of Hermanis et al.'s design of

an intervention for monitoring the posture of users in daily activity[69].

2.3.1.1 Sensing

This step involves the use of various kinds of sensors depending on the category of biofeedback being explored. The preceding task to biofeedback is finding a metric that may be observed that is closely linked with the attribute that is under investigation. For posture monitoring, Hermanis et al. made use of an array of accelerometers which could be worn by a user. These accelerometers were arranged in a 4 by 4 grid and strapped on to a users back. By observing acceleration of the different sensors during a user's day, the movement of each of the devices would paint a picture of the user's posture.

2.3.1.2 Processing

This stage is used in combining the data obtained from the sensors in order to help understand what is being done by a user and what they may need to do to improve their performance. Connected to this step, would be communication, how the data is transported to the processing step. This may be done in real time or post training session. Hermanis et al. uses a wired connection to collect data from each of the accelerometers and send that to a microcontroller. The microcontroller would then send this data to a smartphone using Bluetooth where it would be used to estimate the user's posture.

2.3.1.3 Feedback

The feedback step is the most important step in this loop, as it is the step that initiates change. Various methods may be used, such as visual cues or a post-action conversation based on the data. Hermanis et al. utilises the vibration motors in the smartphone to achieve this for haptic feedback. The same was also done using the speaker of the phone as a form of auditory feedback.

2.3.1.4 Monitoring

This is an optional step that depends on the system being used and observed. In the case of a slower paced activity, one may not need a monitoring step if feedback is given as the activity is being conducted. Hermanis et al.'s system does not directly mention such a system but the smartphone application could be used to store collected data and allow for further analysis.

2.3.2 Biofeedback in use

In order to see the benefits of biofeedback, a few case studies may be examined from various fields and sources where such methods have been used. While one such area is sport, various other disciplines have utilised such interventions. One such example has already been presented in demonstrating biofeedback which was created by Hermanis et al.[69]. This study was very helpful in the healthcare space and such posture monitoring could help greatly improve the quality of life for people suffering with bad posture.

Another example of a system that allows for biofeedback is given by Kos and Umek in a study which involved building a wearable system for swimmers[2]. In their system, they utilised 6-axis IMUs to detect strokes, stroke rate and stroke count. In this current state, the system relied on a rehabilitation therapist to provide the feedback based on the obtained information. Based on the data obtainable from this system, it was shown that a further system could be built over it to provide the feedback autonomously or to further digest and present the data to either the user or the therapist.

A final intervention is presented in Li et al.'s swimming biofeedback system[70], which is a form of realisation of the goals of Kos and Umek's goals. This system includes vibration sensors which were used to provide users with haptic feedback when they exceeded given thresholds. The main focus of this study was to validate a system that met the criteria of the biofeedback loop, hence a lack of empirical data showing its effects on an athlete. Though lacking, Li et al. managed to show how the system would provide feedback to the athlete as well as to a trainer in real time.

2.3.3 Summary

This section provided a brief look into the definition of a biofeedback system, and its potential uses in both healthcare and sports training. The key principles of biofeedback are sensing, processing, feedback and monitoring (optional). Having defined the biofeedback, various examples were examined where the use of such principles was implemented and the benefits gained were presented.

2.4 IMU Based Motion Capture Systems

Having gained a broad understanding of motion capture and IMUs from Sections 2.1 and 2.2 respectively, a broad sweep of where these two fields intersect will be presented. This section will include a review of some systems produced by other researchers and companies to gain an appreciation of the use cases and system architectures. A greater

number of the systems presented here were used in sporting or medical environments.

In order to be classified as a motion capture system for investigation in this review, a list of questions has been drawn up which a motion capture system must meet. This list is as follows:

1. does the system utilise at least 1 IMU for motion capture?
2. does the system guarantee/specify an output data rate?
3. does the system offer unprocessed IMU readings and transformed readings?

If all the questions were answered in the affirmative, a system qualified investigation and is presented in Table 2.1. The accepted studies will be covered in more detail by examining them from two different perspectives, hardware and software, as will be shown in Section 2.4.1 and Section 2.4.2 respectively.

To help further assess the systems, a model standard known as IOTSport will be used. IoTSport is a branch of sports data processing where Ray gives a layer by layer overview of what needs to be implemented to aid the growing industry[71]. The layers, in order, are:

Physical Sensing Layer (PSL) The bottom of the chain where readings are taken by sensors.

Communication Protocol Layer (CPL) These are the different communication mediums to send data through.

Data Processing Layer (DPL) Where the data collected is processed into useful application data.

Internet Layer (IL) Routing processed data to a cloud/server for distribution or further processing.

Storage and Preview Layer (SPL) Allowing the collection of data from the cloud/server or directly bypassing the IL to collect the data directly.

Visualization and Service Layer (VSL) Performs the greater analysis of the data, allowing predictions and interpretation.

User Application Layer (UAL) The final layer, useful in managing what data is received by athletes, coaches, and managers.

While not all of these are necessary, they underpin the basis of most systems that will handle data collected from human motion and show the different considerations.

Table 2.1 showcases a breakdown of the various systems according to various key criteria, with three of the systems being developed in the last 15 years, two in the last decade and the remaining three from the last five years. These were the key factors that the studies mostly made reference to. Based on these defining characteristics, basic expectations of a new system can be defined and act as the figures of merit which will be compared against.

Table 2.1: A brief comparison of some motion capture systems that are have been produced by various researchers.

System Name	Author	Number of Axes	Accelerometer	Gyroscope	Magnetometer	Filtering /Fusion	Communication	Number of IMUs	Sampling Frequency (Hz)	Portable	Open-source	Real Time
OpenSenseRT	Slade et al.[25]	6	ISM330DHCX (STMicroelectronics)		None	Mahony	Wired	7	30	Yes	Yes	Yes
WB-4R	Kong et al.[72]	9	LIS331DLH	LYPR540AH	HMC5843	Kalman	Wireless	7	200	Yes	No	Yes
Custom Wearable	Gonzalez-Alonso et al.[73]	9	BNO080 (CEVA, Inc.)			Kalman	Wireless	10	60	Yes	Yes	Yes
Tennis Motion Capture	Buthe et al.[74]	9	EXLs3 (EXEL)			None	Wireless	3	200	Yes	No	No
ISWIM	Li et al.[70]	9	MPU-9250			Low Pass	Wireless	3	10	Yes	Yes	Yes
Abbate et al.	Abbate et al.[75]	7	LIS3LV02DL	LY530AL	None	Kalman	Wireless	1	50	Yes	Yes	Yes
Wade and Mataric	Wade and Mataric[76]	9	MXR9150	ADXRS300	HMCI051(2)	Kalman	Wireless	1		Yes	Yes	Yes
HMCD	Q. Jin et al.[77]	9	MPU9250			Low Pass	Wired	20	120	Yes	Yes	Yes

2.4.1 Hardware in IMU based motion capture systems

When it comes to the hardware used in motion capture systems, there are a few architectures available that have been used in different scenarios. Figures 2.9 and 2.10 show these configurations, and they will further be discussed in the sections to follow.

2.4.1.1 Single controller to many sensors

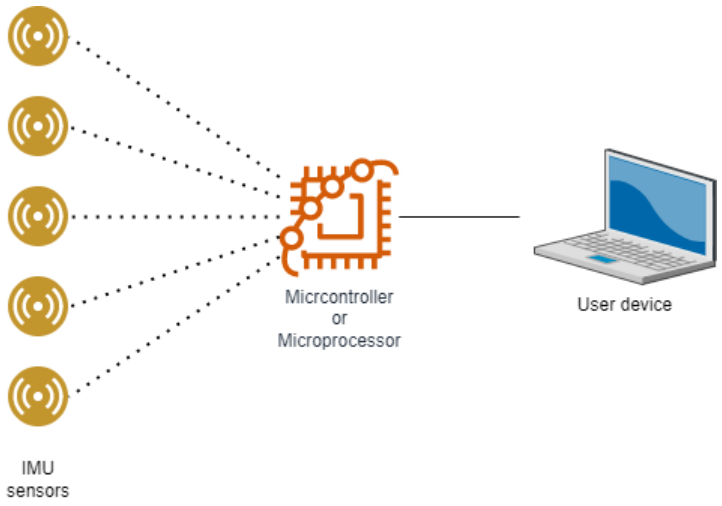


Figure 2.9: A simplified model of a basic motion capture system using a single controller and multiple sensors.

Similar to a mix of bus and star network topology, each of the sensors report their data to a single controlling device that will collate the data and submit the packet to the user's device as shown in Figure 2.9. This topology is dependent on the processing speed and power of the controller, as well as the interface used to connect to the sensors. A faster controller that can read all the sensors data would be preferred, as this allows a single data packet to be sent, easing the workload on the user's device when processing the packet. The disadvantage of this topology is the need to poll every sensor in the network, even with hardware interrupts on the sensors, each sensor can only submit its data packets when polled.

Authors that use this architecture as per Table 2.1 are Slade et al.[25]. In their paper, the controller used is a Raspberry Pi 4B+, with a motion capture software called OpenSim[78]. It is also noted that this controller also acts as a user device in their implementation. The sensors were connected using wires to the main controller and could be worn all over the body. When running this, it was found that an update rate of 30Hz was obtained when 7 segments were being used. Kong et al. uses this topology as well for their WB-4R motion capture system which communicates via CAN bus[72]. This system

was able to reach sub degree errors with regard to yaw pitch and roll at a data collection rate of 200Hz with data being sent to the user device via Bluetooth.

An exceptional use case of this topology in use may be given by Abbate et al. with a bespoke system that utilises single axis gyroscopes and a magnetometer alongside a 3 axes accelerometer[75]. The mode of data collection involves polling each device from an STM32F103 microcontroller via it's I2C and SPI communication interfaces and collating the gathered data. After polling, this data is preprocessed in an Extended Kalman Filter and is sent to the user device at a rate of 50Hz. This specific example may extend to an independent controller-sensor pair topology and will be further expanded upon.

2.4.1.2 Independent controller-sensor pairs

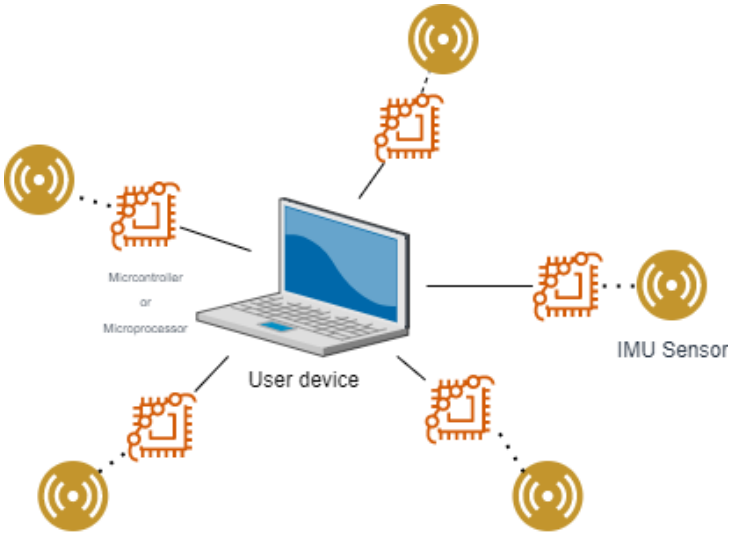


Figure 2.10: A simple model of a multi-controller motion capture system using multiple devices to capture motion data.

An easy method to offload most of the processing and collating being done by a single device, is to set up multiple nodes that independently collect data and send either fully processed or semi-processed data to the user device as is outlined in Figure 2.10. This would be leaving the user device to perform any further computation on the data, often allowing a higher data collection rate. In one scenario, a main node may be collating the data and sending it to the user device, offloading some extra processing to that node and letting the user device deal with much more quickly digestible data.

There are drawbacks with this system, such as the potential for packet loss during transmission, delays in packet reception and processing delays linked with receiving data out of order. The single controller – many sensors' method would solve this by storing sensor data in a predetermined order, but due to the potential lack of polling in this system,

there may be significant packet shortages or missing data points. Polling may be introduced on a communications level, but this would slow down the system significantly. With each node being independent, a single device may fail but would not greatly affect the rest of the devices in the system, which may be a benefit with this architecture.

To build on the potential of Abatte et al.'s system[75], their work may be duplicated with each new sensor node reporting to a single processor or user device. A better example of this topology is found in the Custom Wearable system[73]. Each node contains a battery, nRF52 microprocessor, 2.4 GHz Transceiver and BNO080 IMU. This system has a master node that polls each slave node at a rate of 60Hz. Gonzalez-Alonso et al. compare the CW System[73] to the XSens Dot by XSens Technologies, a proprietary system that is build around this architecture and Bluetooth communications[79]. The XSens Dot system relies on the user device to collate the collected data, which is collected at a rate of 800Hz and processes using their XSens Dot Software Development Package. Wei et al. use the XSens dot to develop their own system to track limb motion as well as improve the estimations of the limb using a recurrent neural network (RNN) but up to 10% at the time of writing[80].

The final two systems in this category are Buthe et al.'s[74] and Li et al.'s[70] systems. Both are used in the context of sport, with the former working on a system to capture tennis racket and player motion and the latter focusing on swimming data. Both systems rely on the athlete having a Body Area Network (BAN) to collect and send the data to a user device.

2.4.1.3 Transmission and Serialization

A major question that needs to be addressed with regard to transporting data is linked with the transportation medium and format. Depending on the medium of transportation, the format used may need to change or be adapted to simplify the transportation. Examples of mediums of transportation are Bluetooth, as is used by XSens Dot[79] and Buthe et al.[74], RF bands such as 2.4GHz as is used by Gonzalez-Alonso[73] or 433MHz in the case of Li et al.[70] and in some cases, wired solutions such as the CAN bus used by Kong et al.[72].

Referring back to Ray's IoTSport layers[71], the lower layers need a greater level of standardisation to ensure multiple different devices may collect data and interact with systems, irrespective of the higher layers closer to the user. One example of a low layer (CPL) standard is ANT[81]. This message format is used to communicate crucial configuration and data messages in a network. Structuring data packets collected from device to device would allow easy packet parsing and relaying, as well as channelling. Sama et al. present a simple IMU data packet structure as is shown in Figure 2.11[82].

Start_Byte	Device_ID	Device_Type	Data_Byte[0] ... Data_Byte[N]	Packet Count	End_Byte

Figure 2.11: Sama et al.'s data packet structure used for a motion capture glove.

Apart from Sama et al.'s paper, there is not a lot of literature relating to a standardised packet structure for IMU data. One source that greatly considered the packet structure and transmission protocol was Jin et al.[77], who's Human Motion Tracking Device (HMCD) utilised time sensitive networking to obtain high update rates with many sensors in the network. This may indicate a need for some investigation in data collection and transmission between the different layers of a system to allow multiple parties to utilise the data.

2.4.2 Software in IMU based motion capture systems

There are multiple classifications of software that are in use with regard to IMU motion capture. These fall into one of two categories, Add-ons and Applications. Add-ons typically are used in association with a host application which may be responsible for a majority of the execution and computations, while applications will contain everything required for the data to be captured, processed and visualised. When compared to each other, full applications tend to be highly specific in use and not as easy to modify when new equipment is being used. If any change is possible, it will often take more time and releasing a new version could potentially affect various systems already in use. Add-ons and plugins however do not take as long to produce or update due to being smaller in nature and reliant on a separate code base and framework. When building within these restrictions, only users utilising a particular add-on will be affected should there be a major change, but the time taken to fix such an issue is much shorter.

To have an understanding of how these applications are built, there needs to be a base understanding of some core concepts about integrating two systems. These include understanding Application Programming Interfaces (APIs) and Software Development Kits (SDKs). A basic definition of these is given by Stylos as high level interfaces for code reuse[83]. Upon these elements is built an application or an add-on for motion capture systems and various other tools used in capture, processing, and visualisation. A base understanding of each of these backbones is presented in the following paragraphs, followed by a look at some of the work that exists in the in this space relating to IMU motion capture in Section 2.4.2.1. Relative to Ray's layer model, this section deals with the SPL, VSL and UAL layers.

Application Programming Interface Simply described, an API is a

means that allows two applications or systems to communicate with one another. There are many types of API especially in the web development space, however, API in the context of this dissertation will refer to communications between a hardware system and software system.

APIs are more commonly used by developers when creating user software or tools to collect, process or send data between two systems. This would also encompass triggering the execution of specific commands on the requested system for the sake of data acquisition. An example of these are the Windows API (Win16 and Win32 APIs) which are developed by Microsoft for developers to use when creating applications for the Windows platform. They are written in the C programming language[84].

It must be noted that APIs are very limited in functionality. For the client software or developer, there is a limit on what can be done and requested from and by an API.

Software Development Kit An SDK is very much similar in objective to an API, but they are fundamentally different. SDKs will often have their own compilers or interpreters that may be built around a specific programming language with various supporting libraries. SDKs may also implement APIs to further allow inter-system interactions.

A good example of an SDK would be the Java Development Kit (JDK). In itself, it is not a programming language, but it relies on the Java programming language. It offers tools that allow a developer to create applications using Java such as libraries, a runtime and virtual machine. This means that the tools afforded by an SDK often have little restriction, allowing developers to do what they would like with any of the tools.

2.4.2.1 Software applications used for Motion Capture

There are various types of applications that integrate motion capture hardware and software. These range from purpose built and specific applications to smaller products that serve as add-ons to other applications and tools that are general purpose. In this section, the integration of motion capture systems and such tools will be presented.

First to be examined will be plugins and add-ons. These often take advantage of APIs

or SDKs and a host application to obtain data from the system and process it. One such example is the Biomechanics of Bodies (BoB) developed by Shippen and May at Coventry University[85]. BoB relies on a developer using Matlab. This reliance allows this add-on to benefit from the tools natively provided by Matlab, while extending this functionality with application specific software. Other common extendible applications include Blender and Unity 3D, where developers can take advantage of the host application specialities with the context of Motion capture. Blender, being specific to modelling and rendering, allows the use of the Blender API to interact with models as was done by Raghavendra et al.[86].

Slade et al. makes used of OpenSim, an open source application made to equip users with biomechanical models and simulation tools[78]. The application also allows users to build around an API through C++ or scripting in Java, MATLAB, and Python. While OpenSim mainly focuses on musculoskeletal analysis, with the ability to capture both bones and muscles as well as simulate their response to various scenarios. The application is also able to read multiple different motion capture file types and replay them. In Slade et al.'s case, the data was captured and directly relayed to OpenSim's inverse kinematics solver and joint orientations calculated and stored[25].

Wade and Mataric presents a different method of feedback by using a socially assistive robot (SAR)[76]. They developed a software driver to issue commands to the robot based on IMU data collected from a user and programmed the robot. The data itself may be stored and analysed in a different software, what may be of benefit to some users of a motion capture system that can directly control a robot or puppet. In this case, a purpose built application is used to allow the data collection, analysis, and usage.

2.4.3 Summary

This section looked at the use of IMUs in various motion capture systems. Having selected systems to examine by a set of criteria, these systems were analysed based on their various compositions such as the IMU elements used, filtering used, communication medium, number of IMUs used as well as the sampling frequency of each IMU. This was followed by a discussion of the hardware used in these motion capture systems, first by examining the various network topologies that were used, followed by looking at the transmission mediums and protocols implemented by the hardware. The software used in some motion capture systems was last to be examined. This was done by first defining what an API and SDK are, followed by a brief look at some of the systems used by various authors were as well as some of their key features. Of note was the used of plugins in order to facilitate the capture of motion capture data from the hardware, and have these supply host applications with pose estimation data.

2.5 Solutions of interest

When considering methods relating to building a new IMU based motion capture and biofeedback system, two previous systems were of interest while researching IMU motion capture systems with respect to hardware. Of note was the work done by Li et al.[70] and Slade et al.[25], whose systems were briefly presented and explored in Table 2.1.

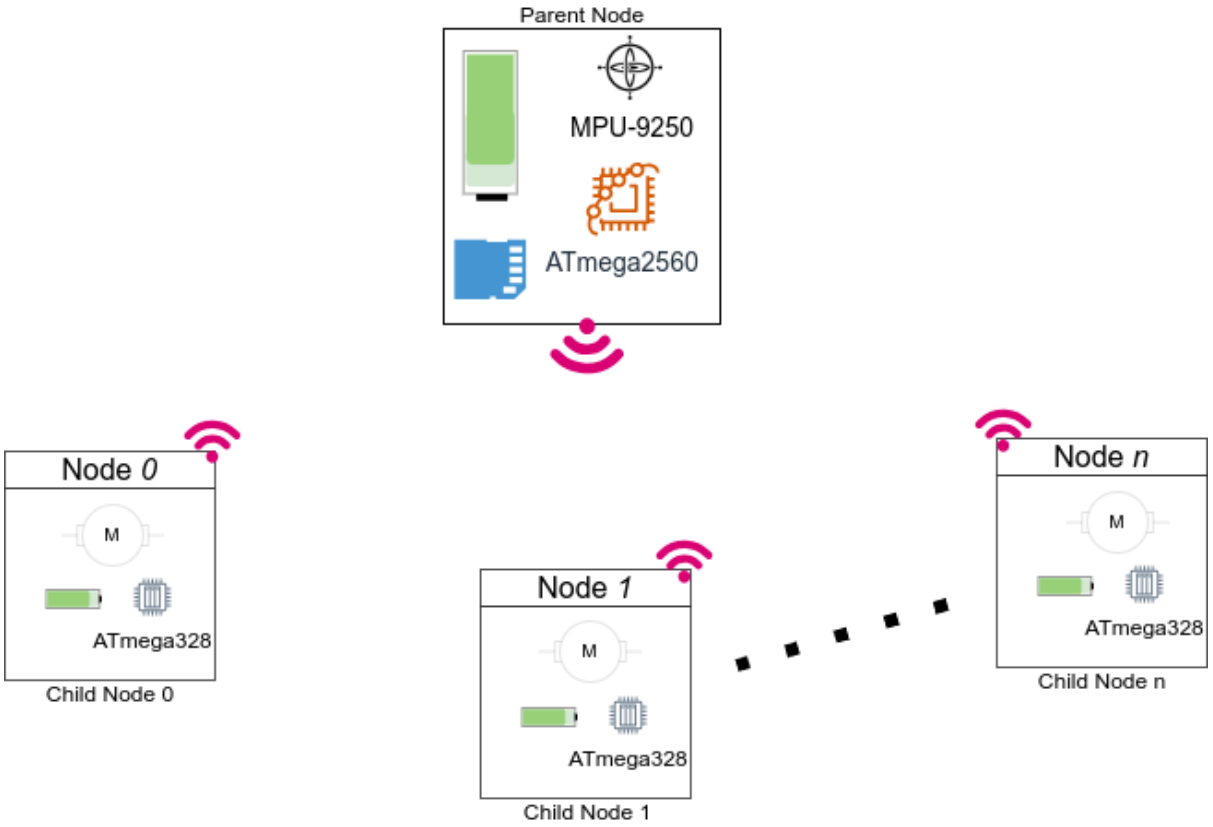


Figure 2.12: An image summarising the architecture and operational principles of Li et al.'s system, ISWIM.

Li et al.'s design is better represented in Figure 2.12[70]. The system focused on being able to provide feedback to users by way of vibration motors while they were swimming, as well as collect rotation data from the user's actions in the pool. As is shown in Figure 2.12, each child node had a processor, feedback device, battery, and communication device. The parent node had the same components as the child nodes with the exception of a feedback device, and the inclusion of storage and an IMU. The admirable feature of this design was its ability to command feedback to each node independently. This system left areas of development such as the addition of IMUs to each node, as well as the transmission of IMU data to a processing device. This system utilised an ATmega328p microcontroller on the nodes, while the parent node had a more powerful ATmega2560 microcontroller and an MPU9250 IMU(9-axis). The nodes communicated by way of an

unnamed 433MHz transceiver.

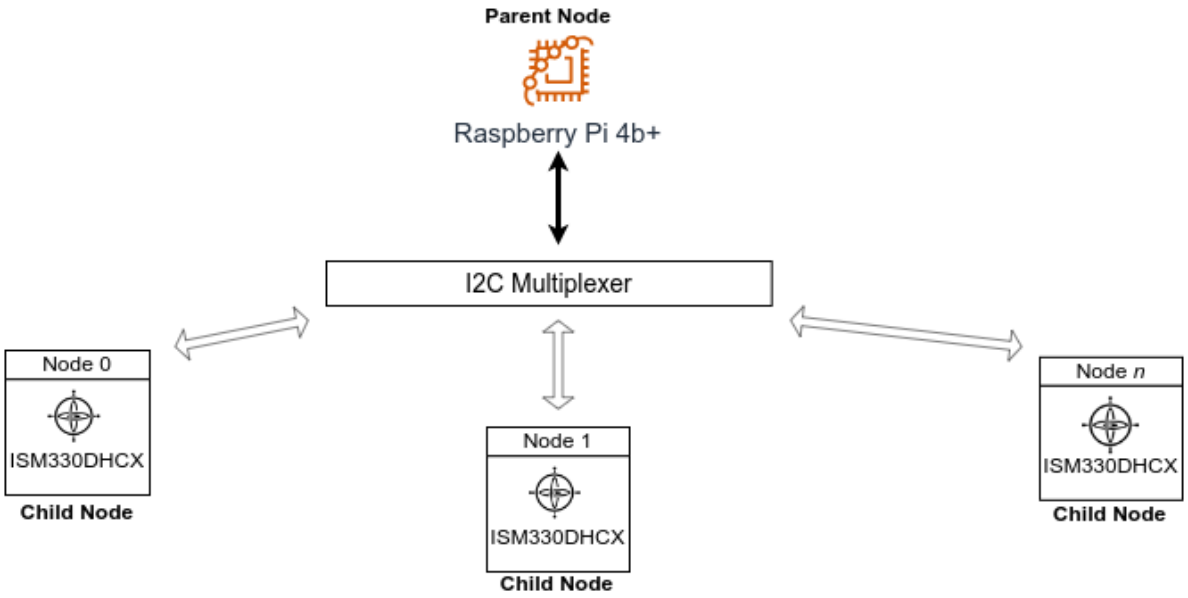


Figure 2.13: An image summarising the architecture and operational principles of Slade et al.'s system.

With regard to Slade et al.'s design of OpenSense RT depicted in Figure 2.13, the ability to process data on the processing node was greatly desired[25]. Looking closer at the system's design, all nodes contained an ISM330DHCX, a 9 axes IMU, and were connected to the processing node, which is a Raspberry Pi 4b+. This processing node connects to the child nodes via I2C (multiplexed as the processor has a single I2C bus) and had a battery attached as well as a single IMU that would be used. Beyond collecting and processing data in real time, this system left room for wireless data transmission to reduce the amount of cables being used.

With respect to software, Slade et al.'s work is of note because it offers a powerful computing resources on the processing node for centralised data processing. The feedback element to this project will be based on Kos and Umek's analysis of biofeedback sensor networks[87]. In this paper, Kos and Umek present some standardised wireless technologies, alongside their bit rates and ranges. One of the goals of Kos and Umek's paper is to advise a reader on selecting tools and technologies to build a system relating to biofeedback. They present three scenarios which may apply to various situations, with the goals of this project being reflected in one of these, including a network topology that may potentially be helpful.

The subject of network topology is more important with regard to software as opposed to hardware due to its profound impact on data packet handling as well as looking at the hierarchy of the system as a whole. With regard to Kos and Umek, Application Scenario

2 from their paper sufficiently describes this project's aims[87]. This scenario advises a star topology, very similar to the single controller to many sensors shown in Figure 2.9. In the same vein, Slade uses the same architecture, running their system on multiple system threads. The first thread ran the program and managed the communications with the sensor and converted the IMU data into quaternions, while the second thread would take this data and use OpenSim's inverse kinematics solver[25] to produce pose estimations.

2.5.1 Summary

This section presented two systems which housed some key takeaways for designing a motion capture system as well as foundation to build upon. The two solutions made by Li et al.[70] and Slade et al.[25] showcased some key ideas which proved to be foundational to a new motion capture system with Li et al. showing the successful implementation of wireless data transmission for processing between nodes for processing and feedback control. In addition to these a very effective feedback device was utilised given the nature of the activity being undertaken. Slade et al. showcased the successful implementation of a similar architecture processing collecting data and transmitting it to an application to estimate poses using inverse kinematics. While there was no feedback device here, there was a very effective monitoring tool available. In both cases, systems were portable and capable of being worn by any user for their intended use cases.

2.6 Summary

This chapter has presented some work exists in the state of the art of motion capture. A general overview of motion capture technologies was first presented, followed by a few underlying principles to the operation of IMUs and their underlying sensors. The methods by which IMU data was then presented followed by a few examples of IMU based motion capture systems, paying close attention to how they process raw data and present this data to users of the systems. The general architecture of motion capture systems was presented relative to both the software and the hardware, as well as how the two relay information to each other.

In exploring this work, some key items were noted that will be further looked at in this dissertation, as well as inform future chapters. These include the potential designs of the Action Tracer system, the various ways of converting IMU data to either Euler or quaternion representation and the revelation of a lack of exploration in data transmission latency. What was more striking was the lack of 6-axis IMU motion capture systems, with most of the systems being explored being 9-axis systems. Advances in motion capture systems which utilise 6-Axis IMUs could help achieve cheaper systems whose accuracy

and shortcomings could be accounted for in software.

Based on this entire review, a base set of requirements can be formulated as overall system requirements which Action Tracer would need to meet to be considered functional and adding to the body of knowledge in motion capture. These system requirements are presented in table 2.2.

Table 2.2: The system requirements necessary to develop and build Action Tracer.

Requirement	**Description**
SR01	The system must be low cost.
SR02	The system must be able to capture human motion in real time.
SR03	The system must be usable from a distance.
SR04	The system may be built upon for various use cases.

SR01 is based on the general observation that 9-Axis IMUs were commonly utilised and using 6-Axis IMUs could help reduce the costs of device. SR02 centres around the two systems examined in Section 2.5 where both systems were able to collect data during an activity and process it in a matter of moments. SR03 was based on a similar notion to that found in SR02, however in this case, the user of a device need not be near a trainer or coach observing the activity or the actual processing device. Finally, SR04 comes from the many systems explored in Section 2.4 which were used in many activities such as swimming, gait analysis and tennis among many other use cases.

Having utilised the information gathered from the review, this chapter will be followed by the research methodology used in order to realise the Action Tracer system's design and testing.

Chapter 3

Methodology

Having completed Chapter 2 where an understanding of the project was gained through looking at some effective systems, this chapter will present how the gathered knowledge was utilised to formulate this research dissertation. As shown in Figure 3.1, the overall development of Action Tracer occurred in four phases: planning and overview, design and theory, implementation and experimentation, and finally, the write-up. These are presented in the following sections in detail, showing the research method followed to design, build and validate the Action Tracer system.

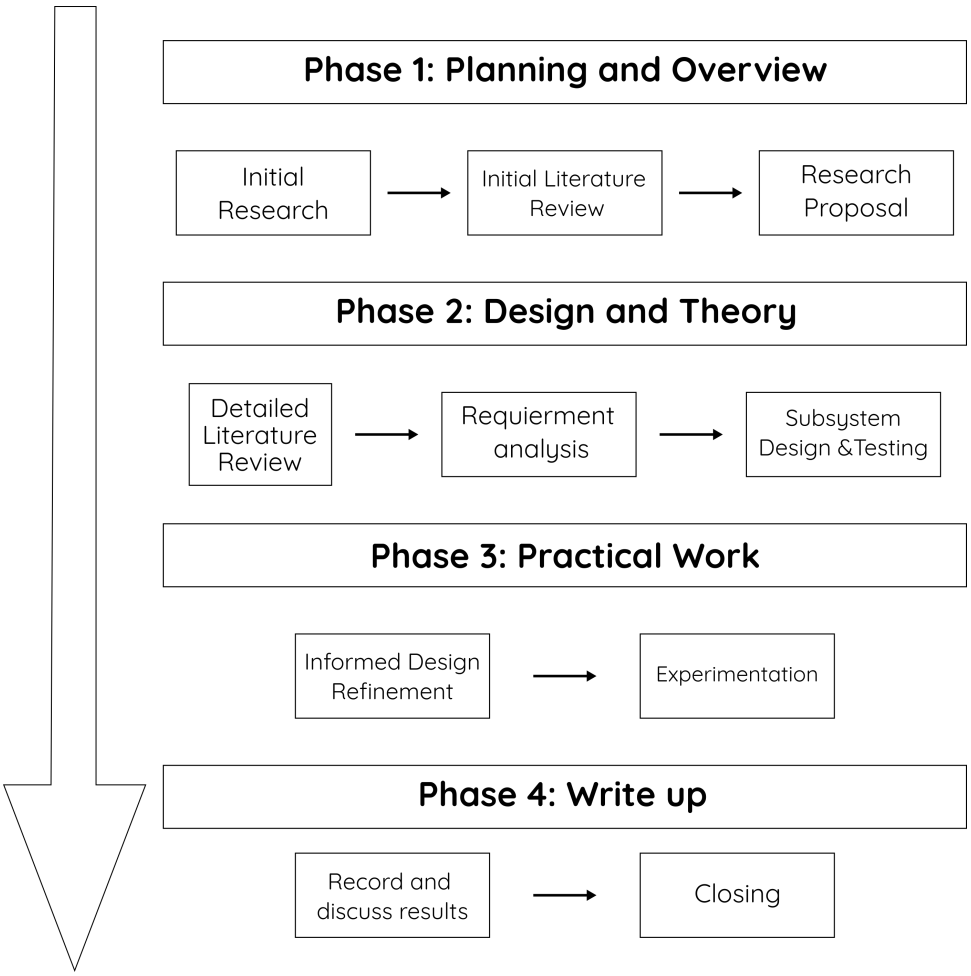


Figure 3.1: A short diagrammatic representation of the phases used in designing, building, and testing the Action Tracer system.

3.1 Planning and Overview

To begin, having chosen to build a motion capture system with feedback, it was key to refine the objectives that this system would meet. To complete this phase, the following process was followed. First, initial research was conducted to refine the problem space. This was followed by an initial literature review used to help gather knowledge about the required subsystems and the relevance of this system in the broader field. Finally, a research proposal was drafted based on this literature review, which laid out the project's primary objectives and scope.

3.1.1 Initial Research

Having completed the planning, the general problem space was explored with relation to this project to obtain an understanding of the concepts and tools required to complete

this project. This stage helped to formulate questions that may be addressed based upon this project's outcomes. It was decided at this stage that the system would be built iteratively, following Boehm's spiral model, an example of which is shown in Figure 3.2[88].

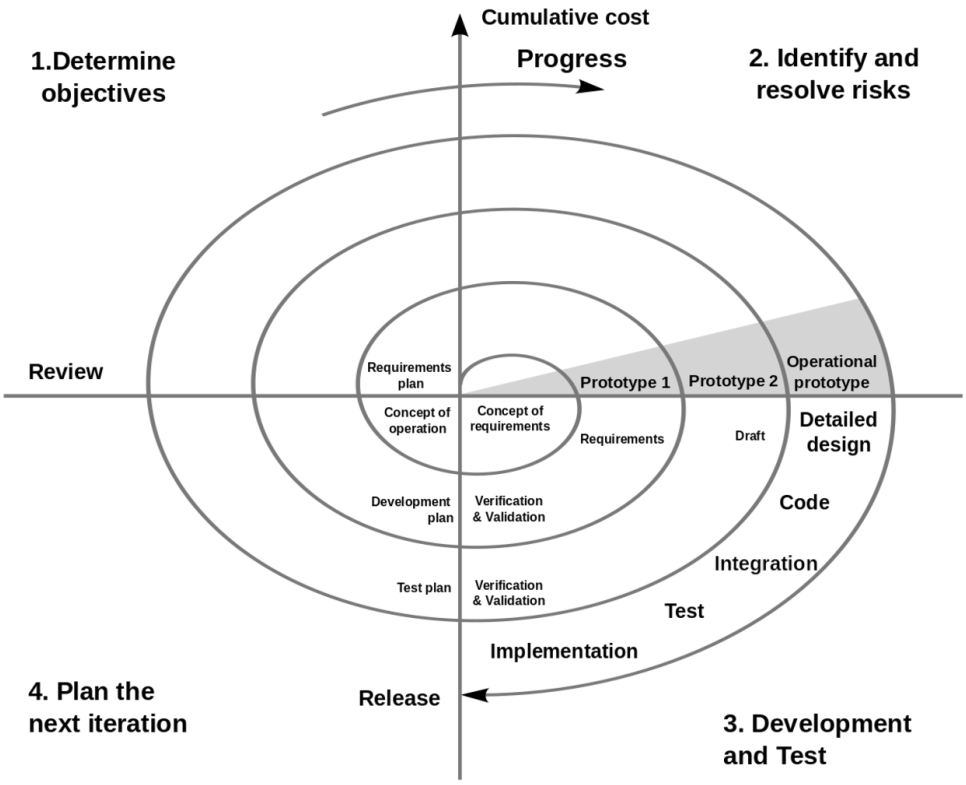


Figure 3.2: Boehm's spiral model, which was followed in the design and evaluation of Action Tracer's hardware and software.

3.1.2 Initial Literature Review

In conjunction with the initial research, a small dossier of documents was closely examined to better define the problems that could be solved by a motion capture system. This was conducted using the funnel method in Figure 3.3.

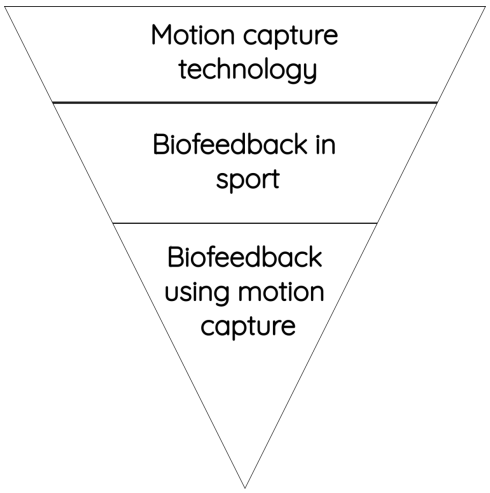


Figure 3.3: The literature review funnel used to define the project's objectives.

First, the general components needed to build a motion capture system were examined, and the base requirements were laid out based on this information. An exploration of biofeedback was then conducted, to learn about its importance in the modern world, as well as examining some of the areas it has been applied in. Finally, the space between biofeedback and motion capture was explored, noting areas that may need innovation or improvement, as well as gathering further objectives relating to this potential gap in knowledge.

3.1.3 Research Proposal

Building from the initial research and literature review, a research proposal containing this gathered information was drawn up, outlining the primary objectives and goals of this project. This proposal provided an introduction to the project, an overview of the literature surrounding motion capture and biofeedback, as well as an outline of the research methodology. Finally, a timeline of the proposed research was presented. Based upon this proposal, clearance for testing was obtained from the relevant ethics committee, with the necessary form being found in Appendix A.

3.2 Design and Theory

At this stage, the initial design of the system to be built was specified based upon the gathered information from the planning phase. The desired outcome of this phase is to come up with a design that will be built and tested, which meets all the main requirements of the system. This was first done by conducting a deeper literature review which helped in outlining the main requirements that the Action Tracer system was supposed to meet,

followed by an in depth analysis of the requirements as they were broken down into various subsystems which would subsequently be designed and tested.

3.2.1 Detailed Literature Review

With an initial idea of what kind of system would be built, from the research proposal, a more detailed review of how such a system would be built was conducted. With the literature and key findings from the initial review guiding, more topics were explored, including similar approaches that have attempted to solve the same problem space. These methods were much more closely examined and specific figures of merit relating to the project's work

3.2.2 Requirements Analysis and Subsystem Identification

Based upon the project planning phase and the detailed literature review, the requirements the Action Tracer system was meant to meet on a which level were outlined. These would serve as the final checkpoints of verifying the success of the project. The set of initial requirements is outlined in Table 2.2. These requirements will be further broken down into further requirements as per subsystem. These subsystems were defined based on the two key factors that will allow the Action Tracer system to be implemented. These are hardware, the physical devices used to collect and process rotation data and the second subsystem is software, the way in which the hardware will be used to attain the main goal.

Based upon the system requirements, the requirements for the hardware based systems are listed in Table 3.1 while the requirements for the software based subsystems are listed in Table 3.2. These were obtained by examining the system requirements as well as the reviewed literature and elaborated upon, in some instances with examples, to create hardware and software requirements.

Table 3.1: The hardware requirements (HW-R) necessary to develop and build Action Tracer.

Requirement	Description
HW01	The device must be cheap to develop
HW02	The design must be easy to reproduce
HW03	The device must be able to obtain rotation data
HW04	The device must be capable of being placed anywhere on the human body
HW05	The device must be usable in sport and portable
HW06	The device must be able to send data to another device
HW07	The device must have a means to communicate to its user
HW08	The device must be capable of being used by multiple people

Based upon the hardware requirements, the following subsystems were identified and will be further expanded upon in Chapters 4 and 5,

1. Embedded Platform
2. IMU
3. Feedback
4. Node Design

Table 3.2: The software requirements (SW-R) necessary to develop and build Action Tracer.

Requirement	Description
SW01	Collected data must be retrievable from any device
SW02	Data transmission should be seamless
SW03	Developers must be able to build upon this system
SW04	Achieve a configurable system

Based upon the software requirements, the following subsystems were identified and will be further expanded upon in Chapters 4 and 6,

1. Data collection
2. Data transmission
3. Processing
4. API

The overall system requirements will be further assessed and broken down in Chapter 4. This chapter will inform Chapters 5 and 6 which will further look into the Hardware and Software requirements and subsystems respectively.

3.2.3 Subsystem Design and Testing

Having identified the various subsystems that make up the Action Tracer system, the next task to be completed is the individual system design. This was done by first expanding the requirements from Tables 3.1 and 3.2 and creating design specifications. Depending on the subsystem, these specifications were used to design modules that could be tested and integrated into the bigger system in various ways, however the end result was an independent and integrable subsystem.

The hardware based subsystems underwent a process of first having their individual components selected, followed by some testing to ensure they meet their respective specifications. This was followed by bringing the subsystems together and producing a single device ready for the development and testing of the software based subsystems.

With regard to the software based subsystems, informed by the hardware based subsystems, the first task to be done was selecting a software design pattern and programming language to allow a more standardised approach to creating the system. The selected design pattern was Object-Oriented Design(OOD)[89] and the selected programming language was C++. Figure 3.4 shows the key principles of OOD.

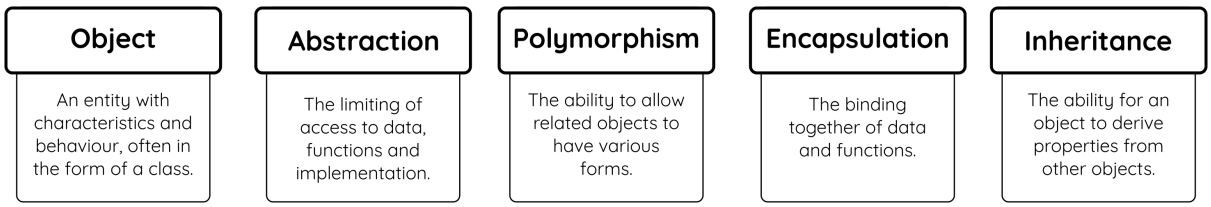


Figure 3.4: The core foundational principles of Object-Oriented Design in software.

This was followed by using the selected design pattern and programming language to design the various subsystems according to their specifications, allowing for independent subsystem testing and integration into the overall system.

As each subsystem was being developed, there were minor tests being done to ensure their functionality, following the principles of the spiral model. In order to ensure the success of the fully integrated device with all the subsystems was functional, there were various focused tests that were conducted during each development loop. At the end of the loop, a major test was conducted which would integrate the smaller changes from the previous stages of development. The minor tests were done independent of human motion capture, while the major tests were implementing the system in the context of the project's aims, motion capture and feedback. This experimentation is covered in Section 7.1 of this dissertation.

3.3 Implementation

Having obtained various working subsystems, the next phase involved combining them to make the whole Action Tracer system. While nearly seamless, this process required various iterations and adjustments, which led to minor refinements to the design. With this completed and a functional system, the system tests could be conducted, which included benchmarking the system and conducting use case tests.

3.3.1 Informed design refinement

In order to maximise system performance after building the minimum system, further investigation into the nuances of the design needed to be completed to improve upon it and its subsystems. This was done by way of further investigations into multithreaded programming, API design patterns and data serialisation. Based on these 3 concepts, the data output rate and data accessibility were improved to both match some of the systems that would be used for comparison.

A particular emphasis was placed on the addition of serialisation into the system, as this would allow a test of the customisability of the system while offering a new and useful feature. This first required acquiring knowledge of the system to be implemented, and was swiftly followed with integrating it into the design and its relevant subsystem.

3.3.2 Experimentation

In order to assess the functionality of this system, various tests need to be conducted. These will be done, first, to verify that the requirements that were set out to be achieved were attained. The secondary objective of this testing was to benchmark the system against an established motion capture system, which would allow one to gain an understanding of the Action Tracer system as a motion capture system. The final, stage of testing will explore the system in some scenarios where it may be used as has been explored in the literature.

First, the verification of the system's functionality was conducted by comparing specific figures of merit obtained from other motion capture systems which were examined in the review. Some of these elements were obtained by examining the hardware specifications of the IMUs used in the inspected systems, while others were based on the results of the research outputs examined. The attributes to be monitored are the following:

1. data output rate (sampling frequency) (between 2Hz and 30Hz)

2. accelerometer and gyroscope full scale range ($FSR_a = 2, 4, 8g$, $FSR_g = 2000^o/s$
3. node count (>= 3 nodes)
4. latency (< 2 seconds)
5. accuracy and precision (5% of the true value)

These metrics were tested initially according to their subsystems as shown mentioned in Section 3.2.3 and finally for the entire integrated system. Power consumption will not be measured as the system is running on an operating system, which has various other tasks running, meaning it would be difficult to accurately assess the true consumption of a single new task being run.

Following tests done to ensure the various subsystems were working, tests were done to check that the system was fully functional when put together and that data could be obtained from the Action Tracer system. This was first done by collecting data between two IMUs to observe if the angle tracking was working sufficiently. This was followed by testing the API created for the system and ensuring its usability. Finally, the system was tested to capture motion and compare against an application created to process and utilise the given data from the Action Tracer system. Applications were made to accompany the API testing as well as to present the motion from the system while respecting the API. Alongside this, the method by which angles will be obtained through one of the applications will be discussed.

The final testing phase was benchmarking the Action Tracer system against a more established motion capture system, an open source optical, markerless motion capture system was selected as it was the most readily available and affordable solution. For this, MediaPipe was selected, and the same figures of merit were tested once again between both systems, and these results shared. These tests were conducted with users who would wear the system and were asked to complete various tasks and have the data from these tasks collected by the two systems. These tests were done with volunteers selected according to the ethics committee's criteria, and with users being given the information sheet and signing the form in Appendix A and A. The tests relating to this section are found in Section 7.3.

3.4 Documentation

Having completed all the implementation and experimentation, the data was consolidated, reviewed and presented. The steps necessary to do this are discussed further in this section.

3.4.1 Record and discuss results

This stage of the project allowed for the final results to be presented in a digestible manner, as well as minor retests that may have been needed to ensure the validity of some results. This would also allow a definitive conclusion to the system's validity and viability. Alongside these, recommendations to improve the system were also presented based upon this design. The work covered at this phase of the project as well as the interpretation of the results is given in Chapter 8.

Chapter 9 will close the project as the conclusion, where the major takeaways from this project are presented. Alongside these key points, further improvements to the Action Tracer system as well as the benchmarking will be presented.

3.4.2 Closing the project

The final step in this project is the submission, editing and correcting of the dissertation.

3.5 Summary

This chapter presented the methodology used to build the Action Tracer system, as well as the principles behind how it will be assessed. This was done by following four stages of development: Planning and Overview, Design and Theory, Practical Work and the Write Up. In this process, the chapters to follow in this dissertation were presented, with Chapter 4 housing a survey of the full system's design. This will be followed by Chapters 5 and 6, which will provide deep dives into the software and hardware subsystem's analysis and design decisions and constraints. Chapters 7 and 8 will respectively house the experiments and tests run to assess the system and the results from these assessments. Finally, the conclusion to this project will be found in Chapter 9

Chapter 4

System Design

This chapter provides analysis of the Action Tracer's system level requirements, as well as provide a foundational understanding of the proposed system's make up and how it's different subsystems will interact with each other. The requirement's analysis will be based on the system requirements laid out in Table 2.2 and will show how they were decomposed into their various subsystems. This chapter aims to inform the direction of both the hardware design and the software design, which is found in Chapters 5 and 6.

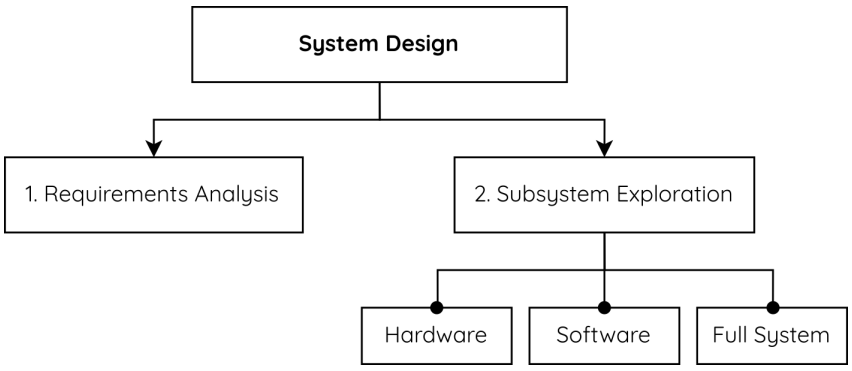


Figure 4.1: An overview of the key topics that will be covered in the system design chapter.

4.1 Requirements analysis

This section will provide a break-down of the system requirements listed in Table 2.2 to produce further refined subsystem requirements. This analysis will give insight into the various roles and responsibilities each subsystem will have in the Action Tracer system. These various requirements directly link to the UML Use Case diagram provided in Figure 4.2.

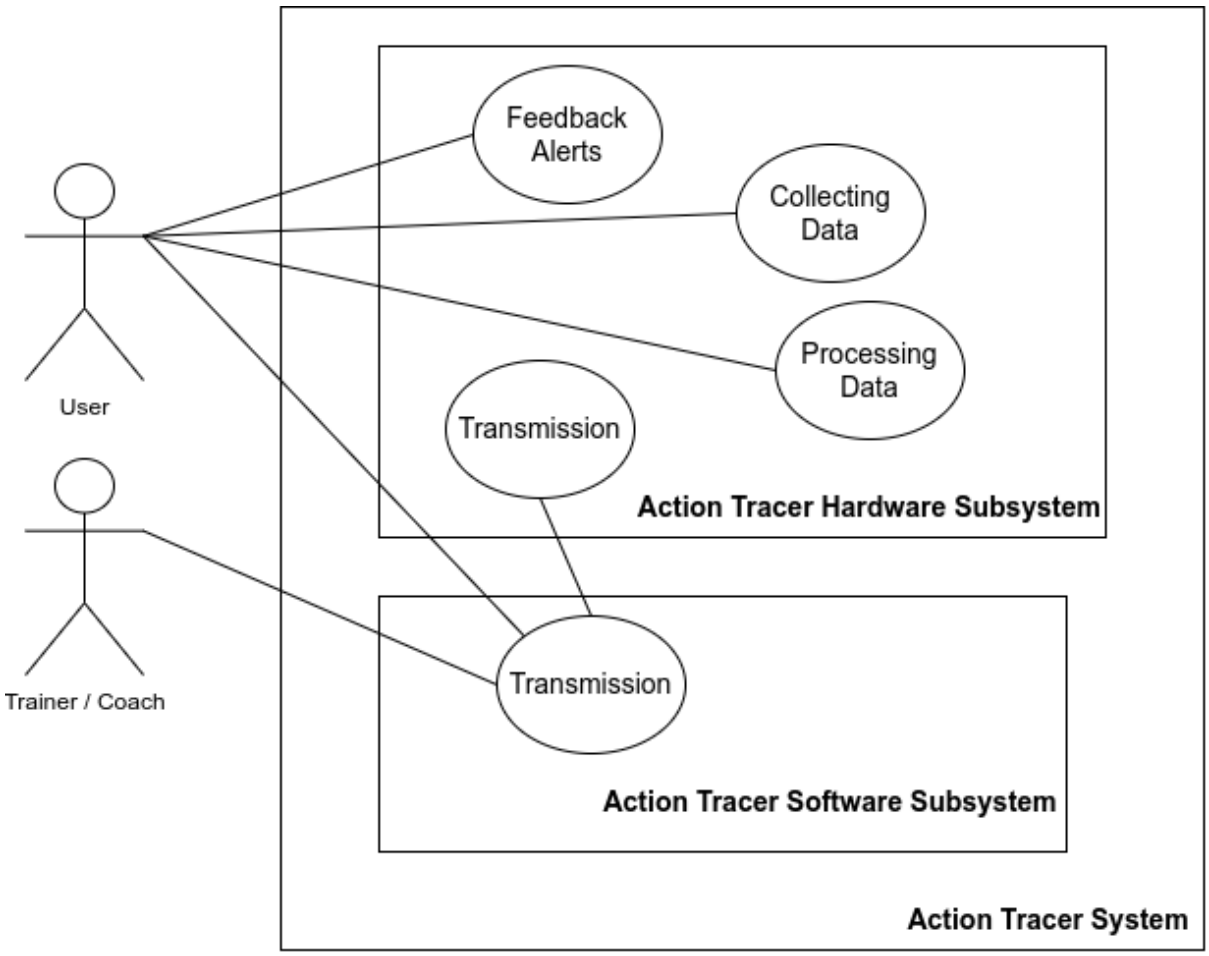


Figure 4.2: A UML use case diagram detailing the key features and functionalities of the Action Tracer system.

4.1.1 SR01: The system must be low cost.

Given that most motion capture systems are expensive, it would be ideal to create a system that is much more affordable. While one aspect of producing the system is its cost, the ease of reproducing the system is a crucial factor that should be considered in the design. Therefore, the cost of the system is based on both how much money is needed, and the equipment needed to support the manufacture of this system. Wei et al. showed the effectiveness of low-cost motion tracking systems in healthcare, aiding in the viability of this project[80].

4.1.2 SR02: The system must be able to capture human motion in real time.

While the main questions being addressed by this system are based on training or rehabilitation, the heart of the system must rely on some form of data being collected. In the case of Action Tracer, orientation data based on IMUs will suffice. Given their size

and capabilities, multiple sensors may be used and placed all over the body, providing multiple points of analysis, as was shown possible by Kong et al[72].

It is preferred if the system would be able to deliver this data to an observer, be it a trainer or therapist, in a relatively short amount of time.

4.1.3 SR03: The system must be usable from a distance.

Given the potential applications of this system include sports training, the system must take into consideration that athletes in some sports are not able to use a device wired to a computer and require freedom. This would point to data needing to be wirelessly transmitted to the processing device. Given that some systems allow a tether to be connected, freeing the athletes would require a reliable and efficient data transfer method.

With data being wirelessly transmitted to trainer's devices, it would be ideal to have the chosen trainer's device to work independent of its make or constraints, as long as it has the appropriate wireless communication standard available. This independence will also aid in reducing the overall costs of the system, allowing users to not need further hardware to support the Action Tracer system.

4.1.4 SR04: Achieve a configurable system

While the primary goal of the system is to capture motion, various activities will require capture in different parts of the body. Given this limit, it would be best if the system's number of devices were variable. Alongside this, different capture parameters within the IMUs need to be adjustable, as different sports and activities have different rates of movement.

Another consideration is that this system should be able to help a user learn an action or activity on their own, making use of the system's biofeedback system. This feature as well as others must be however able to be used as is required or completely left out of an implementation. One of the target users of this system will be application developers who can create custom implementations of their needs on the embedded platform based on its given API.

4.1.5 Hardware and Software Requirements

Table 4.1 shows the hardware and software subsystem requirements based on the analysis conducted in Sections 4.1.1 to 4.1.4. Both subsystem's requirements are given in this

table but are denoted with requirement identifiers, with **HW** for hardware subsystem requirements and **SW** for software subsystem requirements.

Table 4.1: The hardware and software requirements necessary to develop and build Action Tracer.

Requirement	Description
HW01	The device must be cheap to develop
HW02	The design must be easy to reproduce
HW03	The device must be able to obtain rotation data
HW04	The device must be capable of being placed anywhere on the human body
HW05	The device must be usable in sport and portable
HW06	The device must be able to send data to another device
HW07	The device must have a means to communicate to its user
HW08	The device must be capable of being used by multiple people
SW01	Collected data must be retrievable from any device
SW02	Data transmission should be seamless
SW03	Developers must be able to build upon this system
SW04	The system must be configurable

While these requirements refine the given system requirements, they will be further dissected in their respective subsystems. There, each requirement will be analysed to produce specifications for its subsystem.

4.2 System Break Down

In this section, an explanation of how the Action Tracer system's subsystems will be presented. The individual roles of each subsystem will be outlined, as well as their interconnected nature. This will allow the various dependencies to be more clearly presented. These two subsystems will be tied together in Section 4.2.3.

4.2.1 Hardware Subsystem

This section will cover work relating to the hardware of the Action Tracer system. Given that the hardware will do most of the interaction with a user, this will be broken down into it's various components that will come together to form the subsystem. These are clearly shown in Figure 4.3.

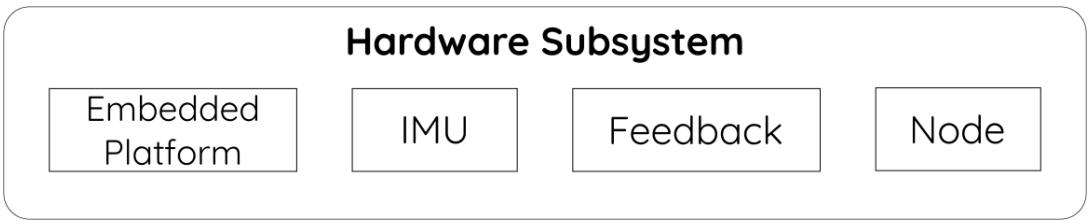


Figure 4.3: The four components that make up the hardware subsystem.

A brief explanation of the importance and functionality of these components is provided below. These will be further expanded upon in Chapter 5.

Embedded platform
This component captains the hardware subsystem as it dictates the possible interfaces of the overall system. This includes the type of IMU, feedback system, as well as the capabilities of the Software subsystem. The work done by the selected device will be central in the direction of the project.

IMU
The IMU will dictate the quality and reliability of the data being collected and sent to the user. The selected device will be used to collect data as well as use any of its internal features to improve the data collection.

Feedback
Given the Action Tracer system is to be interactive, this component of the hardware subsystem will be important in making the system two-way. This system will communicate back to the user on behalf of or alongside a trainer.

Node
While not a fundamental part of the system, the node component of the hardware subsystem will be used to house the IMU and feedback components. This will be the part of the system replicated the most to provide multiple data collection points.

4.2.2 Software Subsystem

This section will present the various components that make up the software subsystem. The goal of this subsystem is to make use of the features and capabilities of the hardware subsystem for the benefit of the Action Tracer system. Figure 4.4 showcases the various components that make up this subsystem.

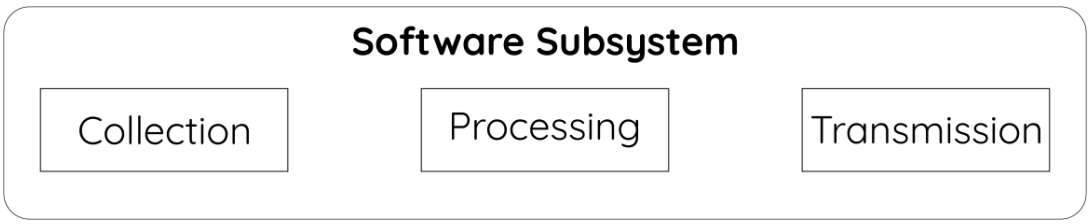


Figure 4.4: The three components that make up the software subsystem.

A brief explanation of the importance and functionality of the various components in this subsystem is given below. Further details of this subsystem will be expanded upon in Chapter 6. The work done in this subsystem will include the development of two APIs. The first will be used by the collection component of the subsystem, while the second will be used to control the entire system.

Collection
The collection component of the software subsystem is used to collect data from the IMU and make it available for the embedded platform to process. This aspect of the system is vital, as the method by which the IMU is connected to the embedded platform will cause various limitations. This will be controlled by an API which will be developed for this specific IMU.

Processing
The processing component of the software subsystem is used to predigest the collected data before it is sent to the user, as well as providing the user with feedback should it be needed.

Transmission
The transmission component of the subsystem is used to send data from the embedded platform to the connected trainer device. Depending on the chosen transmission medium and protocol, this subsystem will be vital in ensuring real time data transmission. Alongside this, the data must be packaged in such a way that allows users to utilize the data in existing applications. A key focus will be placed on the ability to use collected data in programs in much the same way as Raghavendra et al. managed to achieve[86].

4.2.3 Full System

Given the subsystems outlined in Sections 4.2.1 and 4.2.2, a full system description may be outlined. Figure 4.5 provides this outline in the form of a UML deployment diagram.

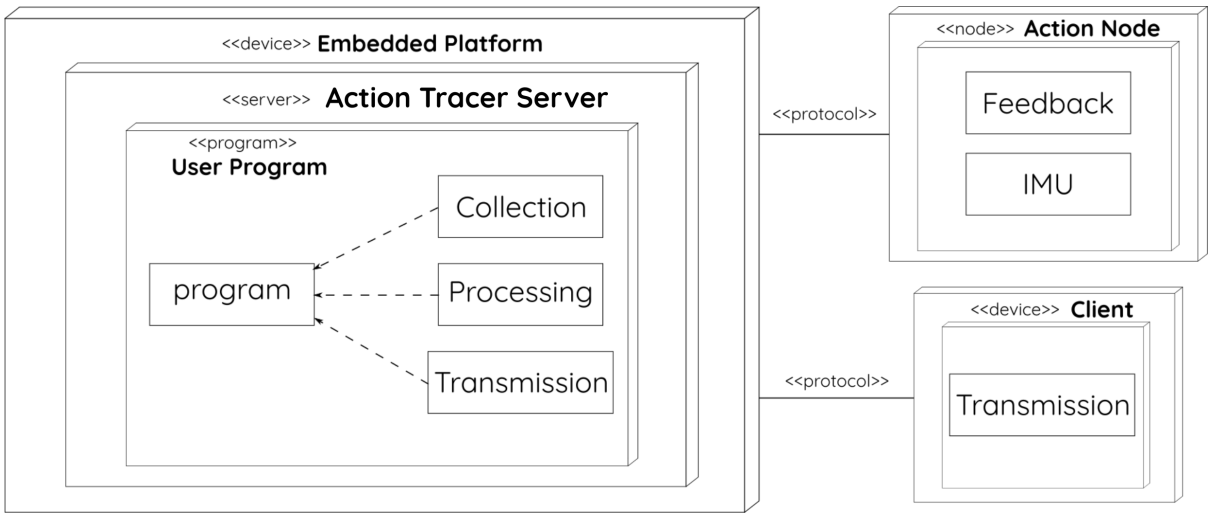


Figure 4.5: A UML deployment diagram of the entire Action Tracer system, showing the main subsystems and their components.

The Action Tracer system will rely on a custom program written by the user, which will interact with the embedded platform to obtain data depending on their needs. The client device will need to comply with

the Action Tracer's transmission subsystem protocols and standards. The Action Node is a component that will be replicate multiple times with the same components on it for each point on the subject to be tracked.

4.3 Summary

This chapter provided an overview of the Action Tracer system and the various components that need to developed for the system's full functionality. This was done by further analysing the system requirements and outlining the components and roles of the hardware and software subsystems. Finally, the full system was outlined and presented to visually represent the system.

Chapter 5

Hardware Design

This chapter will cover the means by which the components of the hardware subsystem were chosen for the Action Tracer system, as described in Chapter 4. First, a requirements analysis will be performed to derive system specifications. This will be followed by a closer look at the various subsystem components that fall under the hardware of the Action Tracer system. This will be done by first conducting a component selection and concluding the chapter with the proposed design, which is the amalgamation of these subsystem components.

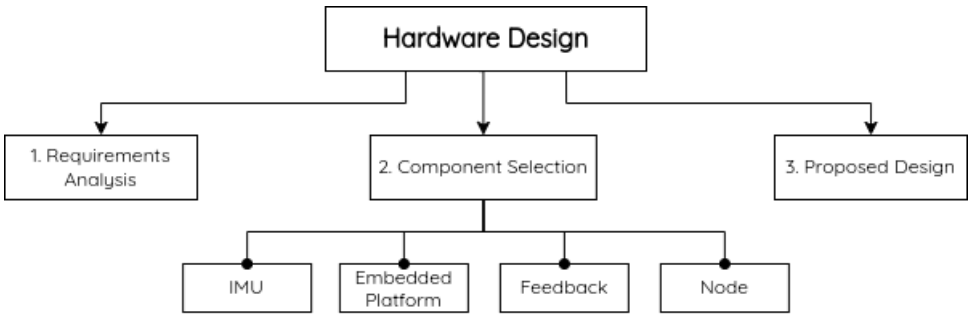


Figure 5.1: An overview of the key topics that will be covered in the hardware design chapter.

5.1 Requirements analysis

To adequately address each design aspect of Action Tracer's hardware, each requirement from Table 3.1 will be analysed and have specifications drawn up from it and presented in Table 5.1. In some cases, past literature will be used to inform specifications and design decisions.

5.1.1 HW01: The device must be cheap to develop

When considering the cost of IMU motion capture systems, there is a wide range of factors that cause the prices to be generally high. While some proprietary systems such as the Rokoko motion capture suit[90], or XSens' Dot and SDK[79], have high costs due to the work done by their creators, some systems, even if

they are open source and readily put together may prove to be costly due to their individual components and assembly costs. An alternative to this may be to draw up a new system based on hand selected and bespoke components to meet the project requirements. This would however prove to take more time and be more costly. The only remaining solution would be to use off the shelf components. In most cases, these modules and components are mass-produced, reducing the cost of their manufacture.

5.1.2 HW02: The design must be easy to reproduce

One of the main factors relating to reducing the overall cost of this system would be its ability to be rebuilt by any user. The first step in this direction would be the ability to easily access the design files and code required to get this system functional. To aid this, the entire project, from design files to code, will be made open source under the MIT License, links to which may be found in Appendix B.5 and B.2. A further consequence of making the design easy to reproduce is in line with HW01 is the need to use common off the shelf components. Not only should these components that are widely supported and easy to obtain, if possible, worldwide.

5.1.3 HW03: The device must be able to obtain rotation data

Based on the literature that has been examined, the common devices used in similar systems are IMUs. In some cases, users used separate gyroscopes, accelerometers, and magnetometers. Based on these requirements, the first consideration to make would be relating to the number of axes to have for this system. Given that both an accelerometer and gyroscope are needed to obtain yaw, pitch and roll, a magnetometer may be omitted. The second consideration relating to this design would be the full scale range (FSR) for both the accelerometer and gyroscope. In most cases, accelerometers and gyroscopes have a swappable FSR that a user may vary depending on use case. A floor must however be set, and this will be done by comparing the FSRs from Table 2.1. Tables B.1 and B.2 show the results of this comparison for both accelerometers and gyroscopes, respectively. In most systems, accelerometers had FSRs of $\pm$2g, $\pm$4g and $\pm$8g while most gyroscopes had an FSR of $\pm$2000dps. Given these figures, the chosen IMU must have available, at the very least, the above-mentioned FSRs available.

5.1.4 HW04: The device must be capable of being placed anywhere on the human body

When mounted, each of the IMU mounts should be able to attached to a trackable marker point on the human body. Due to the difficulty of mounting a node on an articulated joint on the human body, they will be mounted on significant bone sections. Were this placement to be based on a model similar to that used by Slade et al.[25], it would be clear that the node needs to fit on the wrist, as well the thigh of various individuals. To aid this, the size of the sensor plate will be big enough to fit a wrist, in the same way a wristwatch may do so. Given this direction, a maximum surface area of 30 cm^2 will be set.

Given this size constraint, to suit various body types during usage, a reusable and adjustable harness is required for each node, including the main node. This would mean that the design of the node's housing must include a mounting or harness point for the strap.

5.1.5 HW05: The device must be usable in sport and portable

Given the variety of various sports and activities, this requirement is very closely linked with HW03, as they both require equipment that is capable of acquiring both fast and abrupt motion as well as slow and precise movements. Therefore, an IMU with a variable FSR would be needed to accommodate such motion, however, a second factor to be considered is the sample rate of the IMU. The sample rate influences the resolution of motion between two capture cycles, better known as frames. With an increase in sample rate, often, IMU signals tend to accumulate more noise. In some activities, however, a lower sample rate would be sufficient, as shown by Kos and Umek who sampled swimmers at 2Hz while Slade et al used a sample rate of 30Hz[2, 25]. Therefore, an IMU with a variable sample rate is required.

Due to the demands of sport, the nodes may need water proofing as athlete's bodies release sweat, which may interrupt the functionality of the system, potentially rendering it useless. In the same vein, it would be ideal to have a system that can be worn away from a wired power source. This portability would allow for a wider array of use cases in sports such as cricket where the players are scattered over 14 000 m^2. Ideally, the system should be powered by a standard capacity battery that is commonly available.

5.1.6 HW06: The device must be able to send data to another device

Due to the broad nature of this requirement, it needs to be examined from different perspectives. The first communication layer to be examined is that which is between the IMU and the embedded platform. A common interface is required. Most commonly used are CAN, I2C, SPI and UART. With a desire to minimise cost and help with portability, the fewer the physical wires, the better. To aid this, a simple protocol is required, ruling out CAN, and the selected protocol should be able to address each node, UART would not be sufficient. This leaves SPI and I2C as potential options.

The second communication layer to examine is that which exists between the embedded platform and the user. With regard to this, multiple options exist, and would affect the embedded platform chosen as well as potential add on peripherals to enable such communication. With options ranging from Zigbee to Bluetooth and Wi-Fi (IEEE802.11 and its various variations), the possibilities are nearly endless for an interface. To reduce the costs, a standalone embedded platform will be used that has the required protocol will be used. To aid in development, a common and well-supported interface will be recommended, such as IEEE802.11.

Finally, the method of data collection and transmission used may require verification after a user has used the system. This would be in line with most of the systems that have been analysed in Table 2.1. Given this requirement, the embedded platform should have attached storage that may be used to store captured data.

5.1.7 HW07: The device must have a means to communicate to its user

A common feature lacking in most IMU motion capture systems is the ability to provide feedback to a user. This would be ideal if trying to have a user again a specific range of motions or actions, or as a means of correcting form. If a user was to be guided in such a way, they would be able to utilise the system as a corrective means outside a clinic or training session. Such feedback may be attained by means of optical or audible means, such as an LED or a buzzer.

5.1.8 HW08: The device must be capable of being used by multiple people

Due to the guidelines given by HW04 and HW05, it would be ideal to have a system that may be swapped from individual to individual. The first challenge that is introduced is the cleanliness of the nodes. Due to sweat's unhygienic nature and lack of common transfer between individuals, ideally a water-resistant material should be used to line nodes, while also ensuring a firm grip on the attached body member.

With body shapes and sizes being generally dissimilar, it would also be ideal to have a system that is able to adapt to these changes every time the system is transferred from one user to the next. This would mean that a system that can be calibrated is required. Calibration would be the first step taken after any user puts on the system so that whatever the use case, data collected and used is specific to their body and its constraints.

5.1.9 Hardware Specifications

Table 5.1: The hardware specifications (HW-SP) necessary to develop and build Action Tracer.

Specification	**Description**
HW01-SP01	Only off the shelf commercial components will be used
HW02-SP01	Publicly available devices will be used
HW02-SP02	The final design will be open source
HW02-SP03	Well supported, and common platforms will be used
HW03-SP01	The system must utilise an IMU with at least an accelerometer (±2g,±4g,±8g) and gyroscope(±2000dps)
HW04-SP01	The system must have a sample rate between 2Hz and 30Hz
HW04-SP02	The physical system must be water-resistant
HW05-SP01	The system must be capable of being strapped
HW05-SP02	Each node must be smaller than 30 cm^2
HW05-SP03	The system must run off a battery
HW06-SP01	The IMU and embedded platform must have an I2C interface
HW06-SP02	The embedded platform must have an 802.11 Wi-Fi interface
HW06-SP03	The embedded platform must have on-board storage
HW07-SP01	The design must accommodate for a buzzer and LED
HW08-SP01	The system must be quick to calibrate
HW08-SP02	The system must be easy to clean

5.2 Component Selection

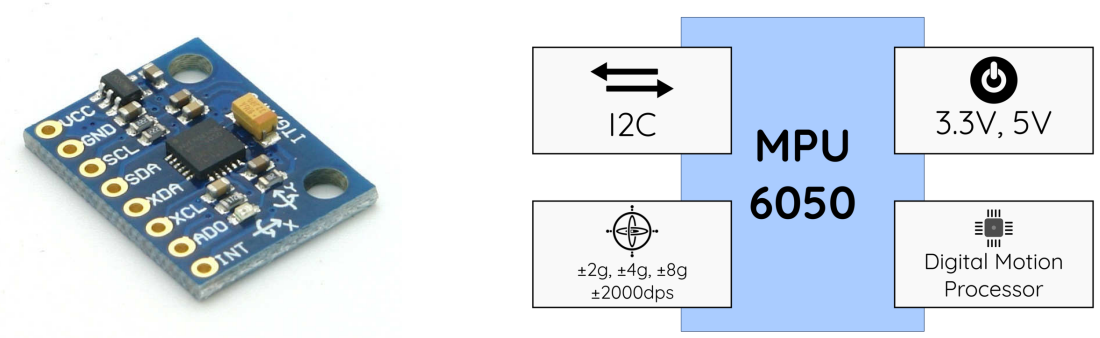
Having expanded on each of the given requirements and condensed them into specifications as shown in Table 5.1, specific devices could be chosen for each node and the main system node. In aiding with this stage of assessment, the testing done to select these devices is outlined in Section 7.1.

When selecting each of the components, multiple factors were considered, including cost and availability. Given that this project was undertaken during the Covid-19 pandemic, there were notable component

shortages affecting all spheres of development, limiting the options available for users. Each aspect will be presented below, as well as the options that were available at the time of selection.

5.2.1 IMU selection

To begin, all 9-axes IMUs were excluded from the search, as the magnetometer could be excluded from the selection. With a limited search criteria of 6-axes IMUs, the cheapest, locally available IMU would be selected to allow rapid development. The IMU found was the MPU-6050 made by Invensense as shown in Figure 5.2[91].



(a) An overview of all the key capabilities of the MPU-6050 in relation to this project

(b) A picture of the MPU-6050 module to be used, as well as its interfaces.

Figure 5.2: The MPU-6050 selected for the design of the Action Tracer system's node.

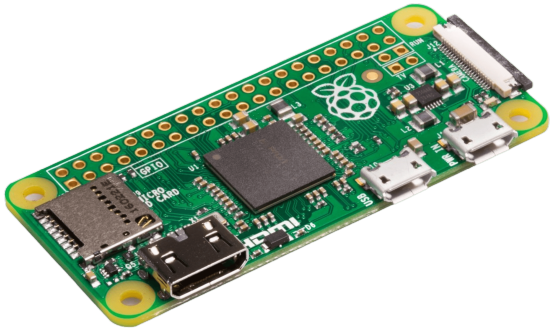
The MPU-6050 fully satisfies the design requirements laid out for this system. In addition to this, the MPU-6050 features an on-board Digital Motion Processor(DMP) which is used to offload some processing load from a host processor. This allows the embedded platform to directly receive both raw IMU data and rotation data which has been computed on the IMU. This DMP implements proprietary algorithms and solutions to aid the user, however this device is widely documented and supported with various libraries in many programming languages.

The MPU-6050 is an ideal candidate for this system as it houses an I2C interface, with two changeable addresses. Alongside this, it has the required FSRs and can be supplied with 3.3V and 5V. A quirk with this device however is its ability to operate it's I2C bus similarly to an SPI bus. By toggling the address to the second, the device may be controlled and ignore the same commands when toggled back to the first address. The address toggle mode acts similarly to the SPI chip select. This avoids the need for a multiplexer, further reducing potential costs.

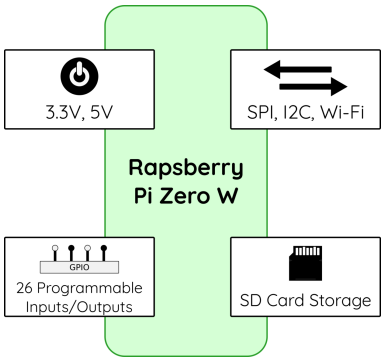
5.2.2 Embedded platform selection

Given the restriction to devices with an in built Wi-Fi interface, the search for an embedded platform was greatly simplified, as options for such embedded platforms ranged from microcontrollers to microprocessors. Considering Slade et al's work[25], the concept of multiple computing threads may be of great benefit to this project. Over and above this, there are great benefits in having on board storage that does not need additional firmware to operate. These conditions point towards a microprocessor, which is also capable of executing instructions much faster than a microcontroller. The most popular variety of microprocessors are found on a Single Board Computers(SBC). One of the leading SBC manufacturers

being the Raspberry Pi Foundation, it would be ideal to use their low cost, greatly documented SBCs for data processing. Due to the need for portability, the Raspberry Pi Zero W was selected for this project[92].



(a) An overview of all the key capabilities of the Raspberry Pi Zero W in relation to this project



(b) A picture of the Raspberry Pi Zero W single board computer, to be used, as well as its interfaces.

Figure 5.3: The Raspberry Pi Zero W selected for the Action Tracer system's processing node.

The Raspberry Pi Zero suits this project's requirements, as it has a Wi-Fi interface, allowing it to connect to a local network and communicate with a user's monitoring device. Alongside this, it has I2C and SPI interfaces, allowing it to connect to any peripherals with these interfaces. With power outputs of 5V and 3.3V, no external circuitry will be needed to connect to most peripherals. (An additional benefit to using this embedded platform is having the I2C lines pulled up internally, reducing the need for external circuitry.) Finally, the Pi boot from an SD Card that is inserted, which will act as both the boot disk and storage, allowing for variable storage sizes. Finally, due to a great deal of compatibility with other Raspberry Pi boards, support, and documentation for this device is abundant. The above-mentioned reasons greatly support the use of this device, including its low price point relative to other microprocessors. With respect to power, this Pi may be powered using a standard 5V power bank due to its low power consumption.

5.2.3 Feedback device selection

With respect to feedback. During usage, it would be ideal for a user to take note when the device is on and operating. Given that no external display will be used, a small visual cue may be used to show that the system is powered and idle, or is collecting data. This would be ideally done with an LED showing when a device is being polled for data, used in conjunction with an IMU that has an indicator LED for its own power. Apart from mode acquisition, another form of feedback that may be implemented relates to specified motions being replicated sufficiently. In this case, a haptic or auditory cue would be ideal, as this may be localised to a node. Another method of doing this would be combining an auditory cue with a visual cue, however this may take time to decipher if many nodes are in use but uses fewer components, which reduces costs. This mixed approach would therefore be ideal, having a single LED per node and a single buzzer on the main node.

5.2.4 Node design

With a maximum of 30 cm^2 to utilise per node, a simple rectangular shape would be ideal. Because this step is influenced by the size of the IMU module being used, its dimensions will also be put to question. The dimensions of this module are 21.2mm x 16.4mm x 3.3mm, giving a surface area of ~3.5 cm^2. In addition to this, each node will have breakout points and an LED, which require very little space. With respect to straps, hook and loop straps will be used to make the device more adjustable. This will be used together with neoprene to make a strap that will interact with the point of attachment.

5.3 Proposed design

Given the selected components and their potential to come together as the Action Tracer system hardware, this section will show the iterations of the hardware and showcase the hardware to be assessed as well as its interactions. Figure 5.4 showcases the interactions of the final hardware. A potential flaw in the I2C communications of this design was noted and resolved and will be presented in more detail in Section 6.3.1.

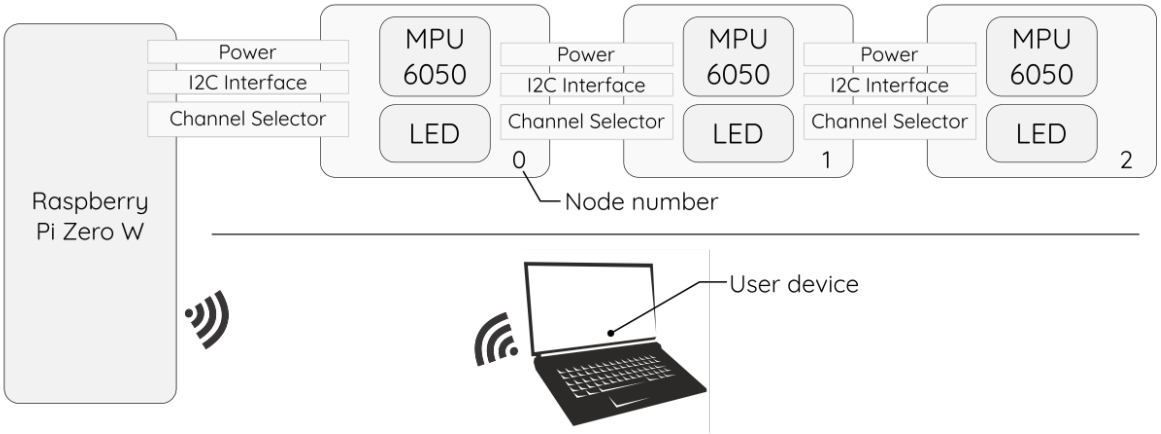


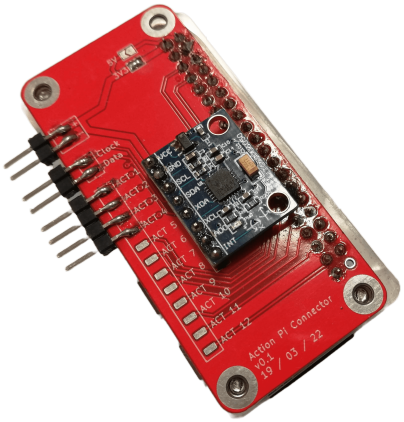
Figure 5.4: A complete overview of the proposed Action Tracer system shown as a block diagram with all the key components.

While a fully wireless system was desired, a first step to such a system will be used where each of the nodes will be connected using cables and daisy chained as is shown in Figure 5.4. Only the data collection to the Raspberry Pi will use cables, meaning that all the wires will be on the wearer's body. The transmission of data from the processing device to the monitoring device however will be wireless and done over a WiFi network.

To begin, schematics were drawn up to show how each of the components should be wired and daisy-chained to allow multiple devices to be used. This schematic is shown in Appendix B.2.1. Based upon this, a draft set of nodes was created and tested. These were then improved by designing and fabricating printed circuit boards.

Having completed the node construction, the next point of focus was the main node and all its interfaces. Due to the standardised Pi header, it was simple to fabricate and design a HAT (Hardware Attached on

Top) that would support this use case, to neaten it up. This is shown in Figure 5.5a alongside Figure 5.5b which shows the final constructed node. The design schematics are provided in Appendix B.2.



(a) A picture of the final Pi HAT node connected to the rest of the system via 4 IMU selector lines, I2C bus and power lines.

(b) A picture of one of the nodes in the system use with power lines, I2C bus and 4 IMU selector lines running through the board.

Figure 5.5: The complete Action Tracer development hardware system displayed.

Figure 5.6: A picture of 3 nodes and the Raspberry Pi being worn by a user to track the upper limb movement.

5.4 Summary

In this chapter, a full overview of the designed Action Tracer system's hardware has been provided. The hardware created and combined will be integrated with the software specified in Chapter 6. This was done by assessing the requirements that the hardware needs to meet, followed by selection of the different components required. Finally, the components were put together, and a final design presented, which will be evaluated in the chapters to follow.

Chapter 6

Software Design

This chapter will provide a deeper analysis of each of the given software subsystem requirements given in Table 3.2 to draw up specifications of what the Action Tracer system should accomplish with regard to its software and firmware. This chapter is very closely linked with the Hardware design given in Chapter 5, as the selected embedded platform limits the possibilities of the firmware. First will be the above-mentioned analysis, followed by a look at the design choices with respect to firmware and software based on the acquired specifications for both the generated API and the data collection. Finally, the proposed design will be presented, showing both what an end user will have access to and what a developer will have access to.

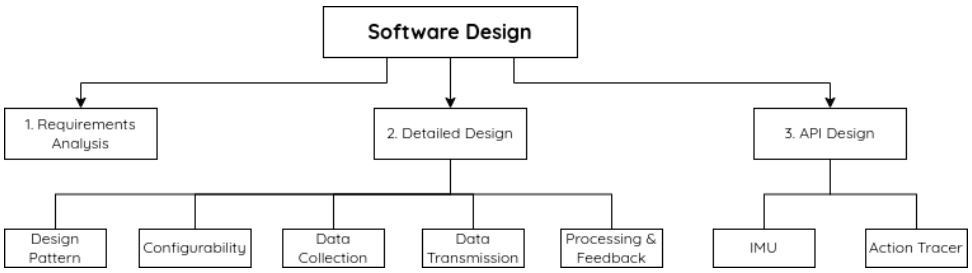


Figure 6.1: An overview of the key topics that will be covered in this design chapter.

6.1 Requirements analysis

To adequately address each design aspect of Action Tracer's software and firmware, each requirement from Table 3.2 will be analysed and have specifications drawn up from it and presented in Table 6.1. In some cases, past literature will be used to inform specifications and design decisions.

6.1.1 SW01: Collected data must be retrievable from any device

Apart from storing the collected data from each node, the embedded platform must be able to send the data to connected devices. When sending this data, however, a receiving device must be able to correctly

decode the data packet that it receives. This calls for a uniform data packet structure being sent. Over and above this, the transmission protocol used between devices must be available to multiple platforms, such as computers and phones. Two common protocols are Bluetooth and Web sockets.

Alongside these protocols, to satisfy the requirement of a uniform data packet structure, a serialisation framework is required. This may be done by way of creating a custom framework, or using existing structures such as JSON or Google Protobufs[93]. Benefits of the latter are their wide use apart from data transmission in areas such as data storage and organisation. Irrespective of both the transmission protocol and the serialisation framework, this data must be retrievable from an Android phone.

6.1.2 SW02: Data transmission should be seamless

A good motion capture system may be defined by the accuracy of tracking, but this feature pales in comparison to the ability to collect and send data. Due to the costly nature of collecting data from multiple IMUs, there is a clear need for an embedded platform that will be able to simultaneously capture and transmit data, which may be done on two separate processing threads.

Apart from sending packets from the system to a user device seamlessly, this process must also be done consistently. At a set interval, the IMU must be able to send a data packet when requested or when it is scheduled to do so.

Finally, to aid in reducing overheads in this transmission mode, the data collected from IMUs must be minimally processed. This should be left to the developer's discretion, as different use cases require different data handling processes. This reduction in extra work will help to increase the rate of packet transmission.

6.1.3 SW03: Developers must be able to build upon this system

This is the most fundamental of the software requirements, as the purpose of this system existing is to allow it to be easily replicated and available to many users. Apart from the hardware, the software used to operate the IMUs, process and transmit data must be easily obtainable and easy to use. As a first step, the firmware must be open source.

With easy to use hardware, there must also be easy to use software. As a result, the system being made must be very simple in its operations and must combine multiple simpler and more common tasks to capture motion. This should be coupled with a developer's ability to build upon the system, allowing them to combine these simple blocks into their own complex program or routine. Such functionality may be compared to various kinds of APIs. In the case of Action Tracer, multiple API levels are required, namely node level and embedded platform level.

A further refinement to this requirement is that this code will only support the components used to make Action Tracer. This is means that code used will be used to support only the selected embedded platform and IMU. No other platform or IMU will be supported.

6.1.4 SW04: The system must allow for configurability

Due to the use of simple small operations as stated in SW03, the system should generally have multiple different configurations available. These configurations should be easily accessible and available to modify. These will aid a developer to select parameters that further suit their end goals with the system.

To further expand on this requirement, a developer must also be able to independently control each node, irrespective of how many nodes are connected to the system. Not only would this allow for a variable number of nodes in the system, but it would also allow for different nodes to have different settings and constraints. This requirement further supports the need for an API that interfaces with the IMU to change its individual parameters as required.

6.1.5 Software Specifications

Given the requirements analysis done in Sections 6.1.1 to 6.1.4, the specifications in Table 6.1 were obtained.

Table 6.1: The software specifications (SR-SW-SP) necessary to develop and build Action Tracer.

Specification	Description
SW01-SP01	Data must be universally serialised in an accessible format
SW01-SP02	The system must allow Android devices to collect data
SW02-SP01	A separate thread to handle transmission will be used
SW02-SP02	Data must be transmitted at a set rate
SW02-SP03	Data must be minimally processed after collection
SW03-SP01	The firmware will be open source
SW03-SP02	The software will be open source
SW03-SP03	An embedded API will be made allowing execution of commands relating to motion capture on the host device.
SW04-SP01	Ensure independent control of each attached node

6.2 Detailed software design

In this section, the fleshed out design specifications will be further investigated, and design decisions made based upon them. As a result of the discussed design, a general programming template will be formulated to allow the development of the firmware as well as any other additional software to be used in this project. Figure 6.2 showcases an overview of the expected firmware functionality in the form of an activity diagram.

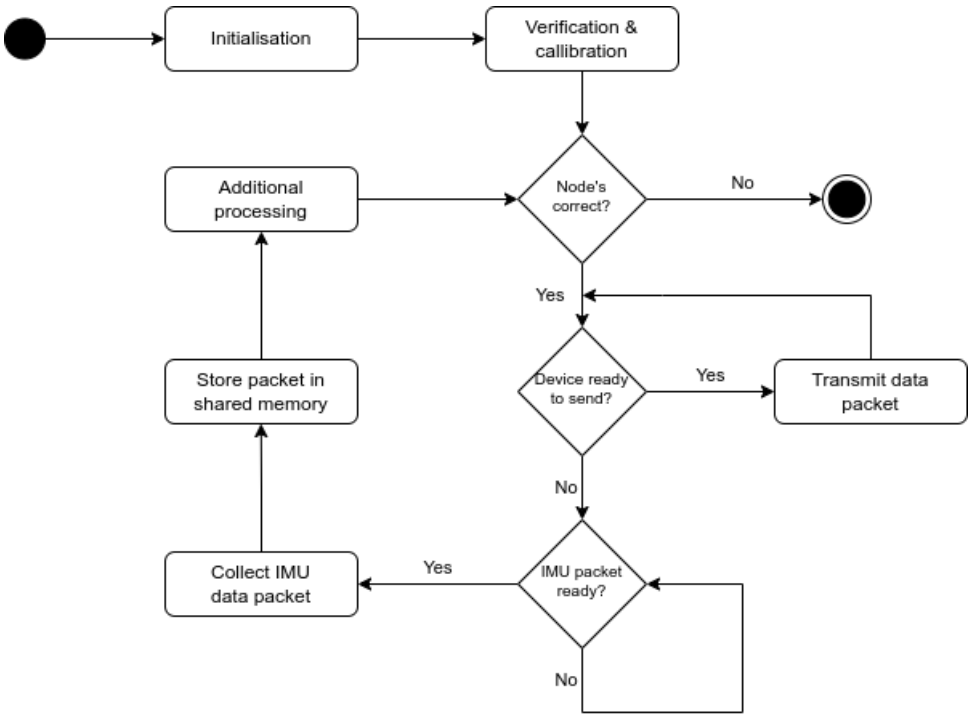


Figure 6.2: A UML activity diagram showing how the software should function in order to successfully be considered as a valid solution.

Based on Figure 6.2, it can be seen that the general program flow that will be followed starts with an initialisation phase. Here the system connections are verified and each IMU node is calibrated. Having ensured this, the system moves into the next state, which involves sending any collected data should it exist. This same state is the first in a recurring loop till the program is stopped. Having sent data, the next stage in the process involves the collection of data from the connected IMUs should each node be ready to present data. Having collected the data, the final stage involves storing the data on the processing node and transmitting it to any clients that may be requesting this data. Having this understanding of the program flow, the following subsystem components can be identified,

- Collection
- Processing
- Transmission

This section will also delve deeper into these three components of the software subsystem, providing details of how they will be implemented in the Action Tracer system.

6.2.1 Design Pattern

Before designing the firmware for Action Tracer, a general programming pattern needs to be specified. This allows the code written to conform to a given standard, making it more intuitive to developers of said standard and allowing unknowing developers to have a practical use case to learn from.

At a high level, this project will follow object-oriented design[89]. The benefits of this paradigm include

the concept of class, a container for an object, data encapsulation, the ability to restrict data and method access and polymorphism, the ability to use multiple data types within a single class. When comparing the goals of the Action Tracer system and this patten, a basic hierarchy may be established, as is shown if Figure 6.3.

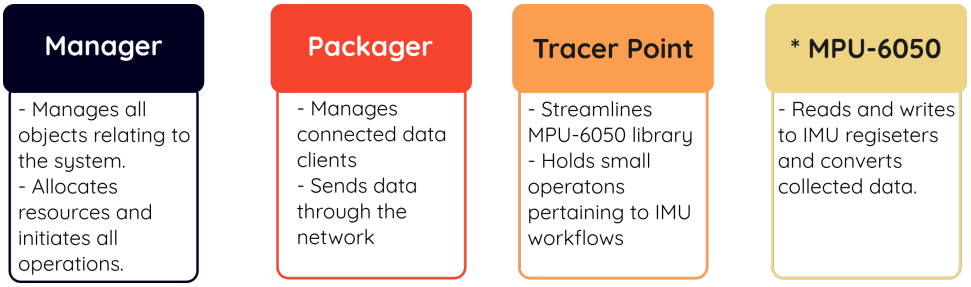


Figure 6.3: A diagram showing the different objects that will be put together in the firmware, allowing Action Tracer to be functional. ** Denotes a class that will not be created by the author.*

6.2.2 Configurability

This section will cover the design ideas that were used in ensuring the creation of a configurable system. This is covered on two different levels. First, a node level configurability will be covered and this will be followed by a system level configurability. Both of these will allow a wider user space to be served by this system.

6.2.2.1 Node Configurability

Based on the proposed hardware node design in Figure 5.4, a single node in this system will have an IMU and an LED. This simpler design allows for each node to have the same control class, with independent control of the IMUs on each node. To help access each node, the I2C Device Library(i2cdevlib) was used[94]. This open source library comprises I2C device communication code for multiple embedded platforms such as Arduino and Raspberry Pi. Alongside this, the library catalogues multiple commonly used I2C capable devices and provide implementations that may be used by developers. Figure 6.4 details some MPU-6050 configuration parameters that can be configured.

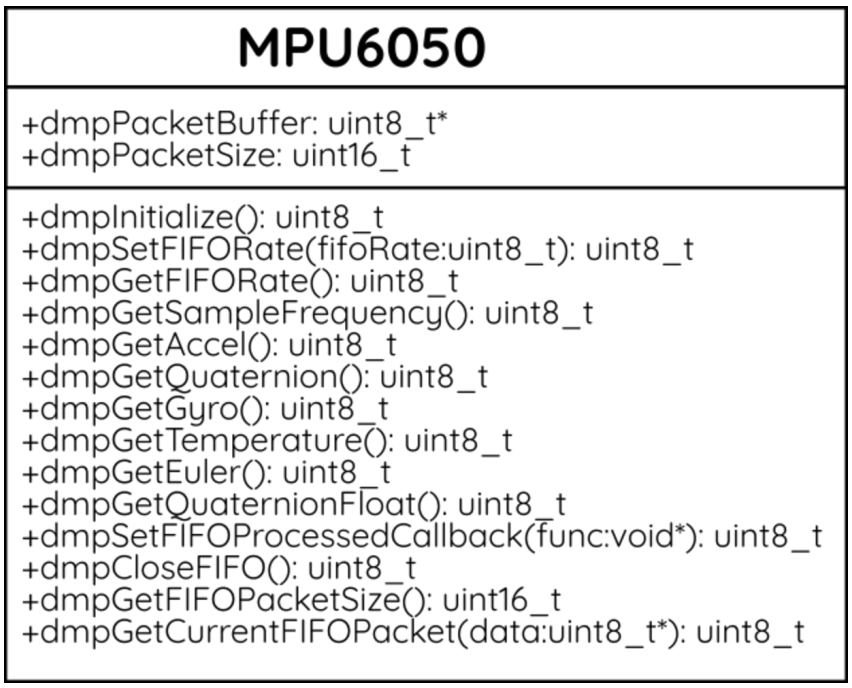


Figure 6.4: UML class diagrams of some methods and variables offered by the i2cdevlib's MPU-6050 library.

This library conforms to the object-oriented design pattern, which leads to each node being instantiated as a unique object. Each node will have unique control therefore of its own settings, and access to it within a program may be easily controlled.

6.2.2.2 System Configurability

Before beginning data collection, there is a need for the system to be verified, ensuring that the system is adequately prepared for the tasks at hand. In the case of the MPU-6050, a unique DMP activation process is required for each device, which will configure the IMU to produce the required data in its FIFO. As a result of this, additional safeguards are required to ensure each node is ready for operation.

After this initialisation, each node needs to be calibrated. In order to do this, calibration, a simple method that may be used may simply be to leave the IMU stationary and have it collect multiple readings and obtain an average zero offset for each IMU. Ideally this runs for as short as possible but, depending on the use case, may be altered. The MPU-6050 is accommodating for this by including registers that control this offset and include it internally, even when utilising the DMP.

Some settings are universal to the whole system and while each node may have unique settings, these settings may apply to some or all of the nodes. Examples of these settings include the data output rate, DMP collection rate and device address (104 or 105) to name a few. These are vital to be set system-wide, as any incoherence may lead to inconsistent system results. Over and above this, some settings may need to be configured before initialisation and calibration to ensure their validity. Such settings may not be changed during run time or may not pair well with the system if configured with different parameters.

6.2.2.3 Design Decision

This section provided a description of the configuration of the Action Tracer software subsystem. Given the requirements, the Action Tracer system will implement a thread which will be used to handle the initialisation, validation and mapping of devices. It will hold most the program logic, and will act as a bridge between the systems configuration and the developer. This is illustrated well in Figure 6.5, which will be built upon further in this chapter.

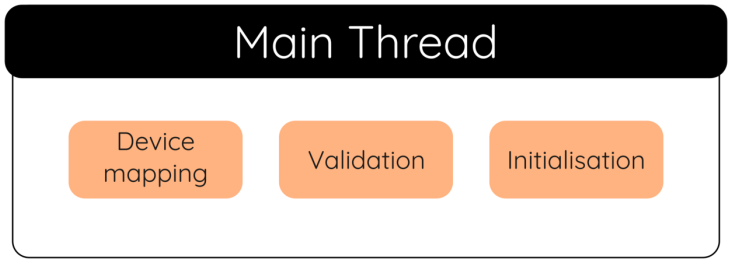


Figure 6.5: An illustration of the running program thread and the tasks it will be associated with when running the system

6.2.3 Data Collection

Having verified the connections to the IMU are valid, that the system is successfully configured for data collection, one should be able to start the data collection cycle. This cycle may be represented for n nodes by the sequence diagram in Figure 6.6. This forms part of the collection loop between the Action Tracer system and the IMU, as was previously shown in the use case diagram presented in Figure 4.2.

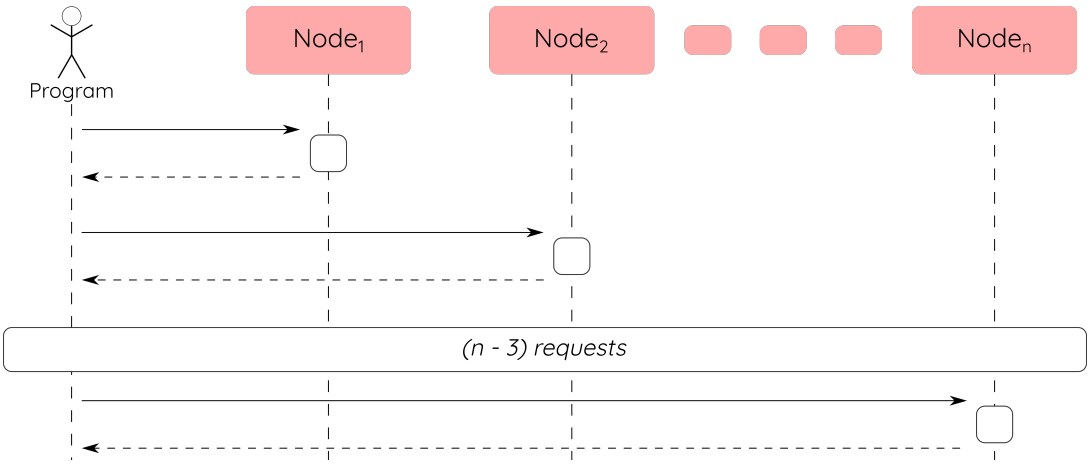


Figure 6.6: A UML sequence diagram showing how the communication with n IMU nodes over I2C.

Figure 6.6 showcases the program's logic when it comes to collecting data from the various IMU nodes. This sequence diagram shows how data will be requested from the program thread to each IMU. This happens in a round robin way because the IMU nodes all share a single communication bus. In this operation, there is no option for parallel data acquisition from all the nodes. The rate that this operation

occurs at drastically affects and limits the system's data output rate, a vital metric that will be assessed in this project.

A deeper look into the communications between IMU and processor shows that the IMU, when using the DMP, will return a 42 byte data stream that contains quaternion(8 bytes), acceleration(6 bytes), gyroscope(6 bytes). In total, only 48% of the data collected from the IMU is usable, but it must be wrapped in this 42 byte container while the remaining data is used as padding. Given the rigid structure of this packet and the IMU being configurable to change the output packet structure, this container will not be changed.

Figure 6.7: The packet structure of the data collected from the MPU-6050's DMP by way of the IMUs FIFO.

6.2.3.1 Design Decision

The MPU-6050 comes with two I2C speed options, Low (100kHz) and High (400kHz). The Raspberry Pi Zero W is more flexible with its offering, allowing a variety of I2C speeds, with 100KHz being the default. The only method for optimisation would therefore be to increase the I2C communication speed to 400kHz, to hasten communications by up to four times. To further keep this process from being any longer, it will run on its own thread, which will continuously collect the data and pass it on to memory shared between threads on the embedded platform. An illustration of this is provided in Figure 6.8.

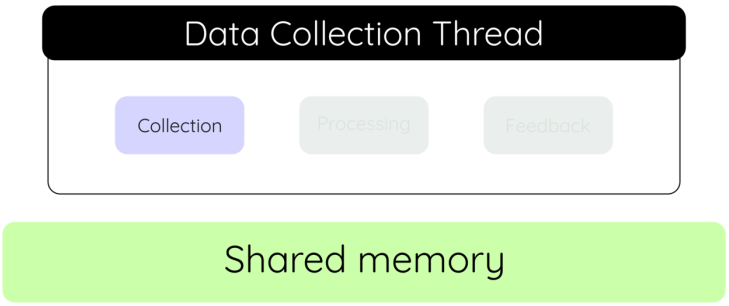


Figure 6.8: An illustration of the Action Tracer system functionality relative to the collection thread.

6.2.4 Data Transmission

This section will provide a closer look at the data transmission component of the software subsystem. First, the transmission protocol that will be used will be given. This will be followed by the desired network hierarchy being presented. When combined, these two attributes will allow for a manageable and scalable network of sensor nodes.

6.2.4.1 Transmission Protocols

One of the main goals of Action Tracer is to have a motion capture system that transmits data wirelessly to a user's device. This may be achieved by setting up a means to transport data to the user's device. In this case, the Wi-Fi interface will be used, and network packets will be sent from device to device. In order to do this, sockets will be required. Sockets allow communication between nodes within a network, utilising IP addresses and IP ports. For this, two transport layer options were explored, Transmission Control Protocol(TCP) and User Datagram Protocol(UDP). The former guarantees in order data transmission to the user, but increases the data packet size due to its larger header, while the latter neither guarantees the client receiving the data packet nor receiving it in the correct order, however it has very low handshaking time losses. Both these methods will be evaluated and the better performing protocol selected.

6.2.4.2 Network Hierarchy

Given the selected hardware structure shown in Figure 5.4, a client-server model may be used to distribute data within a network, where the embedded platform sits as a server, distributing data to user devices connected to it. To achieve this, existing libraries existed that may have been used to handle client interactions with the system, however these would provide multiple overheads which would be detrimental to the system's application. A custom implementation can therefore be used to solve the expected topology. Figure 6.9 provides UML class diagrams for this part of the system and will be used in conjunction with the proposed design.

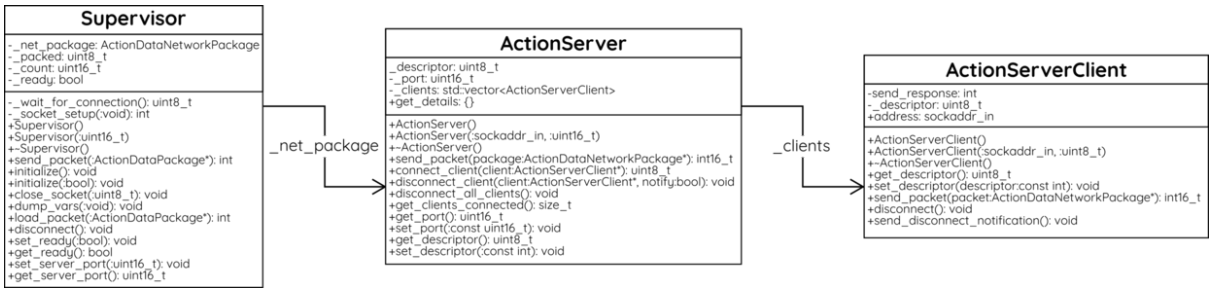


Figure 6.9: UML class diagrams showing the communication component's structure and interactions.

To further expand on Figure 6.9, the 'Supervisor' is in charge of handling all the processed to be done with respect to sending a data packet. This includes collecting it from shared memory to passing it on to an Action 'Server', which will distribute the data to its Action Server 'Clients'.

6.2.4.3 Serialisation

Irrespective of the transportation and transmission methods and protocols, the data need's to be compiled and sent to the user's machine in a digestible manner that developers may be able to utilise. Two methods are available, which are to use or not use serialisation. The communication design allows for both these options to be available to a developer. Not using serialisation would require the network to send an array

of 32-bit floats, which may prove to be difficult to handle as various system architectures handle this data type differently. Using a serialisation framework would mitigate this due to its ability to equate data types internally. This will be done using the Google Protobuf framework. Protobuf, lets the user specify a schema and compiles this schema into a header file for various programming languages such as C++, Python, Java, and many others[93]. For the Action Tracer system, the schema shown in Figure 6.10 will be used to define data being transported to the client devices.

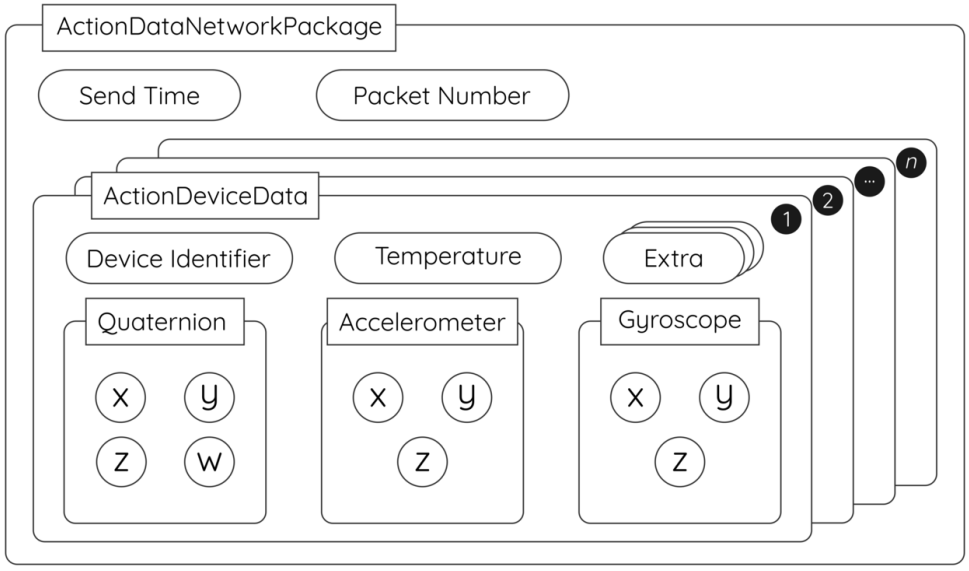


Figure 6.10: The Action Tracer Protobuf specification illustrated to show the different structures that make up the packet.

Each network packet (`ActionDataNetworkPackage`) will be sent with data from an instant in time (`Send Time`) and will have a `Packet Number`. Each network packet will carry the data from multiple nodes (`ActionDeviceData`). Each of these node packets will have a `Device Identifier` as well as the various data elements that have been collected. These are the `Temperature`, `Quaternion` data (*w, x, y, z*), `Accelerometer` data (`x, y, z`), `Gyroscope` data (`x, y, z`), as well as any other data that a user would want to send (`Extra`). The temperature data is being collected as extra data which will not be utilised in this research.

This serialisation helps developers as it allows the correct data structure to be ported to their application with the same object properties as those used when developing and transporting data packets.

6.2.4.4 Design Decision

Given the need for fast client communications and serialisation, all processes regarding transmission and serialisation will be handled by a single thread to not slow down the process of data collection and to allow parallel data collection and transmission, improving the overall data transmission rate. Also needed is a thread to manage client connections, to allow connections even while the system is operational. An illustration of how this system fits into the Action Tracer system is shown in Figure 6.11.

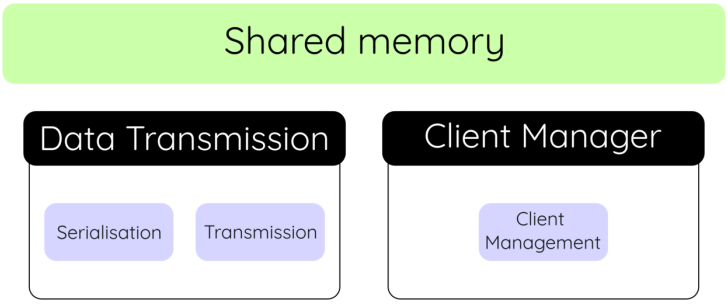


Figure 6.11: An illustration of the Action Tracer system functionality relative to the transmission thread.

6.2.5 Additional Processing and Feedback

In this section, an outline of the design allowing for additional data processing to be done and feedback. The core of this sub-system is to provide additional functionality and insights to the data being received from the IMU.

6.2.5.1 Additional Processing

This will be done by allowing each node to have its own extra set of methods relating to filtering or developer defined operations. These may include running much less intensive filters such as median or mode filters or allowing further calculations above collecting the DMP packet shown in Figure 6.7.

6.2.5.2 Feedback

In order to make a user aware of any adjustments to be made, a device is required to ensure that a user may notice the necessary adjustments. This will be achieved by using an LED connected to each node that may be independently controlled. Each node class will be equipped with data variables which will determine the alert status. These alerts will be issued either by the system or by the developer when being set up.

6.2.5.3 Summary

Due to the nature of this component of the software subsystem, both the feedback and processing elements will be deployed on each node and be based on the collection thread. An illustration of how these elements will fit into the Action Tracer system is shown in Figure 6.12.

Figure 6.12: An illustration of the Action Tracer system functionality relative to the feedback and processing that may be additionally used.

6.2.6 Detailed Software Design Summary

Given the above design considerations and selections, Action Tracer will be built with Object-Oriented Design patterns and based on multiple processing threads. These threads will be responsible for system start-up, assessment and processing, data collection and data transmission. The goal of this multi-threaded system is to allow efficient data flow, which will later be assessed. Figure 6.13 shows the system in its entirety, based on each thread's tasks as well as the memory distribution.

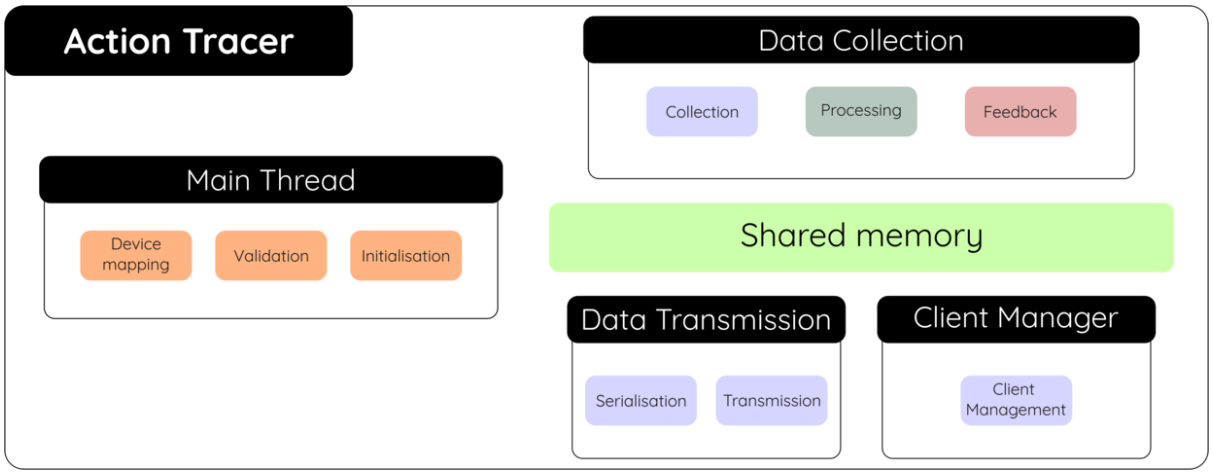


Figure 6.13: An illustration of the Action Tracer system as a whole, showcasing all the threads and memory space.

6.3 API design

A high ranking goal of the Action Tracer system is to allow the use of an API by way of an Android device. While Section 6.2 briefly touched on the operations of the device and Figure 6.3 provided an overview of the classes produced, there remains an exploration of the functionality of the API. This section will provide the design of two APIs, the first which is specific to the Raspberry Pi, useful for interacting with the MPU-6050, and the second API, useful in motion capture, which allows the system to run safely. Both of these APIs have been implemented in the C++ programming language.

This section will only provide a shallow look at the APIs, and only the main and important methods will be discussed in brief. For more information on the methods available and the function prototypes, a link to the documentation is found in Appendix B.5.

For each of the APIs, a function tree diagram will be used to present the main categories of data as well as the functions that will fall under those branches. This will help to understand when each of the functions should be called as well a giving a simple understanding of how they work.

6.3.1 Tracer Point API

This section will outline the main functionality of an API directly linked to the IMU in use in this project. The capabilities of this class are enclosed in the Tracer Point class, denoting an IMU node. This section is directly dependent on Rowberg's library and extends a few of its capabilities[94]. This is a C/C++ compatible library which can be used to interface with various I2C devices on various platforms. This library is known as i2cdevlib. The goal of this API is to execute specific methods relating to set up, data acquisition and minor processing.

6.3.1.1 Aims

The main goals of this class are to manage and execute the following events:

1. manage device communication on the I2C bus, ensuring there is no interference
2. calibration of an MPU-6050
3. initialise the DMP and ensure it is ready for use
4. initialise and set node and system level configuration settings
5. label and identify nodes
6. collect MPU-6050 data
7. perform minor data processes on the IMU data

The above aims are vital processes that need to be managed due to the complexity of the chosen hardware. For instance, (1) is a vital role of this API, as the implementation of the I2C communications had to be altered. Due to the MPU-6050 having two different addresses, two devices may be placed together and used with no additional hardware changes, however the introduction of a third device would cause interference as the device's I2C address can not be altered. as one of the two devices will be active on the same address. This may resolve by using an I2C multiplexor, however more wiring will be required. The selected solution to this issue was to toggle the address of the IMU to 105 when in use and 104 when idle.

Other vital aims are (2) – (4), which are necessary before the main data collection and transmission begins. Granted each user wearing the device has different start positions and body geometry, each time a node is worn, there may be slight placement differences that need to be accounted for. (5) is closely related to the above-mentioned points, as it allows each node to have its own unique qualities and parameters.

Finally, examining (6) and (7) will allow for a developer to add additional functionality to their version of the system, which would add additional processing to data being collected, in real time. The disadvantage to this process is that the complexity of the process will directly impact the data output rate.

6.3.1.2 Commands

Figure 6.14 outlines all the commands that the Tracer Point class will have available to a developer utilising this class. This class is implemented in the Action Tracer Server class, as the latter class executes simple commands to achieve a wider variety of usability.

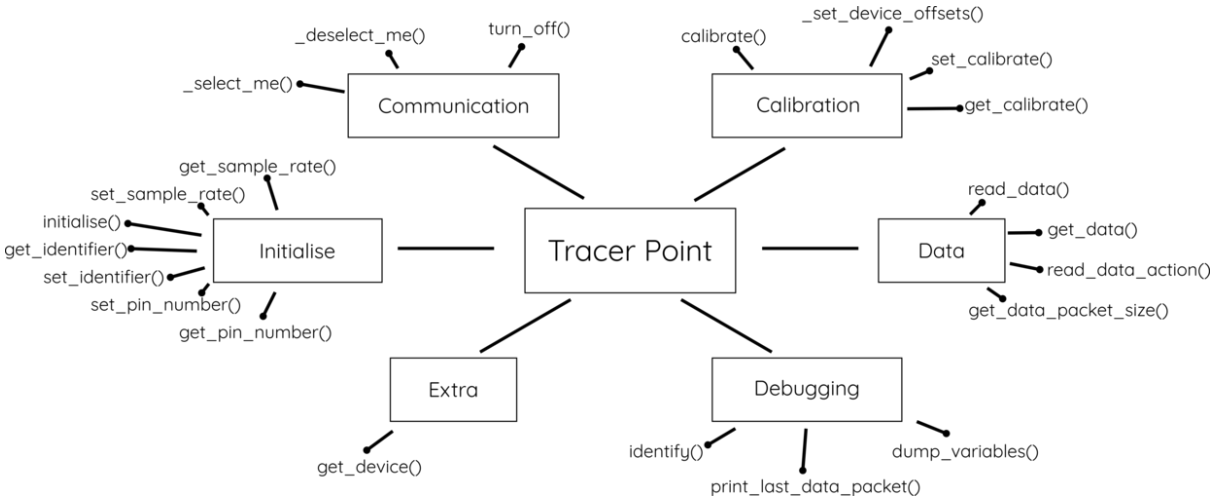


Figure 6.14: A tree diagram of the commands available in the Tracer Point class, grouped by functionality.

All the commands rely on much smaller operations, either bespoke to their function or drawn out from the Rowberg's MPU-6050 library. A brief description of the branches is presented below.

Communication
Under the communication branch, multiple MPU6050 IMUs may be controlled through a single I2C bus. The only available method to a developer would be the `turn_off` command, which deselects a device. The `_select_me` and `_deselect_me` commands allow individual device selection and independent control without interference.

Calibration
Methods under the calibration branch are used to ensure that the IMU will be able to reset its orientation. The `set_calibrate` and `get_calibrate` methods are used to change and read whether a device will run the `calibrate` method, which will run the calibration process.

Data
Methods under the data branch are used to collect data from the IMU. The `read_data` method will collect the data and store it in a buffer on the device. `get_data` will return the data itself to the program calling the API. `read_data_action` will return the data prepackaged for the Action Tracer Server API.

Debugging
The commands under the debugging branch are used to help the developer fix issues that may arise as they create a program. `identify` is used to switch on the feedback device on a node, while the `dump_variables` method will show every internal setting that is being used on the node. Finally, `print_last_data_packet` will present the last data packet that was read from the IMU.

Initialise
When setting up a node, commands under this branch are used before calling the `initialise` method. These are useful in setting the point's identifier, as well as which Raspberry Pi pin will be used (based on Wiring Pi). Options also include `get/set_sample_rate`, which sets how often data is sent out by the IMU.

6.3.1.3 Additional Notes

This API will be built with and for the C++ programming language. Data collected from this API will be supplied only to the program calling the API.

6.3.2 Action Tracer Server API

This section will outline the functionality of the API with relation to the default system's control. These discussed traits are available in the Action Tracer Server class, which controls all the nodes in the system, as well as the communications between clients and the Action Tracer server. The main goal of this API is to control and run the entire system.

6.3.2.1 Aims

The main goals of this class are to do the following:

1. manage and control the clients connected to the system
2. manage and control the server
3. ensure data collection is possible through parameter validation
4. control data collection, including its rate
5. transmission of data
6. distribute the workload of motion capture

This API allows a more friendly interface to start motion capture and gather all the data required from the different nodes (run by the Tracer Point API). Inherently, this would include communications management, (1), (2) and (5) as separating this functionality would introduce further format translation issues. While all the network processes are relatively low level, the API utilises the Communicator class to make this work more intuitive.

Data management is well addressed by aims (3) and (4). These capabilities allow a robust and reliable system, which would lend itself to various other IMUs, making it extensible. By changing the Tracer Point class, further abilities exist to a developer within the space of data collection and processing.

This class will achieve aim (7) by creating and scheduling when threads operate and what work will be running on them. As discussed in Sections 6.2.3 and 6.2.4, to speed up the data collection, transmission and ensure adequate processing times, data needs to be moved through the system rapidly and efficiently.

6.3.2.2 Commands

Figure 6.14 outlines all the commands that the Action Tracer Server class will have available to a developer utilising this class. These commands may be executed in a bespoke program, such as one listening to API requests.

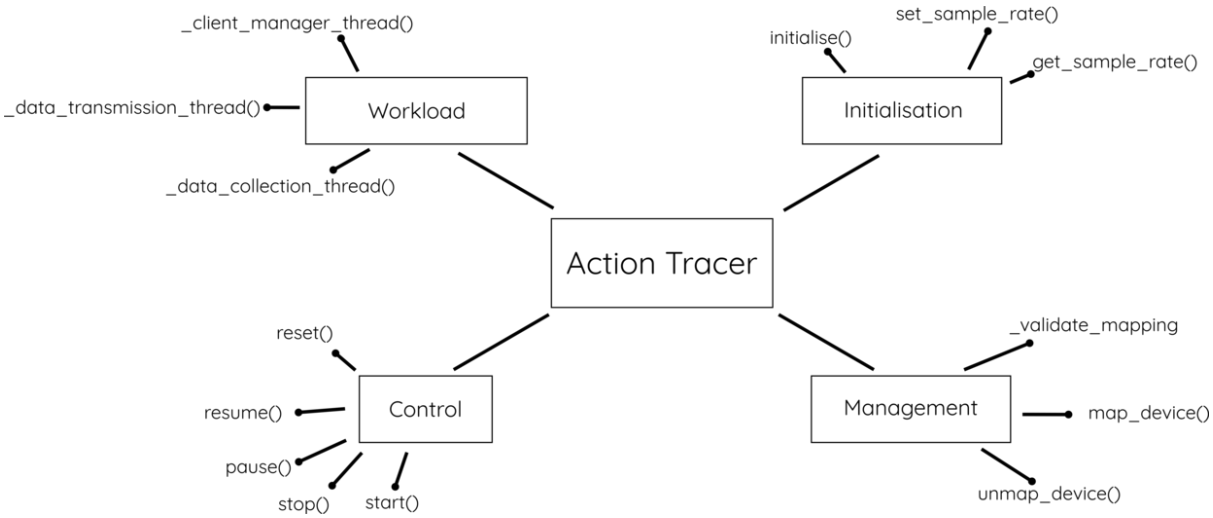


Figure 6.15: A tree diagram of the commands available in the Action Tracer Server class, grouped by functionality.

Below, a brief explanation of the main commands shown in Figure 6.14 is given.

Control
Commands under the control branch are used to manage the data collection, processing and transmission. In order for these methods to be used, the system needs to have run the initialisation commands first. When the `start` and `stop` commands are called, the device will start and stop collecting data from the IMUs and transmitting it to all connected clients, respectively. The `pause` and `resume` methods control the transmission of the data to clients, but will not stop the collection of data from the IMU.

The transmission of data in this API assumes the receiver will have sufficient time to process the data being sent, and will send the data at a constant rate.

Initialisation
The commands listed under this branch are used to prepare the system for data collection, transmission and processing. Commands here are used to set the connected IMU node's data collection rates before transmitting preparing the devices via the Tracer Point API, presented in Section 6.3.1. The `initialise` method will only run if IMUs have been mapped to the system via the management commands.

Management
Commands listed as management commands are used to make the Action Tracer system aware of the IMU devices it should be aware of and control. Used in conjunction with Figure 6.16, the `map_device` and `unmap_device` commands work to make the system aware of the devices it should be managing.

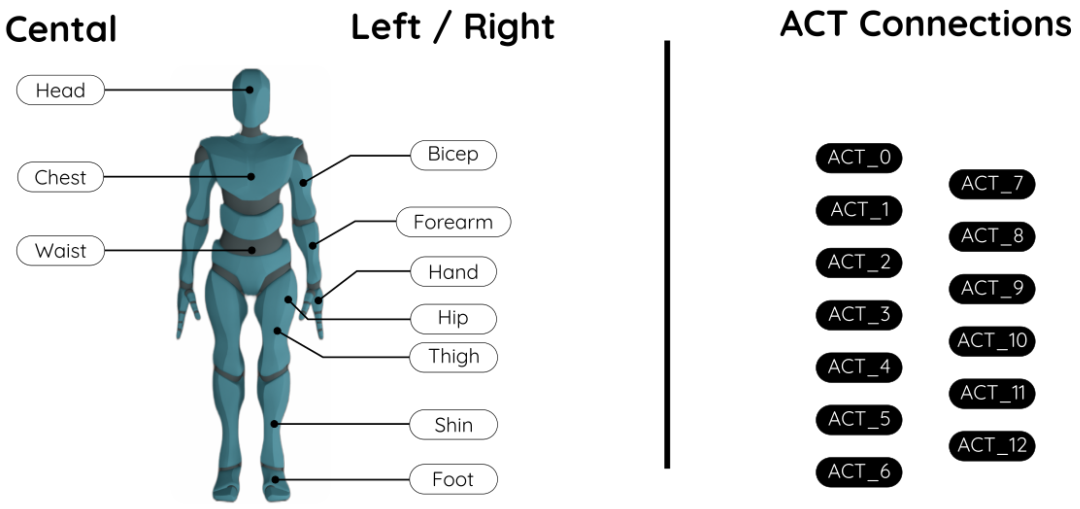


Figure 6.16: A diagram showing the commands used in addressing different parts of the body and Raspberry Pi pins based on the main node of the Action Tracer system.

The `_validate_mapping` command is not accessible to the user but is shown to make one aware of the fact that every mapping must be accurate and relationships between pins and devices are one-to-one.

Workload
Methods listed under this branch are not accessible to the developer but are presented to indicate the use of multithreading. These threads are only run when the user uses the appropriate control method.

6.3.2.3 Additional Notes

The Action Tracer Server API is to be built and used in C++. The program being run on the Raspberry Pi will need to import the relevant libraries and write firmware for the desired application. The data being received from the system will however be available in multiple languages, as is supported by the Google Protobuf format.

No additional timing is provided apart from each transmitted packet having an associated timestamp, as is shown in Figure 6.10.

6.4 Summary

In this chapter, a full overview of the designed Action Tracer system has been provided. This was done by first looking deeper into the various requirements set out for this subsystem and producing design specifications. This was followed by a detailed design of the various components that make up this subsystem, with emphasis on the configurable nature of the subsystem, data collection, transmission and processing. In exploring each of these components, the functions of each of the threads was outlined and combined to present a high level design outline of the subsystem. Following this, the API to be developed for this subsystem was presented and the various functions available to each were defined.

The software created will be integrated with the hardware specified in Chapter 5. An overview of all

the classes created is presented as UML class diagram alongside the files that are related to these classes may be found in Appendix B.5.

Chapter 7

Experimentation

This chapter will detail the experiments used to validate the Action Tracer system on various levels. This will be accomplished by presenting these experiments in three sections.

First to be covered will be the validation of the components being selected to make a motion capture system. This will comprise hardware validation, followed by software validation from Chapters 5 and 6 respectively. This will be followed experiments to test the integration of the whole system put together to do some motion capture alongside the method that will be used to obtain elbow and wrist rotation angles. Finally, benchmarking tests of the capabilities of the Action Tracer system's motion capture will be outlined. The benchmarking will be done by comparing the Action Tracer system to a separate motion capture system. In addition to the tests being conducted, the various different experimental set-ups will be presented in detail alongside any assessment tools that will be used before the various experiments.

Figure 7.1 gives a logical breakdown of this chapter based on the above-mentioned layout, including the various section numbers of where the specific experiments and tests will be found.

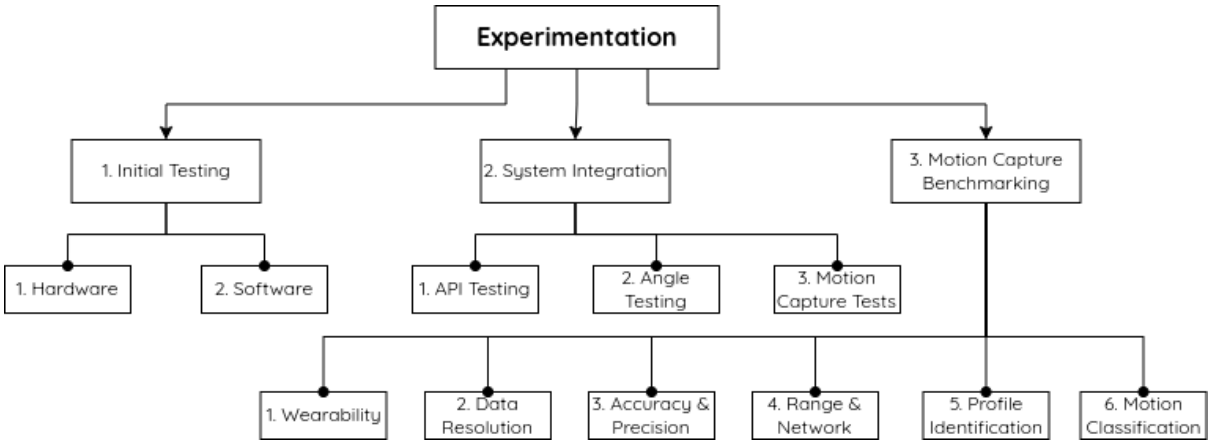


Figure 7.1: An overview of the key topics that will be covered in this experimentation chapter alongside an index to the sections where each of the tests will be found.

The system will be validated according to the question stated in Chapter 1.4. The end goal of all the

experiments is to address these objectives clearly and provide empirical results that provide a conclusive answer whether the system can achieve them, as well as to what extent they do so. Primarily, the system, should be able to capture motion, process the collected data and send it reliably to a receiver. These simple steps will allow the evaluation of various other metrics through data analysis. These include the ability to predict injuries, learn an activity, or to interface with more commercial devices. Based on the results from Sections 7.2 and 7.3, the research questions will be answered.

7.1 Initial Testing

This section will present the initial testing that was required to approve each subsystem to be added to the Action Tracer system. These tests were conducted during the selection of the components to find out if they would be viable solutions for their intended use case. These tests will be presented in two subsections based on the respective subsystems to be tested. The hardware was first evaluated as it determined what experiments would be run for the software subsystem. Not all the aspects of the hardware and software will be assessed, only the major functions relative to this system will be examined.

7.1.1 Hardware

In order to test the hardware, various tests needed to be done involving the IMU and embedded platform, ranging from their specifications to their interface options for the sake of the Action Tracer system. In order to do this, an experimental setup was required which was fair and versatile. Figure 7.2 showcases the hardware validation setup utilised for these experiments with relation to the IMU. This setup is inspired by work done by Guimaraes, who developed a setup to evaluate an MPU-6050 on a Raspberry Pi board[95].

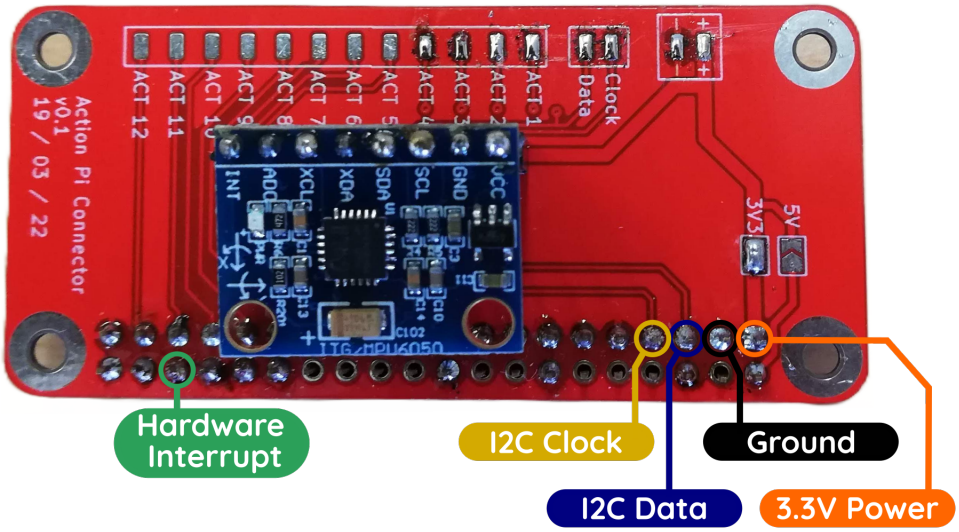
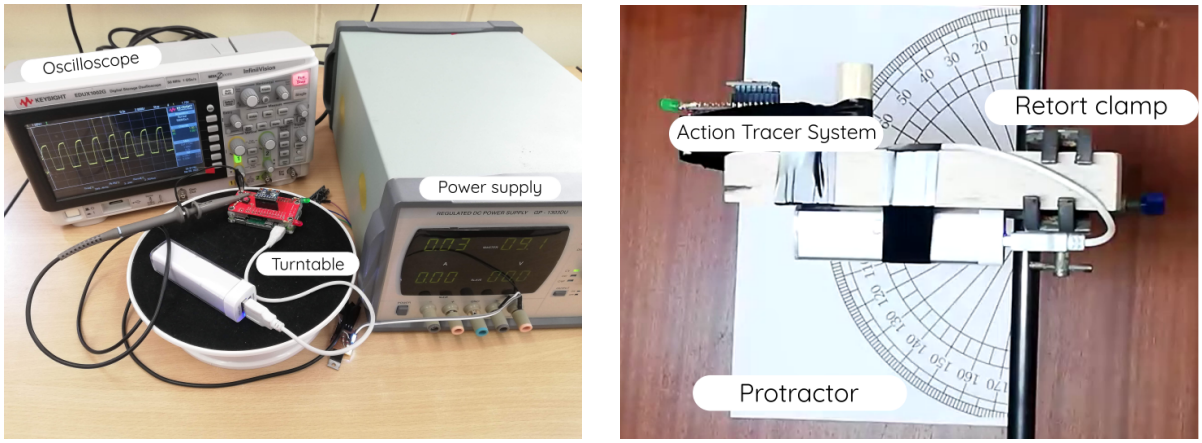


Figure 7.2: The IMU validation experimental set up, which is the main HAT for the Action Tracer system.

This setup is ideal as it allows both the assessment of the Pi's usability and that of the IMU. In Guimaraes

set up, in the collection stage, the user will be able to obtain raw acceleration, raw angular rotation, quaternions, and attitude readings[95].

For tests at this stage, Guimaraes firmware will be slightly altered, obtaining only the acceleration, angular rotation rates and attitude alongside timestamps for each event[95]. The MPU-6050's interrupt will be polled as well to ensure readings are taken when they are ready from the DMP. Based upon these readings, will be various assessments of the hardware and its capabilities. Most of these tests will be done on a turntable with a variable speed controller, while some will be done on an adjustable platform as shown in Figure 7.3. Before each test, the IMU will be calibrated. The programs used to run all tests will be found in Appendix B.5.



(a) (b)

Figure 7.3: Pictures showing the two experimental set-ups used for assessing the hardware. 7.3a showcases the exact setup used with regard to the turntable, while 7.3b showcases the use of the adjustable platform in these experiments.

It must be noted that the sample rate of the IMU is adjustable when the DMP is being used. This sample rate is defined as follows:

$$f_{DMP} = \frac{200}{1+n}$$

where n is an 8-bit integer.

7.1.1.1 DMP calculations

Aim

Using the set-up in Figure 7.3a, the accuracy of the DMP calculations will be assessed by comparing them to those calculated using a modern complimentary filter.

Method

Equations 2.2 - 2.4 will be used to calculate attitude data based on the gyroscope and accelerometer data at a fixed rate. The DMP collected quaternion data will be converted to attitude data based on Equations 2.5 - 2.7. In order to collect data for this test, the DMP will be set to 50Hz, with the hardware interrupt and run for 1 minute. The complete dataset collected in this minute will be used for comparison. This

test is a measure of how accurate the DMP is relative to conventional methods.

7.1.1.2 IMU Drift

Aim
Using the set-up in Figure 7.3a, the drift that is inherent in the gyroscope will be evaluated. As drift is one of the key factors affecting gyroscopes, the level to which these effects affect the MPU-6050 must be evaluated.

Method
To set up for this experiment, the IMU will be placed on the stationary turntable and left to run for 5 minutes, which will give a general idea of the level of drift inherent to the IMU. This will be followed by intermittent movements over the next 5 minutes, to observe how this drift is affected, if at all. This will be conducted at a constant DMP rate of 50Hz, with the hardware interrupt being used.

7.1.1.3 Feedback alerts

Aim
Using the set-up in Figure 7.3b, the ability of the test system to produce alerts given a specific condition will be tested.

Method
This will be done by tilting the platform from being level with the ground to specific positions. Alerts will be set and tested individually for specific angular rates, accelerations, and angles. All these tests will be run at constant f_{DMP} rates. First, the motion will be slow enough to not trigger the alerts, this will be followed by tests to see if the alerts will be triggered at varying rates of angular change.

7.1.2 Software

With regard to testing the software to be used, most of the covered tests will not focus on the data being supplied to the system, rather they will focus on other metrics such as execution speed and capability. These tests will be relevant to deciding how the final firmware to be used for motion capture testing will work, as well as how it should link to the API.

Due to the collected data's non-importance, an IMU and turntable will be used in assessing the software. The set-up for this operation will be done as shown in Figure 7.4, and will run fully on the Raspberry Pi and, where needed, run on a desktop machine (primary evaluation machine), with the specifications given in Appendix B.4. Secondary evaluation devices will also be used, such a cellphone and a laptop, as shown in Figure 7.4, who's specifications may be found in Appendix B.4.

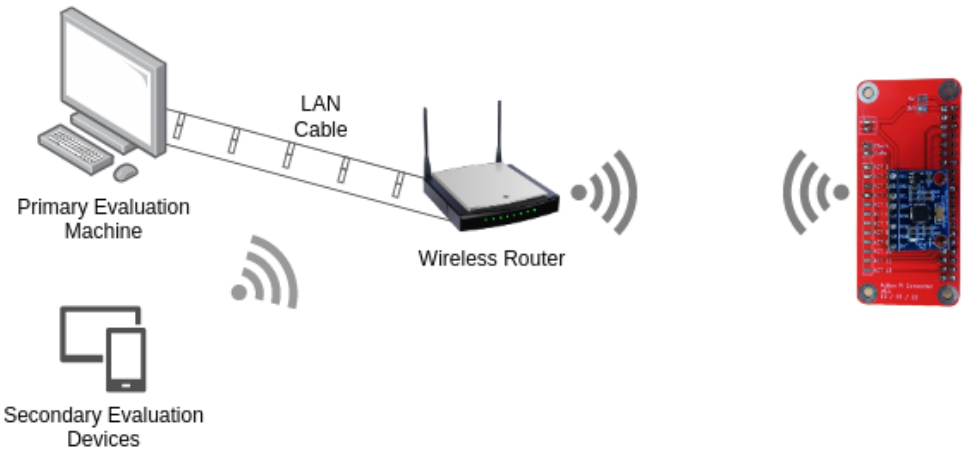


Figure 7.4: The experimental set-up for the software used to run the Action Tracer system.

A 300Mbps Wireless N USB ADSL2+ Modem Router manufactured by TP-Link and will act as the main router through which communication with the Pi will be done at a fixed distance away from each other. The experiments being done will build up on each other, which means that learnings from prior tests will be carried forward to proceeding experiments.

7.1.2.1 Packet Transmission

Aim

The transmission of a data packet over the wireless interface of the system in real time is of high importance in the Action Tracer system. This experiment will attain this by collecting packets and sending them through the interface, from the embedded platform. Above this, the transmission of data to multiple clients will also be tested.

Method

In order to adequately assess this capability, f_{DMP} will be set to 10Hz with the hardware interrupt active. These packets will be captured by the embedded platform and regulated as they are sent to the recipient machine on different ports and noted. Data packets will be sent over the network directly as 32-bit unsigned integers. Data packets will have the packet structure shown in Figure 7.5.

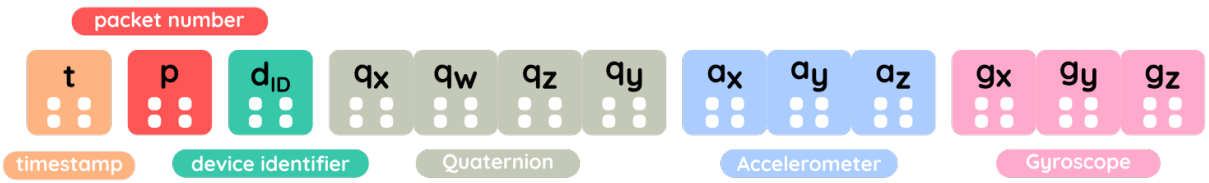


Figure 7.5: The basic data packet structure chosen to transmit data from the Action Tracer system.

Finally, the capabilities of sending to multiple clients will be tested but sending data first to a primary machine, then to various other secondary devices from the Pi.

7.1.2.2 Data Processing

Aim
With a working collection and transmission system, it is vital to evaluate the type of data being passed through the system. This experiment will investigate the effects of various types of processing. These include the execution times for obtaining different types of data based off of the output from the DMP. This will also include testing the effects of various add on filters to the system.

Method
Two types of computation will be executed on the data, conversion and filtering. Both will be done with a constant f_{DMP} of 50Hz, and hardware interrupts will be inactive. A timing block will be placed after data collection and before transmission, where some minor data processing will be done.

Under the umbrella of data conversions, the first to be evaluated will be a conversion from the raw IMU data to attitude readings. This will be measured against conversions from DMP data to attitude readings. These conversions will be placed in the aforementioned timing block. This test is very similar to that which is conducted under the hardware based systems, however the correctness of the data is not under investigation, it would only be the execution time to obtain this data.

Following the conversions, the data being output by the DMP will be passed through a further filter within the previously mentioned timing block. The filters to be tested are the mean, median and mode, which will help in reducing the effects of noise on the data. These will be done with different filter data sizes.

7.1.2.3 Serialisation

Aim
Having obtained a working transmission and processing system, the serialisation of the data is to be assessed. Initially, data was directly being transported from node to client, however this experiment will delve into minor conversions and transmission of logical and organised data using Google Protocol Buffers. Further tests as to the versatility of this system will be presented in 7.2.3, where the advantage of such serialisation will be further noted.

An additional test that arises as a result of this feature, is the ability to test the customisability of the system. While there are no definite metrics to track this process of doing, this will be presented as a result.

Method
To assess the effectiveness of serialisation, two key aspects will be observed. First, the loss in time that arises from needing to serialise the data and secondly, the gain in data package size relative to sending the raw data. For both these tests, f_{DMP} will be set to 50Hz and the hardware interrupt will be inactive.

Much like the Data Processing tests, this will be examined in a timing block over the transmission stage. The first point will be just before the data packet gets loaded with data and after the data is sent. This will be compared to non serialised data being examined at the same section for a fixed number of packets.

The second goal will be attained by comparing the size of the data packet being sent to that being sent after being serialised. These will be compared to a fixed size packet as specified in Figure 7.5.

7.1.2.4 Parallelism

Aim
Given that data collection and transmission are independent data tasks that rely on separate communication protocols, the effectiveness of separating these two tasks is yet to be seen. This will be done by implementing each task on different threads of execution. This test will cover first the possibility of such a task, followed by observing if there is a speed-up in the rate of packet transmission.

Method
To accomplish this task, f_{DMP} will be set to its maximum limit, 200Hz, and the measured metric will be the output data rate after transmission on the client's side. Should there be a packet not collected on time, repeat packets will be transmitted until a new data packet is collected.

Data will be passed from thread to thread by means of shared memory, which will be guarded by C++ native thread locking tools. This will be compared to a procedural program with the same parameters. The time taken to send 500 packets will be recorded and compared between a procedural program, working on a single thread and a multithreaded program, with a thread dedicated to sending data and another dedicated to collecting data. It must be noted that the data being transmitted will be serialised.

7.1.2.5 Idle times

Aim
A potential advantage that may arise from converting the procedural program into one that runs on parallel threads is having idle time not being utilised by the processor. The amount of time available to each thread will be evaluated, as this may play a role in the processing time available in this system, as well as time to perform other tasks.

Method
In order to evaluate the amount of idle time threads in this system have, the total time it takes for each thread to complete one cycle of its tasks will be obtained. The amount of time between starting a new cycle and ending the previous cycle will be indicative of how much time exists between cycles, while also obtaining the amount of time it takes to execute a single simple cycle. This is done using the same `ctime` library as was used in Section 7.1.2.2.

The above test will be done first in the collection thread, to evaluate how long it takes to collect data and if there is any time available in that thread. This will then be replicated in the data transmission thread, where the amount of time between transmissions will be examined.

7.2 System Integration Testing

While many experiments regarding the sub-systems have been conducted, they were done with the goal of ensuring the system works as a data collection system as well as a motion capture system. Testing in this section will focus on ensuring first that the APIs are functional, followed by testing if the device is capable of transmitting data usable in motion capture software. This section will also present the method by which angles around joints will be obtained. The success of these tests are a prerequisite to

the subsequent participant tests in Section 7.3.

Experimental Set Up

In order to test the Action Tracer system's integration success, a controlled environment needed to be created which allowed for the main features of the system to be tested. This environment would need to be able to connect to the system as well as send commands back to Action Tracer. This required that two applications be developed alongside the development of Action Tracer. The first was used to assess the API as well as show the data being transmitted from the system, while the second application was used to visually show these rotations on a human model as well as to calculate the joint angles. These are both discussed below.

API Testing Application

The first application that will be created will simply communicate between the Action Tracer system and an Android phone. Built using the Flutter framework, it will send commands to the embedded platform, which will return data to the system. This data will be stored and displayed on the Android phone. The interface for this application is shown in Figure 7.6. This will be programmed in Dart, which is the programming language used in the Flutter framework. The code as well as application file to this program will be found in Appendix B.

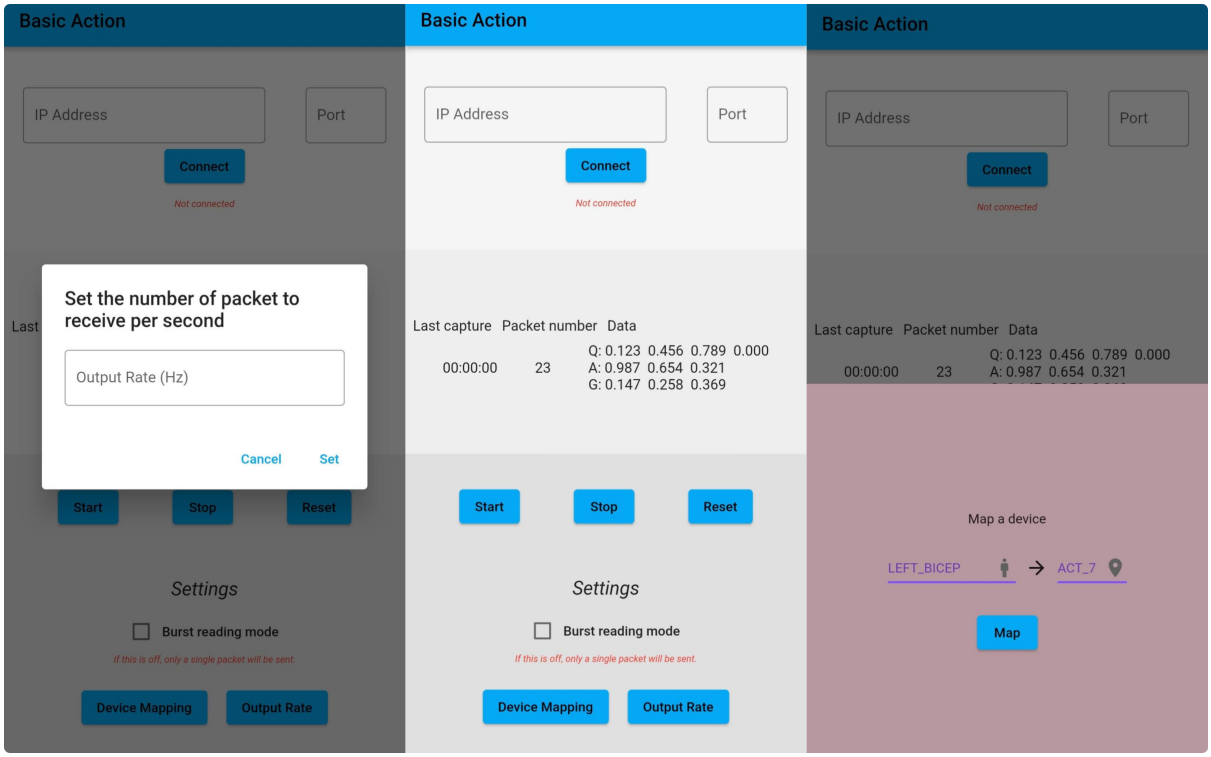


Figure 7.6: Three screenshots from an application built using the Flutter framework used to test the functionality of the Action Tracer Server's API.

Motion Mirror

The Motion Mirror application will be built as a computer and mobile phone interactive application, allowing for data to be captured and mirrored onto a 3D model. This motion mirror will be built in the Unity 3D game engine. This application will receive data and relay this to a 3D model, which will move in the same way as the user, as shown in Figure 7.7. This will be tested on both a computer and on a mobile phone. The code as well as application file to this program will be found in Appendix B.

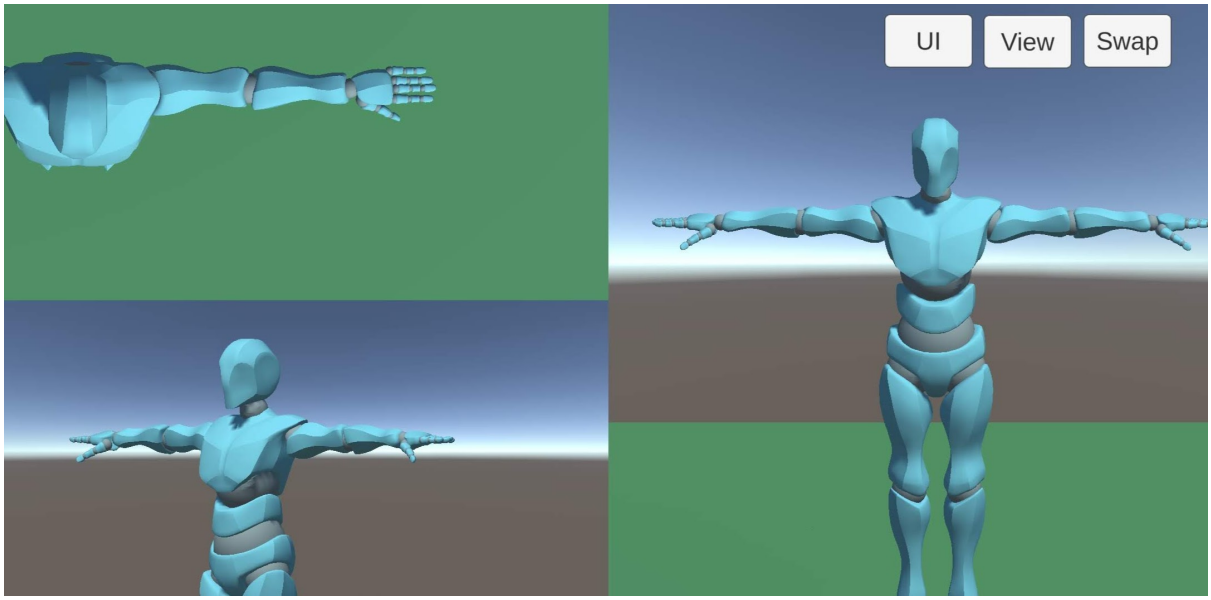


Figure 7.7: A screenshot of the main screen showing a model that will mirror a user's motion.

This application will be used to do more than prove the ubiquitousness of the application of the Action Tracer system. It will be used to obtain rotation data based on the location of various joints that are moving. Given that the model is constrained within a 3D space, each point has 3 data dimensions, x, y, z allowing the use of trigonometry to obtain angles at the given joints. This can be demonstrated by looking at Figure 7.8 which has three points highlighted.

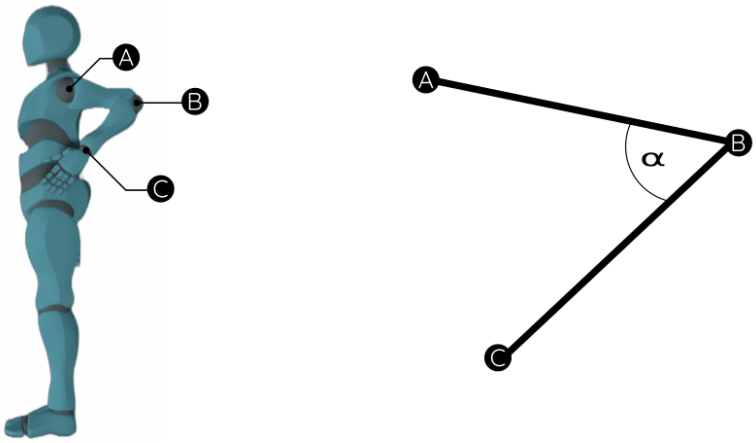


Figure 7.8: An illustration of how angles will be obtained from the Motion Mirror application. The angle α is obtainable using both 2D and 3D coordinate data.

In Appendix B.5.1, two code samples are provided which may be used to find the angle between three points in 2D and 3D space are provided. With this in mind, the Motion Mirror application will be used not only to present the motion, but also to obtain the position data from the selected joints. These position data points will be used in the analysis to obtain the angle at joints for the analysis in this project.

7.2.1 API Testing

Aim
Tests in this section will be based around ensuring the Action Tracer and Tracer Point APIs are functional. Given that the Action Tracer API uses the Tracer Point API, only the Action Tracer API will be assessed, as its success infers the success of the Tracer Point API.

Method
In order to assess the operability of the API, the basic API testing application will be used. This will be done by ticking off the different functions that are available via the API. This will be done in the various groups of usable API methods based off of the Figure 6.15 which are Initialisation, Management, and Control.

7.2.2 Angle Testing

Aim
The objective of this test is to track how well Action Tracer is able to follow changes in angle at a single joint. This will show the margins of error for any given angle reading.

Method
By placing a node on a servo, while another is stationary, the servo will be rotated about multiple angles between 0° to 180°. This will first be done by increasing the rotations from 0° to 180° with brief stops at 45°, 90° and 135°. This will be followed by starting at 150° and dropping to 0° with brief stops at 100° and 50°.

The aim of these tests is to ensure that Action Tracer can track simple angles between two sensors before tracking multiple angles with more sensors.

7.2.3 Motion Capture Tests

Aim
The goal with these tests is to ensure that every system is fully functional and available for use for the motion capture benchmarking tests. This test will also ensure that the data being collected by the system is representative of the motion being conducted.

Method
To conduct the these experiments, the motion capture suit will be worn by a user on their left arm, strapped around their left upper arm, forearm and on the back of their hand. The main node will be strapped around their waist, as well as a power bank to supply the system with power, as is demonstrated

in Figure 5.6. They will stand in a designated area which is a fixed distance from a camera, which will have the user's arm in full view recording the session. This is shown in Figures 7.9.

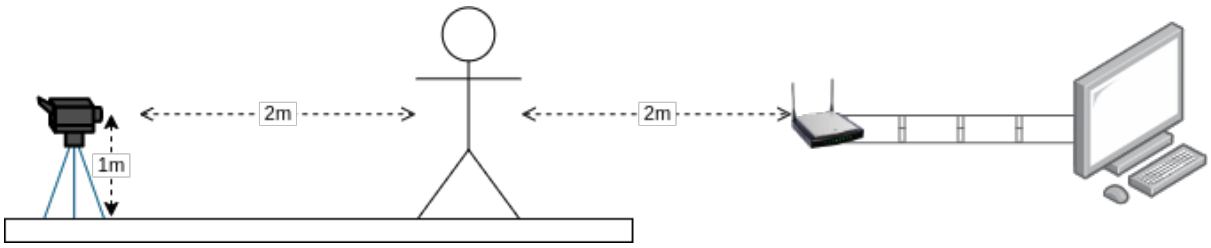


Figure 7.9: The general layout of the lab tests conducted by the researcher on a secure network.

The camera in use will be a mobile phone, capturing video at 30 frames per second, at 720p and aspect ratio of 16:9 unless otherwise specified.

The IMU data collected will be passed through the Motion Mirror application described in Section 7.2 and the results stored, much like the camera data, and processed after it has been collected. The collected data will then be analysed visually to ensure Action Tracer is correctly observing and representing the motions being performed correctly

7.3 Motion Capture Benchmarking

Having completed the system integration testing required for the Action Tracer system, the overall functionality of the system would need to be assessed in a context that it might be used in. This section will detail the experiments to be conducted that will be essential with relation to motion capture with a multi-marker motion capture system. The experiments in this section will be specifically crafted to assess how well this system addresses the research problem, as well as how well it performs compared to an established motion capture system.

Overview

To accurately test a motion capture system, a reliable benchmark system will be required. To assist in this, an established optical markerless motion capture system should be used side by side with Action Tracer and comparisons drawn. With this in mind, MediaPipe will be used alongside Action Tracer in various experiments.

These experiments will be done to compare the usability of Action Tracer in more realistic use cases. These tests were approved by the relevant clearance board as they will involve volunteers from the institution. These tests will build from where the system integration testing left off, but with more representative data sets.

These experiments will be conducted and assess the following factors,

1. Wearablility (Section 7.3.1)

2. Data Resolution (Section 7.3.2)
3. Accuracy and Precision (Section 7.3.3)
4. Range and Network Quality (Section 7.3.4)
5. Participant profile identification (Section 7.3.5
6. Motion profile classification (Section 7.3.6)

Experimental Set Up

The following section details the equipment used for testing Action Tracer against MediaPipe. In both cases, the Action Tracer system will utilise three nodes, while MediaPipe will be set to track the limb in use by Action Tracer by way of a single camera.

In order to conduct user experiments, three members of the institution will be asked to don the motion capture system and provide feedback on some of its more subjective qualities. While wearing the system, they will then be required to perform two sequences of motion. The first will be a basic chest fly, as is shown in Figure 7.10. The second motion will be to keep the start and finish positions the same, but changing the mid-point. This motion is shown in Figure 7.11. For both situations, participants will be holding a 1 kg weight.

Figure 7.10: The primary motion to be done by participants for this study. This motion requires that participants move straight to pose 2 without pausing.

Figure 7.11: The second motion participants are to do for this study. This motion requires that users first obtain pose 2 before proceeding to pose 3.

Participants will have to perform these motions over the course of three rounds:

1. Regular chest fly (Figure 7.10)
2. Modified chest fly (Figure 7.11)
3. Regular chest fly (Figure 7.10)

Much like the lab experiments in Section 7.2.3, the participants will stand in a designated area which is a fixed distance from the camera and data will be recorded and analysed afterwards using statistical analysis. A vital difference in these tests is that the camera will also be placed a radial distance of 2 meters away from the user, as opposed to a single fixed distance. This is shown in Figure 7.12. The goal of this addition is to allow marker occlusion to play a role in the assessment of the two systems.

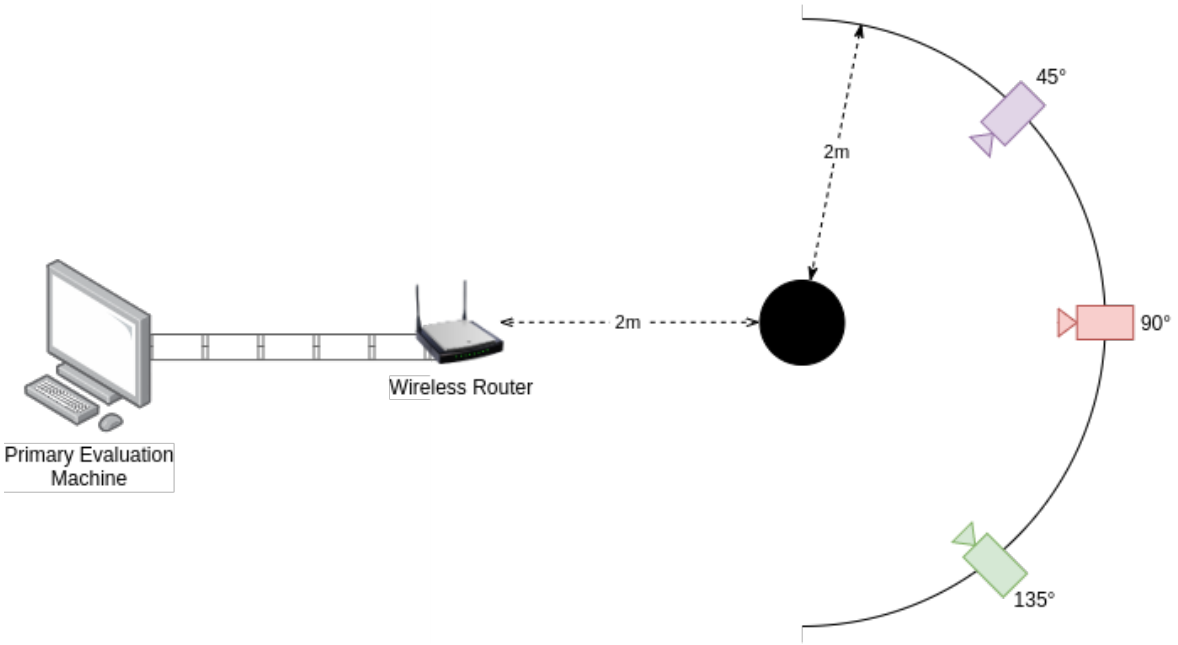


Figure 7.12: The general layout of the participant tests conducted by the researcher with cameras placed in various positions along a 2-meter arc.

Finally, in order to ensure consistency across tests, audio cues will be used to ensure participants lift and drop their arms as is required. The time signature for these cues is given in Figure 7.13. This ensures that a user will perform a given motion three times over the space of a minute.

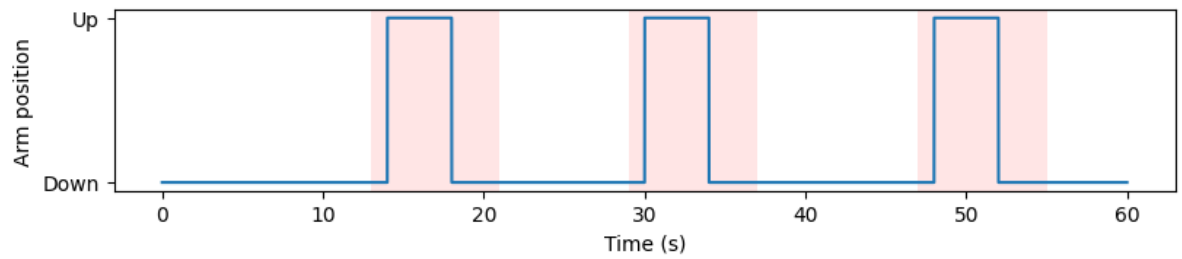


Figure 7.13: A graph showing the required actions during an interview with a participant relative to time. The time's highlighted are those to be used for analysis.

7.3.1 Wearability

Aim

The goal of this analysis is to obtain the thoughts of users on the way in which the hardware is worn, as well as finding out if wearing the system has any impact on the activities they will conduct.

Method

Much of the primary analysis is trivial and will be answers regarding basic questions such as,

1. How comfortable is the device?

2. Does the device fit well around your arm and body?
3. Do any of the nodes move around as you wear the system?

Due to MediaPipe being a markerless, optical system, the same questions would not apply. This alludes to MediaPipe not being examined in this first test.

For the second phase of this test, a look at the impact of participants performing tasks with the system on will be conducted. This will give data alluding to the impacts of wearing a motion capture system while performing various tasks, and if the losses as a result of this are negligible.

7.3.2 Data Resolution (Output Data Rate)

Aim
These experiments will relate to the ability to configure the data output rate of the Action Tracer system. While similar to tests done in the software packet transmission tests, these test will include multiple data nodes under a normal use case.

Method
To help gain a frame of reference, the markerless system will rely on a 30 frames per second video stream, which may only be decreased. Motion interpolation (generation of intermediate video frames between two frames) will not be used in this testing.

The effects of having multiple nodes in a system will also be tested, and the deduction of a node to data output rate ratio. This metric will quantify the system's inherent latency that arises as a result of the number of nodes, which would limit it usage in some sectors where such frame rates are required. As a means to gain a basic understanding of the inherent node latency, a general formula will be derived which specifies the maximum recommended frame rate per additional node. To obtain these quantities, a log of the data being transmitted by the Action Tracer system to a client device will be examined the number of node packets analysed and counted.

7.3.3 Accuracy and Precision

Aim
Experiments done in this section will quantify this system's ability to capture motion. By observing the accuracy and precision of the system, the true benefits of this system over markerless systems and other marker based systems will be much more closely evaluated.

Method
While most of the previous experiments focused on comparing the features and functions of the Action Tracer system, these experiments will put the primary purpose of the motion capture systems under testing at the forefront. This will be done by comparing the angles estimated between joints obtained from MediaPipe and Action Tracer.

Accuracy will be defined as how close the Action Tracer estimation will come to that of the MediaPipe estimation. Due to the lack of more reliable motion capture systems, MediaPipe will be taken as the golden standard for this metric. This will be analysed by first synchronising the data obtained from

Action Tracer to that obtained by MediaPipe, and analysing the two angle datasets by obtaining error measurements and obtaining a mean error value.

Precision will be defined as how consistent Action Tracer will be when it comes to obtaining data. This will be conducted by way of repeat lab experiments based on the same, repeatable action being conducted. Much like the accuracy tests, these results will be analysed to gauge the consistency of this system and will be based on the error measurement's standard deviation.

7.3.4 Range and Network

Aim
These tests will examine how well both systems will perform when subjects are at a distance from a key device in the systems. For Action Tracer, this will be a network access point, while for MediaPipe, this will be the distance from a camera.

Method
The range of this system will be assessed by conducting the experiments from various distances. This will be done in the lab to obtain a suitable distance for the MediaPipe use case, and for Action Tracer, this allows for the faults in data transmission to be observed and seeing how they will be handled by the system.

For Action Tracer, range tests will be conducted by placing the Action Tracer system a fixed distance from an access point while the user device will be a varying distance away from the access point. This will help simulate a primary use case for the system, where a coach or trainer may be walking around a subject. The distance and obstacles between a user and the system will be used as a means to obtain how the network's performance will be affected. The number of packets received over the course of a minute will be recorded as well.

With regard to MediaPipe, tests will be done moving a participant a given distance away from the camera and observing whether they will be captured by the system. At these distances, the ability to obtain angle data from the participant's limbs will be observed.

Finally, these tests will be done by observing the time delay between participants during their data collection sessions. This will yield indicative results relating to how the data was affected during participant tests. These delays would lead to a general lag that the Action Tracer system exhibits relative to real time, and may be corrected if needed based on the distance from an access point as well as the quality of the network.

7.3.5 Profile identification

Aim
Given that the data collected is collected from various participants under similar conditions, experiments under this section will be focused on seeing if this data may be used to uniquely identify an athlete, based on their profile.

Method
Having obtained the recordings of the participants following the given motions, the data will be extracted

from the zones highlighted in Figure 7.13. The three datasets will be averaged to give an average performance, which will in turn be their profile. This is done for both data captured by the Action Tracer system and the data processed by MediaPipe. This initial analysis will be done for their first and second rounds of recording.

First, the profiles will be compared to their MediaPipe equivalents and the consistency between the two analysed. This analysis will aim to see how well Action Tracer can replicate the profile generated by MediaPipe. This will be followed by comparing the various participants profiles first between MediaPipe datasets. All observable features will be highlighted and carried over to the Action Tracer datasets. The goal of this portion of the analysis will be to see if Action Tracer will be useful in creating profiles, much like the MediaPipe profiles.

The final test regarding profiling participants will be done by obtaining data from one of the participant's round 3 recordings. This profile will then be compared to those from round 1. Based on the profile most similar profile to the new profile, predications will be made on which participant is being examined.

7.3.6 Motion classification

Aim
This section's experiments will be aimed at seeing if an Action Tracer is able to distinguish between two similar motions sequences, independent of participant. This is similar to work done under the accuracy tests, however is targeted towards comparing profiles between rounds 1 and 2.

Method
Using recordings from rounds 1 and 2, profiles will be generated similarly to the profiles generated in the Profile Identification tests. This will first be done for each participant and the results compared between rounds 1 and 2 for each participant. The observations on this analysis will be noted and reported. These tests will be done across both MediaPipe and Action Tracer datasets.

These initial tests will be followed by creating a general profile for each motion will be generated by combining the three participants datasets from each round respectively. This will produce an average profile for each motion. These average motion profiles will be compared between MediaPipe and Action Tracer, with notable observations being discussed.

7.4 Summary

This chapter presented various methods that will be used to test and evaluate the performance of the Action Tracer system. First, experiments were presented on a subsystem level for the purposes of validation. These experiments were followed by system integration experiments, which sought to put the various subsystems together and ensure the system was working robustly. Finally, full system tests which aimed to compare Action Tracer to a known markerless system were done. This was to obtain the quality of the data being collected from the system as well as its potential conveniences.

Chapter 8

Results

In this chapter, the results from the experiments and tests conducted as specified in Chapter 7 are presented. First, some key results based on the initial testing will be provided in Section 8.1. This will be followed by the results obtained from system integration phase in Section 8.2. Finally, Section 8.3 presents the data obtained from the experiments relating to comparing the Action Tracer system and MediaPipe.

This chapter will only serve to present the data collected as well as provide some key findings from the testing, as well as provide a discussion of the importance of these results and how they fit into the entire project. Chapter 9 presents the conclusions drawn from these results.

The structure of this chapter will greatly resemble that of Chapter 7, and an overview of this is provided in Figure 8.1

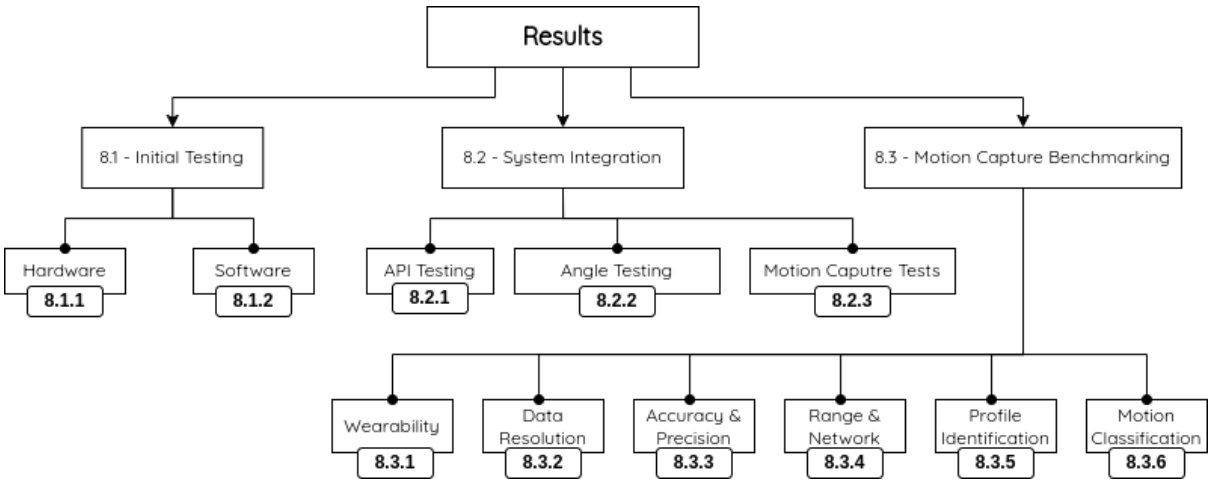


Figure 8.1: An overview of how this results section will be laid out. This overview also shows which sections of this chapter will house specific results.

8.1 Initial Testing

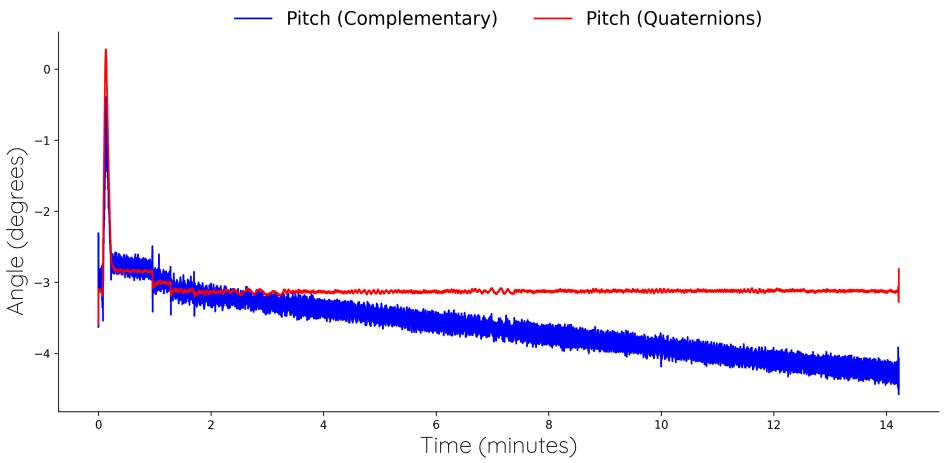
This section presents the results from the initial platform tests done to verify that the Action Tracer system could be built using the selected components. These tests were presented is Section 8.2 and will cover both the hardware and software tests. First, the hardware results will be presented.

8.1.1 Hardware

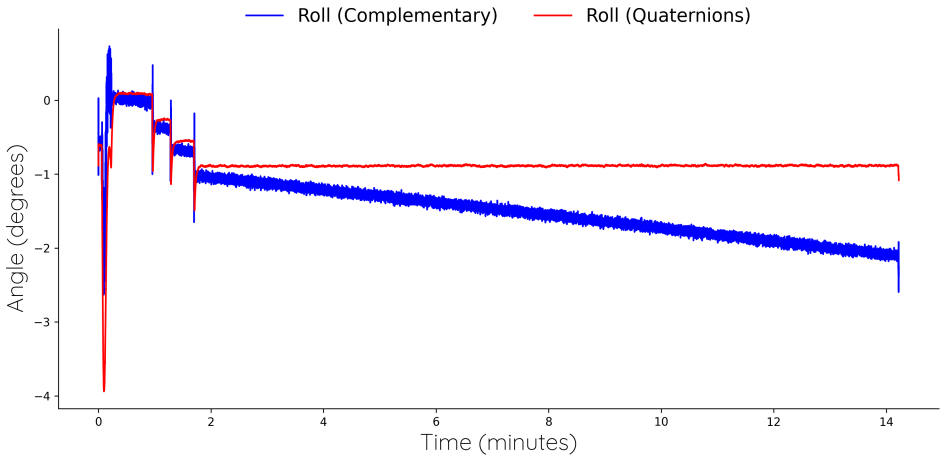
In this section, the initial tests relating to the hardware subsystem will be presented. Some results from this section will be relevant to the software subsystem's results.

8.1.1.1 IMU Drift and Calculations

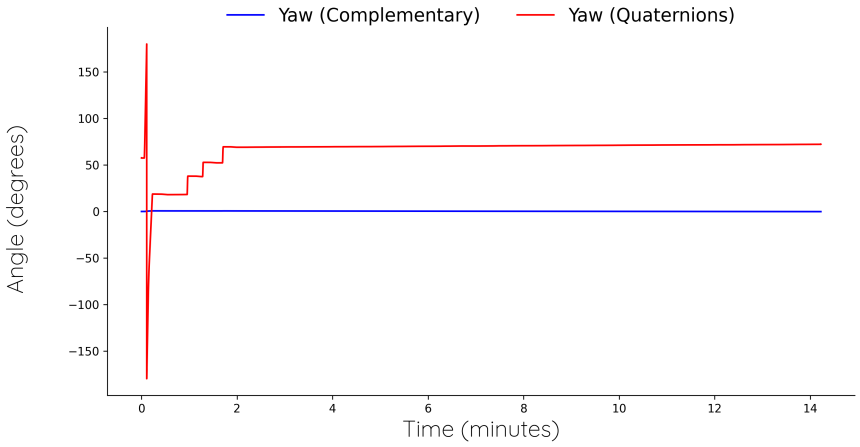
This section presents the results based on tests outlined in Sections 7.1.1.1 and 7.1.1.2. Figure 8.2 showcases the results from leaving the IMU stationary for 15 minutes.



(a)



(b)



(c)

Figure 8.2: The results of the stationary IMU tests converted to yaw, pitch and roll values with $\alpha = 0.05$.

By studying the pitch and roll data from this test in Figures 8.2a and 8.2b, it is clear that most of the data from the DMP quaternions closely matches the data calculated from the raw acceleration and gyroscope data. This is done with a complimentary filter coefficient, α, of 0.05. The key features of

these both graphs are the various spikes in the first two seconds, indicative of the initial calibration stage of this test. The quaternion data shows a much smoother data output with much less noise or rippling compared to that of the complimentary filter. Finally, looking at the stationary period of the test, the complimentary filter's output is greatly affected by the drift introduced by the gyroscope. This drift is expected and various values of the filter coefficient could be used to change the gradient of the drift profile, however it would not be as good as that which is shown by the quaternion data obtained from the IMU's DMP.

When it comes to the yaw data, it was expected that there would be a marked difference in the generated profile. Relative to the DMP output, the complimentary filter's data faired poorly. The output data was effectively 0 degrees throughout the test. The reasons behind this were outlined in Chapter 2.2.2.1, where the lack of a reference axis made it impossible to obtain a reliable yaw reading from the raw acceleration data.

By observing the errors between the two graphs, various comparisons may be made, as is shown in Table 8.1. This was achieved by interpolating the output of the DMP to match that of the Complimentary filter. With these two datasets, the mean and standard deviation was found using the difference of the two datasets for each Euler axis. The Mean Squared Error of each axis was then calculated for each axis.

Table 8.1: A brief analysis of the errors obtained from the drift experiments, presented in degrees between the filter and the internal calculation's data.

	Yaw	**Pitch**	**Roll**
Mean	65.4	0.534	0.577
Standard Deviation	18.2	0.376	0.423
MSE (degrees)	67.9	0.653	0.715

Based on this data, it was decided to use the IMU's DMP data, as its yaw component was more stable as it conformed to the expected profile relative to the movements being performed. The values of MSE for this component were indicative of how far off the complimentary filter was from the expected motions. It seems the proprietary algorithm used by the manufacturers of the IMU to calculate these values is able to take into account moments of motion and moments without motion. Comparing the other components, the mean error as well as mean squared errors are very low.

8.1.1.2 Feedback

With regard to the feedback system tests outlined in Section 7.1.1.3, the node attached to the rotating platform was able to flash the LED when it entered a given state. This state was defined as having rotated 45 degrees away in either direction of the initial position. In addition to this, this processing was being conducted on the processing device attached to the system supported the processing abilities of the system before data transmission as well.

This feature may be used to identify cases where impossible motions or rotations have been achieved, allowing recalibration warnings as well as for motion correction when being used in training environments.

8.1.2 Software

Having conducted the hardware subsystem tests, the software subsystem was tested, and this section outlines the results from the experiments outlined in Sections 7.1.2.1 to 7.1.2.5. These tests were used to form the foundation of the Action Tracer system. This section will not cover the results of the API testing, as this will be done in Section 8.2.

8.1.2.1 Transmission and Parallelism

This section will cover results from the tests described in Sections 7.1.2.1 and 7.1.2.4. The tests done to test the transmission capabilities were simple and showed that the system took on average $\approx 2ms$ to send data from to each of the clients that were connected to it. Building on this knowledge, only a single device was connected for the rest of the tests that would be conducted.

With regard to the parallelism tests, the results from various runs to collect data packets with a single IMU is shown in Figure 8.3. The threaded solution had an average of 3.98 seconds while the non-threaded, linear solution took an average of 4.11 seconds. Looking deeper into the data, it was found that the threaded solution had a greater standard deviation compared to the non-threaded solution with the results being 0.335s and 0.259s.

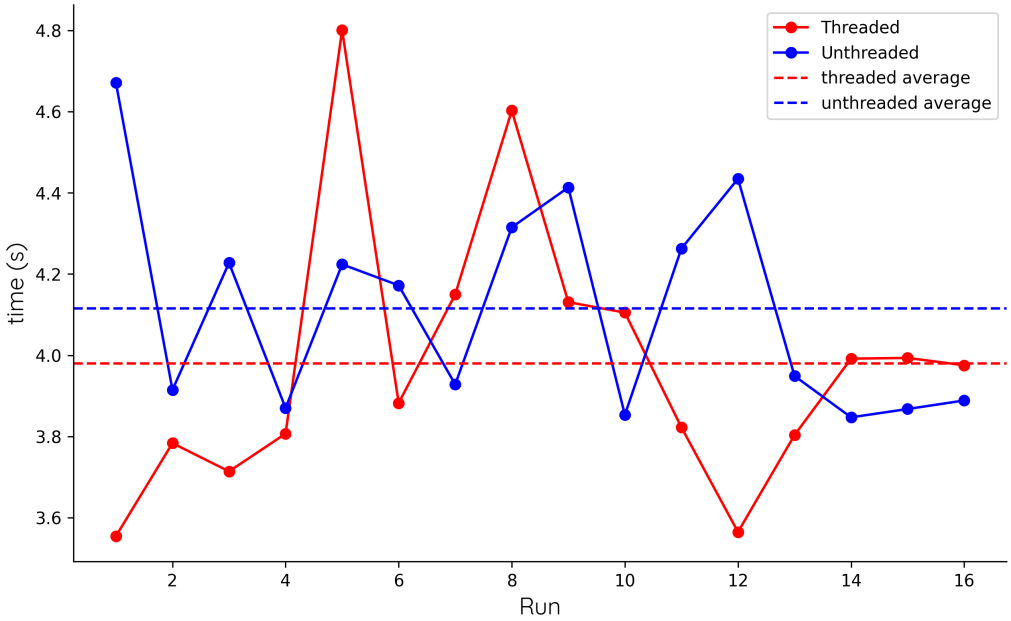


Figure 8.3: The results from the parallelism tests using a single IMU to capture and send 500 data packets over 16 runs.

The small increase in the average processing time would greatly provide better system performance as it allows for a higher system sampling frequency. The drawback with the threaded solution however is it's high variance, which means that when the system is performing at it's best, it is far better than

the non-threaded solution, however, when the threaded solution is not performing well, it performs far worse than the non-threaded solution. This may have big implications on the ability to process data in the system, as will be examined further in the processing and serialisation testing. For now, the benefits of a threaded solution will be carried forward as a general average better transmission time would be preferred.

8.1.2.2 Processing and Serialisation

This section presents the results from the tests and experiments outlined in Sections 7.1.2.2 and 7.1.2.3. These two sections have been grouped together, as they are performing similar functions in the pipeline of data collection and transmission.

First to be examined were the effects of implementing various filters before sending data to the recipient device. The average results of implementing these filters is shown in Figure 8.4.

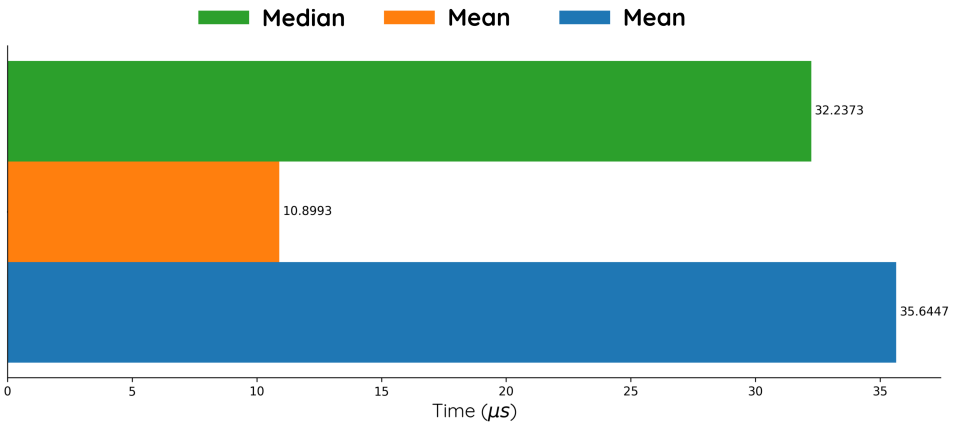


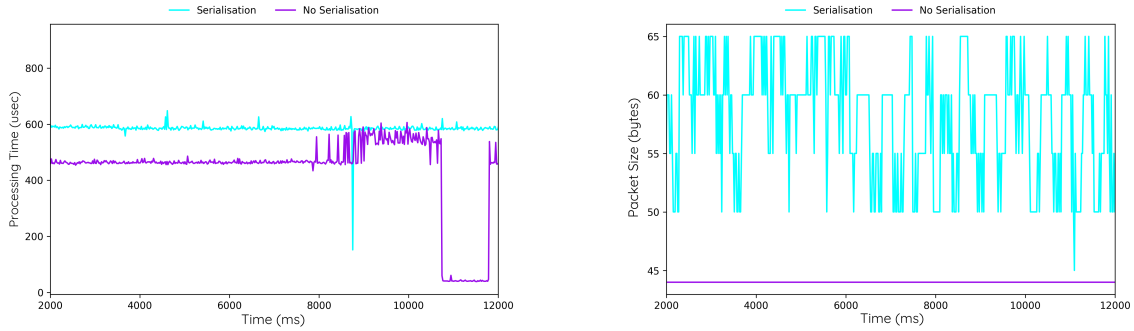
Figure 8.4: A bar chart showcasing the average times taken by each of the filters to process data before transmission, with a filter window size of 20.

The varying complexities of the filter implementations and their execution times show that the processor has plenty of time between transmissions on a small scale to allow various kinds of processing. Since this test was run with a collection rate of 200Hz (5ms per collection), it is possible to see how much free time the processor may have with a single IMU. This is shown in the calculation to follow where t_{free} was the amount of free time that existed between transmissions. t_{cycle} was the amount of time it took to complete a single cycle, $t_{\text{processing}}$ being the time it took to run the filtering and finally, $t_{\text{transmission}}$, the time it took to send the data packet from the device to an observing device.

$$t_{\text{free}} = t_{\text{cycle}} - t_{\text{processing}} - t_{\text{transmission}}$$
$$t_{\text{free}} = 5ms - 0.035ms - 2ms$$
$$t_{\text{free}} = \mathbf{2.965ms}$$

With the goal of exploiting this free processor time, the serialisation tests were done in order to use this

time as well as to assess the ease to modify the system. Figure 8.5 showcases these results with respect to the time taken to serialise the data packet as well as the change in the packet size as a result of the serialisation.



(a) (b)

Figure 8.5: Results from the serialisation tests run to see if there were any benefits to the serialisation.8.5a compares the time taken to serialize the data, while 8.5b showcases the variations in the packet sizes over the same time period.

From the results of serialisation tests, it is clear that this process not only takes longer but also make the packet size vary wildly. This is shown first by looking at Figure 8.5a, where the overall processing time for the serialised data takes on average $120\mu s$ longer to process with a little bit of jitter. This is in stark contrast to the non-serialised data processing which took less time but also proved to be inconsistent, showing great variations towards the end of the observation period. These variations were likely due to the processor having a heavy workload from other tasks, however no explanation could be given for the sudden dip between 10 and 12 seconds. As for the package data size, it was observed that the serialised packages not only varied more in size, but they were significantly larger than those from the non-serialised tests. This is due to the manner in which the serialisation library worked, where each variable is packaged according to the space it occupies in memory, as well as the transmission of tokens to denote the end of a variable. As the data being sent changes, the packet size will fluctuate as various variable type bit-widths change with each data point. Table 8.2 presents these variations more clearly.

Table 8.2: The variations in time to serialise data, as well as in the size of the packets alongside the percentage increase relative to the non-serialised data.

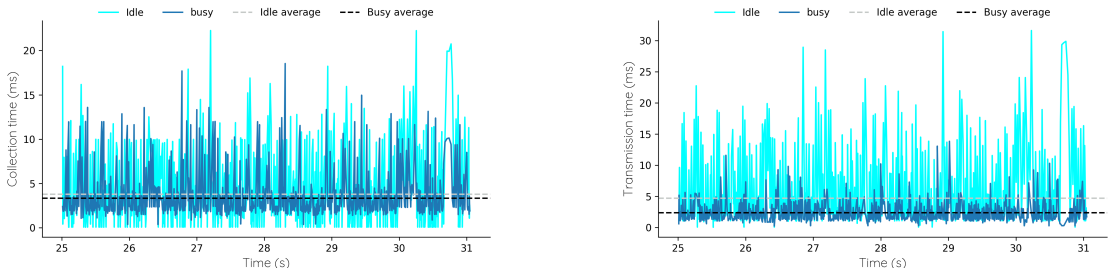
	Processing time (μs)	**Packet size (bytes)**
Maximum increase	205 (+37%)	21 (+48%)
Mean increase	120 (+28%)	13 (+30%)

Unlike the filter tests which filtered raw data, these tests were done with processed and organised data, which leads to the relatively high serialised processing times. The true differences between the time and size tests on average lies around 120ms and 13 bytes, respectively.

The results from 8.5b are telling of costs of utilising a serialisation framework. While there was a noted increase in packet size and processing time, it can be seen from Figure 8.5 that the variable packet sizes do not affect the processing time when using Google Protobufs.

8.1.2.3 Idle Times

Building up from the tests relating to threading and processing, the results from this section will address the test outlined in Section 7.1.2.5. Figure 8.6 presents a snippet from the results obtained from the test that was conducted.



(a) (b)

Figure 8.6: The results from the thread idle time tests. Figure 8.6a shows results from a section of the data from the collection thread, while Figure 8.6b shows the same instance in time for the transmission thread. In both cases, the average from the session is shown.

Based upon the method of collection and transmission used, it was expected that there would be less idle time in the collection thread as opposed to the transmission thread. Of interest however are the average times for the periods of busyness. Table 8.3 more clearly presents this data.

Table 8.3: A brief analysis of the variations in idle and busy times for the transmission and collection threads that were used in this test.

	Collection	**Transmission**
Average busy time (ms)	3.77	2.43
Average idle time (ms)	6.23	7.57

First, a look at the *busy* time for the collection thread should be examined. Granted, this test was done with a single IMU running at 100Hz, were this number to be scaled to a full body system with 17 IMUs, there would be concerns with regard to the final output data rate. This can be seen by using Equation 8.1 ot calculate the expected system sampling rate.

$$f_{\text{FBUR}} = \frac{1}{t_1 * n_{\text{IMU}}} \tag{8.1}$$

where t_1 represents the time taken to collect data from a single IMU, n_{IMU} is the number of IMUs in use and f_{FBUR} represents the Full Body Update Rate, the maximum possible system sampling frequency that would be available given n_{IMU} IMU nodes.

$$t_1 = 3.77ms$$

$$t_{17} \approx 3.77ms * 17 \implies f_{\text{FBUR}} = \frac{1}{3.77ms * 17}$$

$$f_{\text{FBUR}} \approx \textbf{15.6 Hz}$$

In this calculation, t_{17} represents the time taken to collect data from 17 IMUs. Figure 8.7 presents the estimated update rates for various numbers of nodes using the Equation 8.1 with various numbers of IMUs. This shows that inverse relationship between the maximum attainable frequency and the number of IMUs used during motion capture, with a very steep drop between 1 and 10 IMUs.

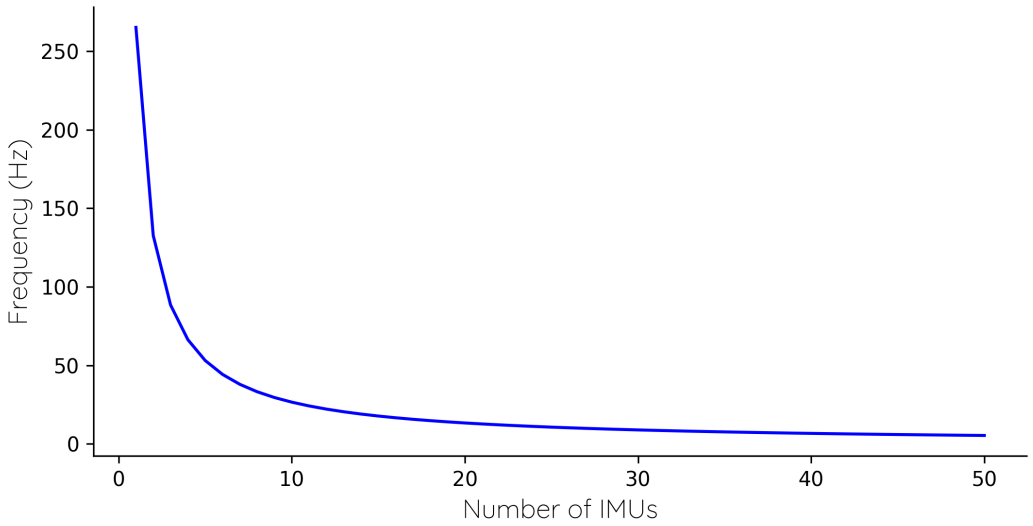


Figure 8.7: A graph showing the maximum frequencies that can be achieved with a given number of nodes.

8.2 System integration tests

This section presents the findings from running the system integration tests presented in Section 7.2. These results serve as a foundation for the motion capture benchmarking results, which are presented in Section 8.3.

8.2.1 API Testing

This section will cover the results relating to tests outlined in Section 7.2.1. Not much detail will be provided here, as most of the tests were resolved through yes or no answers. Tables 8.4 and 8.5 will present a brief overview of the success of each API when tested for functionality.

Table 8.4: Results from the various API commands for the Tracer Point API, grouped into their categories as is presented in Section 6.3.1.2.

Group	**Outcome**
Communication	These functions performed well to select and deselect devices. This was verified by using these methods to read data from various nodes and confirming that the feedback LED was being appropriately toggled.
Calibration	These functions did well to ensure that each IMU was set to 0 when called, and verified by confirming the output data values were as expected.
Data	These functions were able to collect the data from a given node granted the appropriate selection call had been made. The calculations additional to these calls were also sufficient.
Debugging	These functions greatly helped in the development of the tested system but allowing the outputs of the nodes as well as its present state to be show while being worked on.
Initialise	These functions were adequate in setting up each node having called the appropriate communication call and allowed for the customisation of individual nodes.

Table 8.5: Results from the various API commands for the Action Tracer Server API, grouped into their categories as is presented in Section 6.3.2.2.

Group	**Outcome**
Control	These methods were sufficient in placing the system into various states.
Initialisation	These methods did well to ensure that the nodes being used were all set as needed, allowing access to the feedback device as well as setting each nodes sample rate.
Management	These methods did in provisioning and preparing the system for capture sessions beforehand. While not able to work during a run, they were sufficient for the purposes of this dissertation
Workload	These methods did well to ensure each of the threads worked to achieve the goals of their states in their individual threads.

With regard to testing the API, messages between the Action Tracer system were sent successfully to all monitoring devices that were connected to it. This allowed the system to easily pass the three major sections of the API control system. When these methods are called on the Raspberry Pi, they successfully performed the relevant checks before execution, followed by conducting the relevant actions.

Of note during these tests was the incomplete nature of the Protobuf library, which has a different suite of functions available depending on the programming language being used. While this library handles the reception of data well in languages such as Python or Dart, the same was not true with C++. It was found that when receiving data in C++, the size of the data to be transmitted needed to be sent first ahead of the data itself. This is opposed to the decoding methods for other languages which do not need this data size to be known before transmitting. While the implementation of this workaround was simple, it shows that there is room for improvement in the implementation of the C++ Google Protobuf library.

8.2.2 Angle Testing

The test conducted in this section is described in Section 7.2.2, yielding the results shown in Tables 8.6 and 8.7 as well as Figure 8.8. With the goal being to track how well the system will will follow joint rotations using a custom angle joint, the angle between two IMUs is shown as increasing and stopping at various points for approximately two seconds.

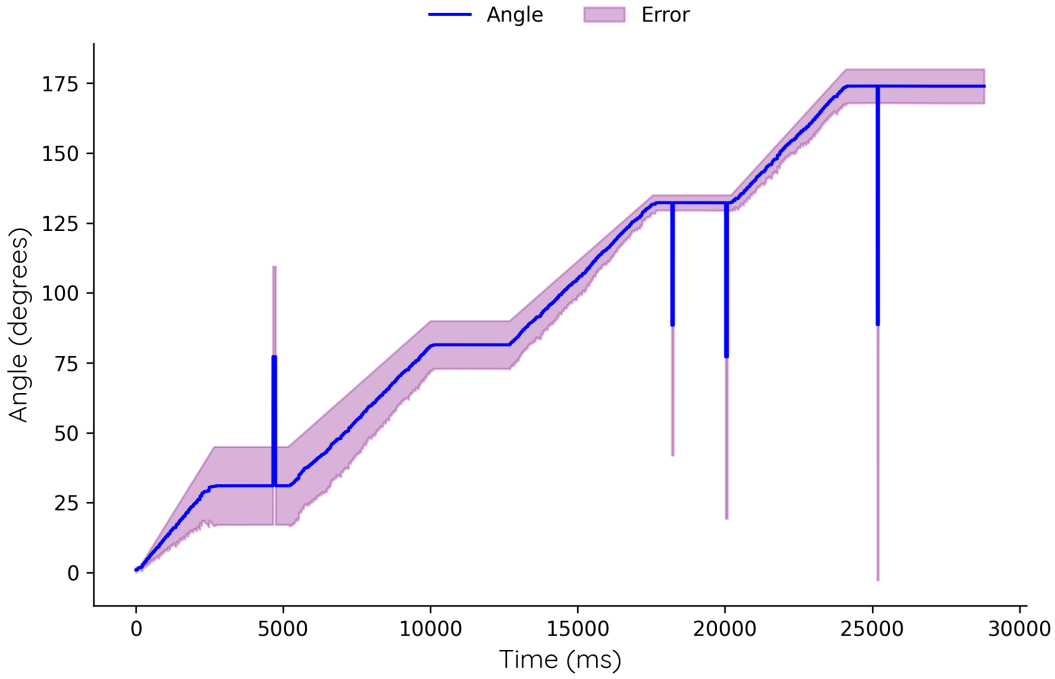
Table 8.6: The results from the angle tests between two IMUs, where one was stationary and the second was rotating, going upwards in steps of 45°.

Expected Angle	**Obtained Angle**	**Error**	**Error (scale) (%)**
45°	31.08°	13.92°	7.73%
90°	81.48°	8.52°	4.73%
135°	132.27°	2.73°	1.52%
180°	173.94°	6.06°	3.37%

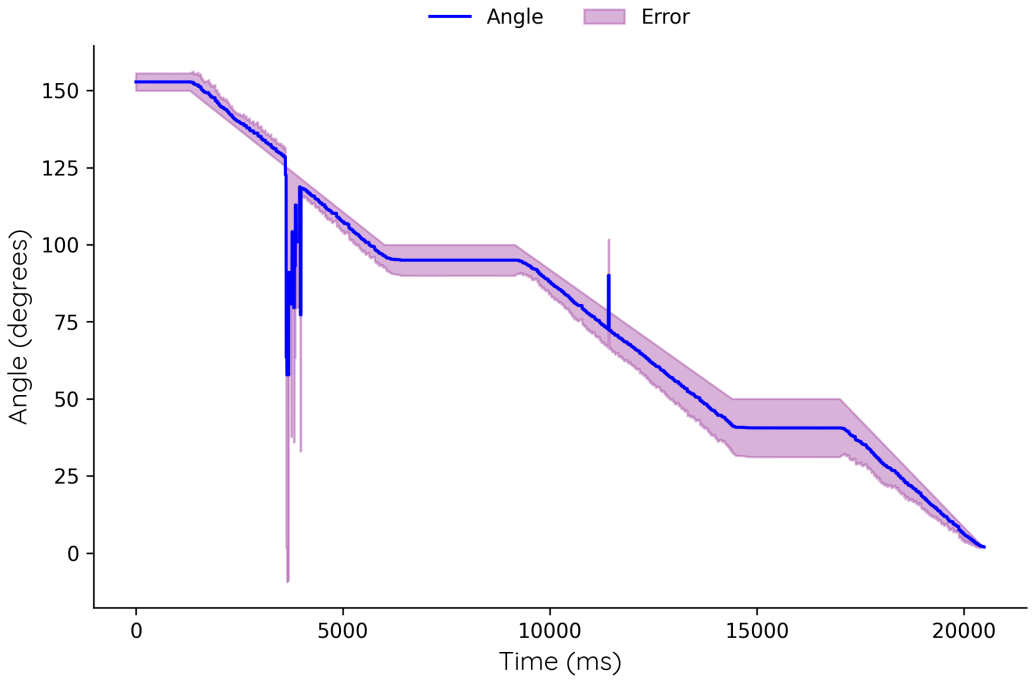
Table 8.7: The results from the angle tests between two IMUs, where one was stationary and the second was rotating, going downwards in steps of 50°.

Expected Angle	**Obtained Angle**	**Error**	**Error (scale) (%)**
150°	152.84°	2.84°	1.58%
100°	94.99°	5.01°	2.78%
50°	40.57°	9.43°	5.24%

The scale error is relative to the full range of motion that will be covered (180°) as is shown in Figure 8.8. The extent to which this affects the angles is shown in Figure 8.8.



(a)



(b)

Figure 8.8: Results from the initial angle tests alongside their respective errors. 8.8a showcases the first part of the test with an angular rate of, $+10°/s$ and 8.8b showcases the second part of the test with an angular rate of $-10°/s$.

Figure 8.8 offers a different perspective of the results presented in Tables 8.6 and 8.7. By first looking

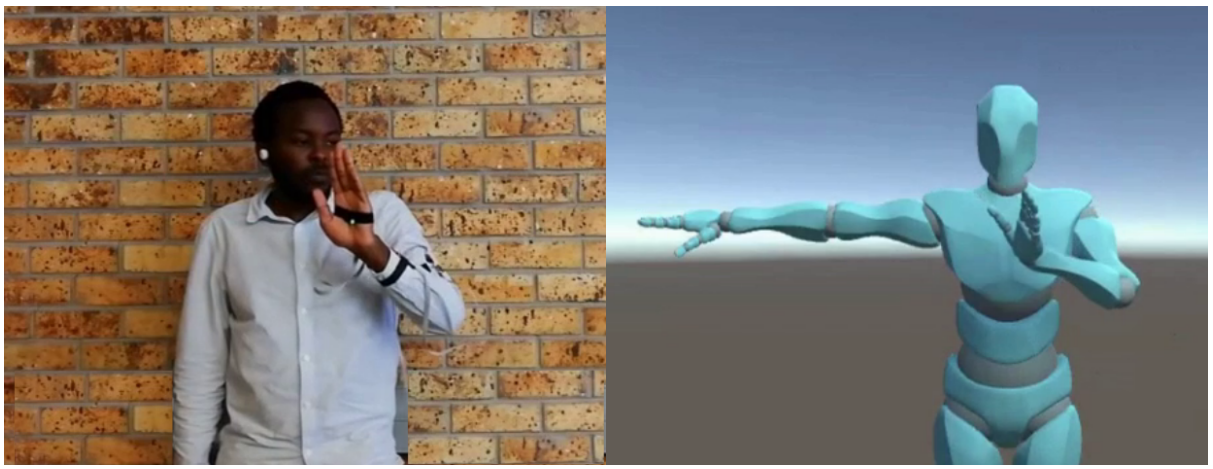
at Figure 8.8a, it can be seen that the angle between the two IMUs when moving is gently increasing between each of the steps. The rate at which this increase was caluclated by obtaining the gradient between the start point and end point of each step. Having done this, the average of the 4 steps was calculated and found to be $8.63°/s$, which is short of the anticipated $10°/s$. During this experiment, a number of outlier data points can be noted which may be due to the noise picked up by the IMU at this sample rate.

Looking at Figure 8.8b, the same method to determine the average angular rate was used as is described for the upward steps. The results of this process was that the rate of decrease with each downward step was on average $10.63°/s$, much closer compared to the expected $10°/s$. When compared the downward steps more closely matched the expected profile as opposed to the upward steps. There was however a longer period of noise observed just before the 5 second mark in Figure 8.8b which was likely sourced from the same reasons as those for the upward steps.

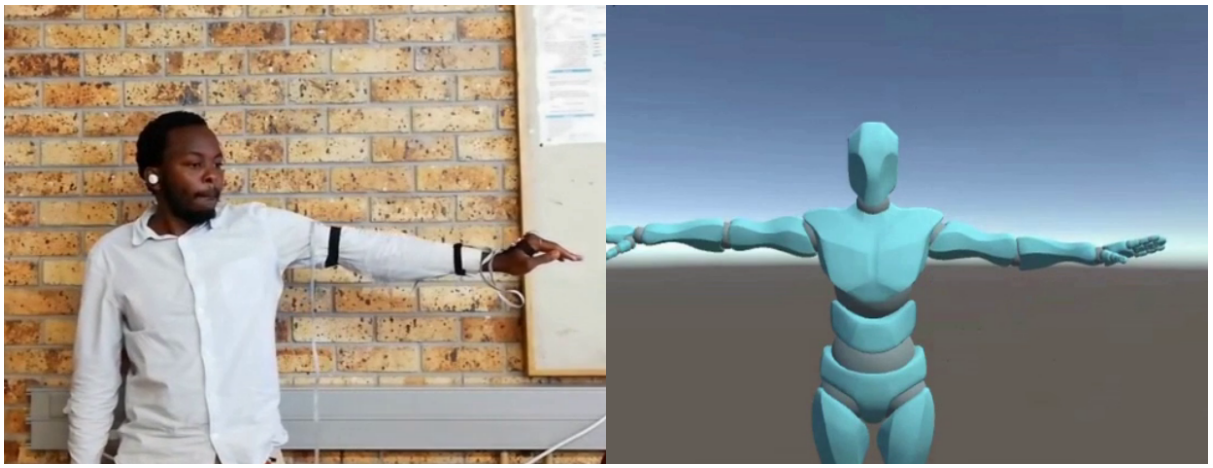
In both cases, the angular was set to $+10°/s$ and $-10°/s$ respectively and the measured angular rates were $8.63°/s$ and $-10.63°/s$. These results meant that at its worst, tracking would come with 13.7% of the expected angular rotation and at its best, tracking would be within 6.3% over short periods of time.

8.2.3 Motion Capture Tests

This section presents results relating to experiments outlined in Section 7.2.3. This will be addressed by showcasing images from tests that were conducted alongside those from the Motion Mirror application.



(a)



(b)

Figure 8.9: Images captured showcasing the motion Action Tracer system being used to track upper left limb motions in the Motion Mirror application.

As is shown in Figure 8.9, by observing the movement of the Motion Mirror model, it can be seen that the Action Tracer system is successfully following the wearers motions. While only two poses are shown in 8.9a and 8.9b, a variety of poses were also compared and visually confirmed the range of motion that could be followed by the system and these can be found in a short clip of a recording session found in the project repository, with the link being found in Appendix B.5.

8.3 Motion Capture Benchmarking

This section presents and discusses the results from the experiments outlined in Section 7.3. The data from these results was collected at once from the participant tests using the experimental setup outlined in the same section. To begin with, Table 8.8 gives the details of the participants that took part in this set of experiments. One of the participants was later asked to attend a second data capture session to be used for the multiple camera angle test.

Table 8.8: Particulars of the 3 participants that volunteered to assist with data collection for tests regarding profiling and the systems robustness

Participant	Height	Mass	Camera Angle (degrees)
1	179	90	90
2	168	67	90
3	174	80	90
Changing camera angle tests			
1	179	90	45, 90, 135

The Motion Mirror application required the participant's mass to estimate bone masses to aid with the application's physics engine. Figure 8.10 presents an example of how both systems could be visually compared to each other against the original video.

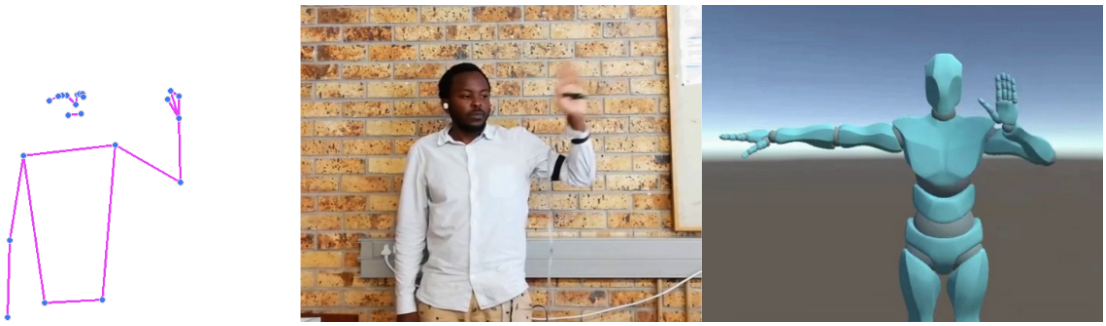


Figure 8.10: A compilation of images showing (from left) MediaPipe estimation, video recording and Action Tracer estimation.

8.3.1 Wearability

This section will address the experiments and tests with regard to Section 7.3.1 who's data was collected from the participant tests using the setup outlined in Section 7.3. While this section is more subjective to the different users, it plays a vital role in how this system may be deployed in future.

The consensus among the participants was the inability to move freely without the interference of cables. This specific issue would not be easily solved by the length of the cables or tidying up, but a fully wireless system may have been a better fit for this system. In addition to this, some nodes did not remain stationary on some areas of the body they were strapped to, even when strapped tightly. While these are issues with the physical materials used on the device, they are vital to good data capture within a recording session. As a result, some participants had multiple recording sessions for a single motion capture round.

Other than the given comments, participants were generally happy with the system strapped onto their body and had no further comments on the wearability of the system. With regard to comparing the achievable ranges of motion for participants, there were no noticeable impediments to the ranges of motion attainable by any of the participants.

8.3.2 Resolution

This section will provide the collected data with regard to the experiments and tests outlined in Section 7.3.2. To adequately address this experiment, a look at the recorded manifest from one of the participants sessions is presented in part in Table 8.9. Rather than present the entire packet data, only the essential details from each packet are presented where **Yes** is indicative of a node's data being present in a packet, while **No** indicates that a transmitted packet did not have data from a given node.

Table 8.9: A table containing 10 records from the data packets received from the Action Tracer system while recording a participant's session.

time(ms)	packet	Devices	Node 1	Node 2	Node 3
0	7659	3	Yes	Yes	Yes
20.7	7660	1	No	No	Yes
37.7	7661	2	No	Yes	Yes
54.8	7662	2	No	Yes	Yes
73.2	7663	1	No	No	Yes
92.4	7664	2	No	Yes	Yes
109.9	7665	2	No	Yes	Yes
133.5	7666	1	No	No	Yes
152.2	7667	2	No	Yes	Yes
170.6	7668	3	Yes	Yes	Yes
190.6	7669	1	No	No	Yes

While going through the data sent by the Action Tracer system, it is first noted that most of the time, data is not received from all the nodes available to transmit data. Though not present in Table 8.9, there were moments where all three node's data was received consecutively. On average, each second of data had 49 data packets being transmitted, which means an output rate of 49Hz. Further more the quality of each packet stood as questionable as not all packets contained all the node data.

Based on the participant datasets, it was noted that on average, 42% of the packets sent by the Action Tracer system had all three devices data present. 74% of packets had two device's data present, but all packets had at least one node present. This output rate was not limited for the tests, however this feature was not fully implemented in the API. It was however implemented for the various IMUs such that each node would reliably produce a new reading in a given time but this time was not lining up with the expected value.

Returning to the 49Hz output rate, this figure was well below the predicted maximum update rate based on Equation 8.1 of 88Hz. The more correct manner of presenting this statistic however involves incorporating the 42% of saturated system packets. This would give $f_{UR} \approx 20Hz$. There are various contributing factors to this value, which would include a potentially poor implementation of the thread management in the application, as well as having the IMU's not be ready for data collection when requested for data.

With regard to MediaPipe, since it relies on the video feed that it is receiving, its update rate will be equal to that of the camera being used. In the case of these tests, various data points could be picked up by the system, due to the amount of the participant's body in the frame. This is shown by the

captured wireframe in Figure 8.11. At 30 frames per second, and MediaPipe would collect approximately n landmarks a frame, the total data points a second would be n*30.

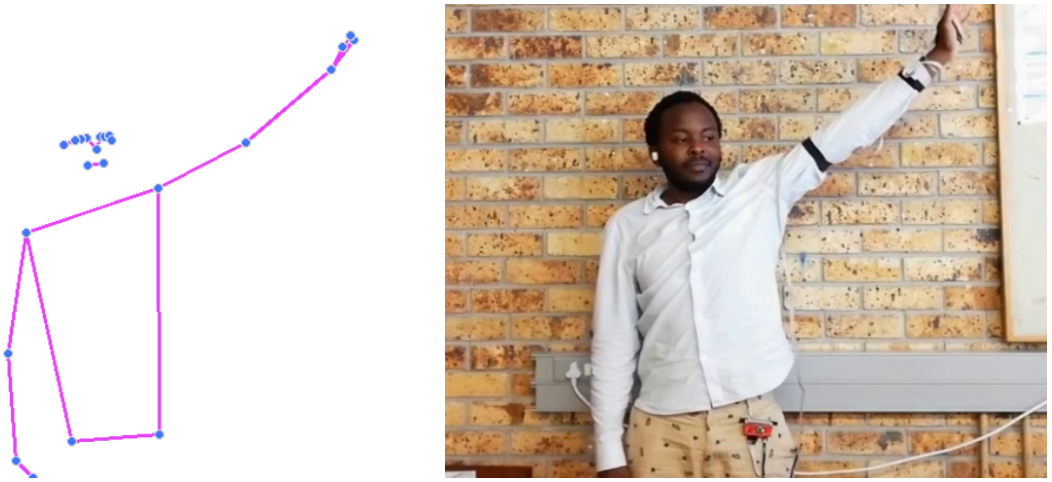


Figure 8.11: A side by side view of the amount of data MediaPipe can capture when in frame. There are 23 landmarks in total, with the face having 11 of the landmarks.

Comparing MediaPipe to Action Tracer, it can be seen that MediaPipe produces a more consistent dataset. Action Tracer, while collecting data from some nodes timeously, fails to reliable collect data from all nodes quickly enough. This may be indicative of a bug in the code used to run these tests which did not ensure that a data packet was collected from every node before transporting the data.

8.3.3 Accuracy and Precision

This section presents the results pertaining to Section 7.3.3 who's data was collected from the participant tests using the method outline in Section 7.3. This section will look extensively at the accuracy and precision of Action Tracer's estimations compared to estimations from MediaPipe.

To begin this analysis, Figure 8.12 presents all the elbow angle representations from Participant 1's session while they were performing motion 1. Four different estimations are presented, two representing the Action Tracer's 2D and 3D elbow angle estimations, and the remaining two present MediaPipe's 2D and 3D estimations.

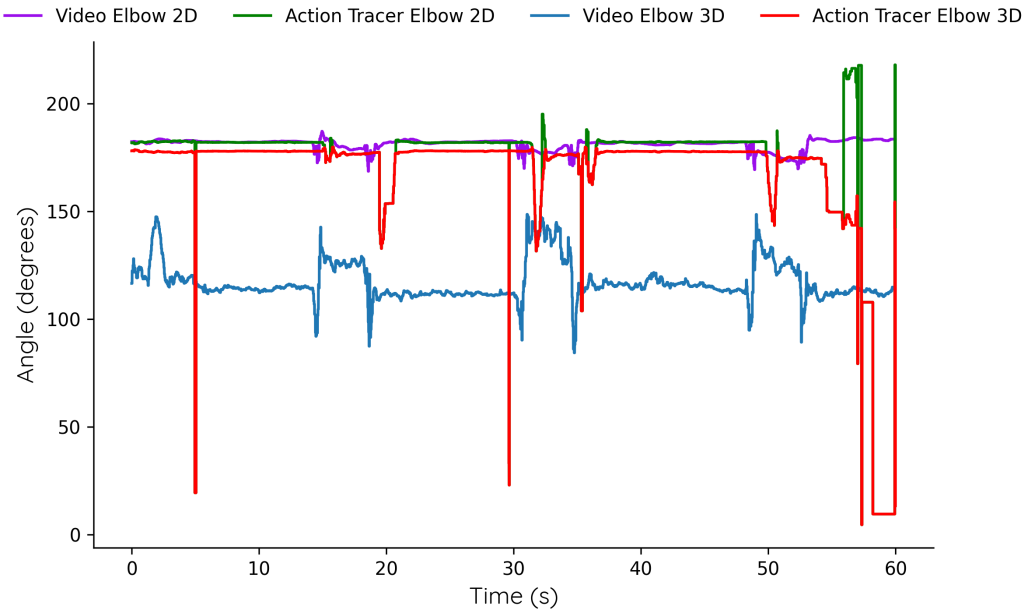


Figure 8.12: A graph showcasing the rotations at the elbow joint as estimated by both the Action Tracer system and MediaPipe using both 2D and 3D coordinate data.

The first observation to be made from Figure 8.12 was simply that MediaPipe's 3D representation is far off the mark when compared to the rest. This is show as the overall profile is underestimating the joint rotations relative to the other estimations. It can be seen that the moments of activity match the other profiles however the magnitude of these motions is also gravely amplified given the motion being shown required this angle to remain relatively stable, which was visually confirmed. This is expected however, as MediaPipe is given a single video stream and to obtain measurements with regard to depth would be very difficult and estimations of this would be very inconsistent. It must also be noted that the Action Tracer 3D data was also filled with more erroneous readings as is shown near the end of the recording session. Given this observation, to make a more fair comparison, only the 2D data obtained from both systems will be compared. Figure 8.13 presents the same estimations from the 2D data sets alone.

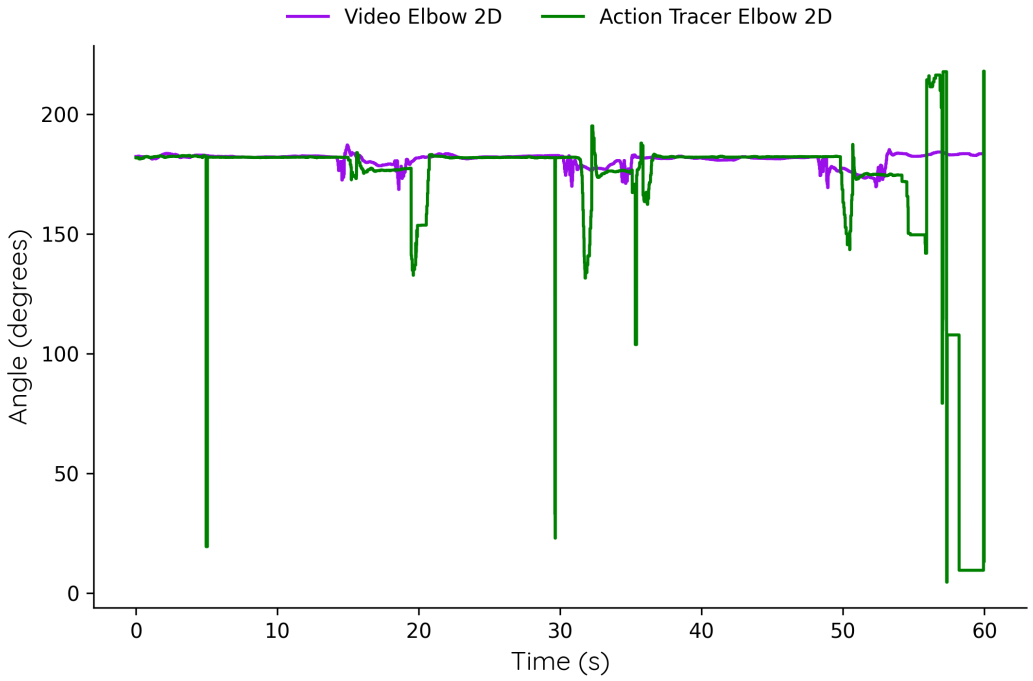


Figure 8.13: A graph showcasing the rotations at the elbow joint as estimated by both the Action Tracer system and MediaPipe using only the x and y coordinate data.

The item to note from these graphs is how similar they are in profile to each other. Visually, it can be observed that Action Tracer has peaks in very similar locations as observations made by both systems were made at the same time. While the magnitudes of peaks may wildly vary, there is also an observed latency in the amount of time which between the peaks being observed by MediaPipe and Action Tracer picking up the same motion. This will be further looked at in Section 8.3.4. As a starting point a small amount of confidence may start to be placed in the accuracy of Action Tracer system when compared to MediaPipe based on following the motion profiles being visually similar with the exception of the last 5 seconds. This may have been a result of a loose node slipping at the tail end of the recording.

While the given observations were inferred by visual observation, a deeper look into the data presents the histogram in Figure 8.14 which is derived from the difference in values between the Action Tracer and MediaPipe datasets. Due to the mismatch in data time scales, to better match the video stream, both the Action Tracer and MediaPipe estimations were run through a 1D nearest neighbour interpolation to allow millisecond approximations to be used for the data analysis.

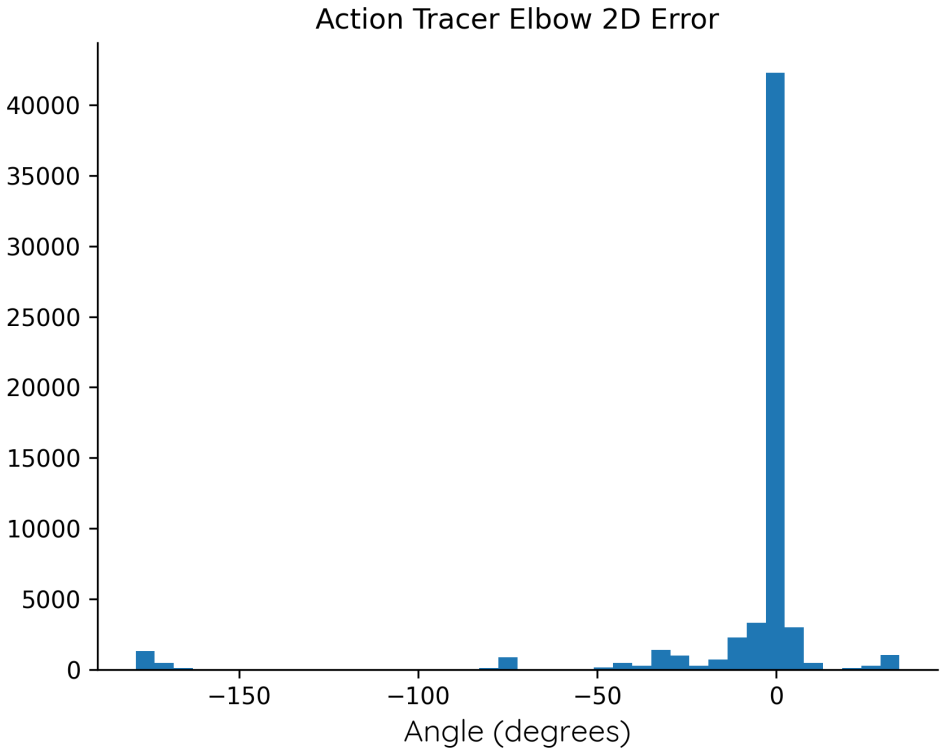


Figure 8.14: The error distribution between the Action Tracer and MediaPipe estimations over Participant 1's first motion.

Figure 8.14 showcases a very high count of observations where the difference was close to 0 degrees. A high concentration of the errors were within 25 degrees of the MediaPipe estimation. This dataset also showcases a number of outlier data points which result in very high errors which are found to be greater than 50 degrees from the MediaPipe estimation.

For all participants, the error distribution had a bell curve, implying that it could be modelled as a normal distribution. This finding is better represented in Table 8.10, which represents this data for all participants.

Table 8.10: An error analysis of the data collected from the three participants, recoded and analysed sessions based on Figure 8.14.

	Motion	**Part. 1**	**Part. 2**	**Part. 3**	**Part. 1**	**Part. 2**	**Part. 3**
			Elbow			**Wrist**	
Mean Error	1	-8.52	-46.35	-0.14	-7.99	-6.29	-7.15
Mean Error	2	5.48	-60.93	-5.39	10.16	-24.88	-11.56
Mean Error	3	5.80	24.03	-2.93	-2.69	-30.05	-7.68
Standard Deviation	1	37.51	51.34	25.03	47.70	52.18	24.47
Standard Deviation	2	32.77	37.31	12.04	12.64	26.72	22.82
Standard Deviation	3	27.80	51.56	24.69	29.05	61.61	15.98

Of note from Table 8.10 are the variations in data quality from Participant 2's readings at the elbow for

all their motions. Relative to the other participants, the values of the mean error was much larger. It was noted during the session, there was drift being observed by the Action Tracer system that could not be removed even after restarting the session. The true cause of this drift was not identified and would occur for unknown amounts of time and would stop after some time. Further pulling apart the data, it could be seen that the mean error for participants 1 and

It was however encouraging to see that the mean errors, apart from participant 2, were relatively low at the elbow, peaking at 8.52 degrees with the lowest mean error being 0.14 degrees. The deviations at the wrist were much greater however, this due to the noted difficulties in securing the IMU strapped to the back of the participant's palm. Unfortunately, while the mean error was relatively low, the standard deviations, in all cases, was relatively high, peaking 51.56 and 61.61 degrees at the elbow and wrist respectively, and the lowest standard deviations being 12.04 and 12.64 degrees at the elbow. It was unfortunate to note that there was no consistency between obtaining the best mean error and smallest standard deviation.

In general, the standard deviation from all errors was fairly large, presenting a maximum of 61.61 degrees for the readings at Partcipant 2's wrist. Relative to the achievable range of motion of 360 degrees, the overall error distribution based on the standard deviation is 34.2% at its worst and is 6.67% at its best with a standard deviation of 12.04, as was the case for Participant 3's 2nd motion. The general spread of the readings from their MediaPipe equivalent can be said to be fairly high as a result of this.

Based on these results, the accuracy(μ) of the system may be defined as 0.04% at best using the best result from Participant 3's mean error from their Motion 1 recording of their elbow, and 16.9% at it's worst when looking at Participant 2's elbow recording from their motion 2 data. The precision(τ^2) would be defined as 6.67% at its best and 34.2% at its worst. This is presented in degrees relative to 360°:

$$0.04^\circ \leqslant \mu \leqslant 60.93^\circ$$

$$12.04^\circ \leqslant \tau^2 \leqslant 61.61^\circ$$

A more absolute value of the accuracy and precision may be obtained by taking the average of all the means and deviations. This leads to a μ of 14.8° and a τ^2 of 30°, which give a scale error of 4.11% and 8.33%. When put into perspective, where MediaPipe would give an estimate of 100 degrees, Action Tracer would produce a reading between 85° and 115°. This is relatively high for an entire session recording of one minute with the lag not accounted for. A better analysis will now be presented where rather than looking at an entire session, each repetition of motion will be analysed. This is done by first isolating a repetition into an 8 second dataset, then correcting it's time scale for Action Tracer, and analysing it in much the same way as has been done already. One such example of this isolation and correction is presented in Figure 8.15.

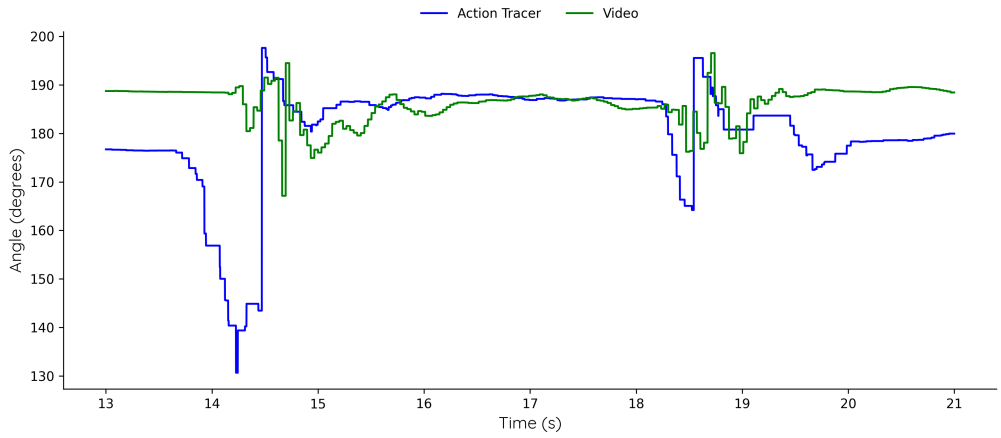


Figure 8.15: A graph showing the variations in motion between Action Tracer (corrected in time) and MediaPipe over the course of a single repetition of Motion 1 for Participant 1 at the wrist.

Examining a single repetition shows Action Tracer as having a greater amplitude of disturbance for both active portions of the motion. Both are in agreement with the direction and nature of the disturbance, however it is clear to see that Action Tracer picks up an extra disturbance between the 19th and 20th second of this sequence. This is better shown in Figure 8.16 which is a zoomed in view of this Figure 8.15.

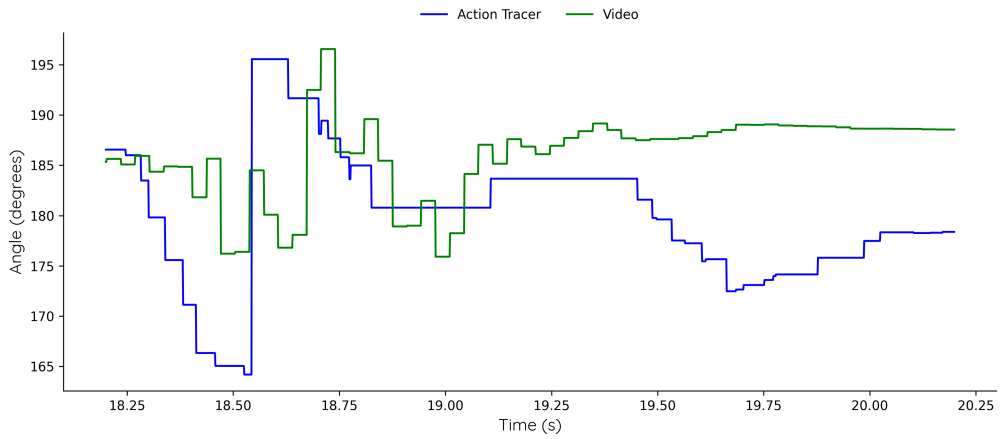


Figure 8.16: A graph showing the variations in motion between Action Tracer (corrected in time) and MediaPipe over the course of a smaller portion of Motion 1 for Participant 1 presented in Figure 8.15.

While both Action Tracer are in agreement, in how the motion occurred, Action Tracer picks up and extra drop in wrist angle before smoothing out. This and other similar observations in the dataset would support the notion of the Action Tracer system having a greater sensitivity, meaning that it picks up smaller motions that may be difficult to pick up on camera for various reasons. There is also a chance that the weight being held by the user interfered with MediaPipe's view of the hand making it miss this subtle change in angle. This result however points to the fact that Action Tracer is more sensitive to changes as opposed to MediaPipe, which serves as a benefit for small and fine movements. In the case

of Figure 8.15's wrist motion, MediaPipe did not pick much of the disturbance up.

When considering the MSE between MediaPipe and Action Tracer, it was found that for the entire motion in Figure 8.15, the MSE was 175, while for the zoomed in data showing in Figure 8.16, the MSE was 97. This shows that over small intervals, the Action Tracer is very closely matched to MediaPipe's estimations, with the exception being when Action Tracer picks up on movements not discernible in video. While this value of MSE may seem high, it must be noted that every second of data contained 1000 observations as a result of the interpolation used.

The key takeaways from this testing was that Action Tracer performs well to when it comes to picking up small motions. This is a benefit which arises due to it's use of markers which are much more precisely placed and providing readings relative to their location as opposed to markerless systems which estimate and superimpose a marker. Another key takeaway is the relatively low accuracy and precision which was a result of multiple factors such as markers not being securely fastened and drift errors which randomly appeared during collection.

8.3.4 Range and Occlusion

In this section, the remaining experiments and tests from Section 7.3.3 will be covered with both from the outlined tests as well as data collected from the participant tests using the setup outlined in Section 7.3. Section 8.3.3 discussed the accuracy and precision of the system by observing the dataset in a fixed and controlled environment, however this experiment will aim to see if real time data transfer will be possible with varying distances from the router as well as methods of connection. In addition to this, a similar phenomenon will be explored with regard to MediaPipe processing by varying the camera angle with Participant 1.

To first assess Action Tracer's data capture distance, the results shown in Figure 8.17 are indicative of a great range that data capture may occur in and would be limited by the network quality.

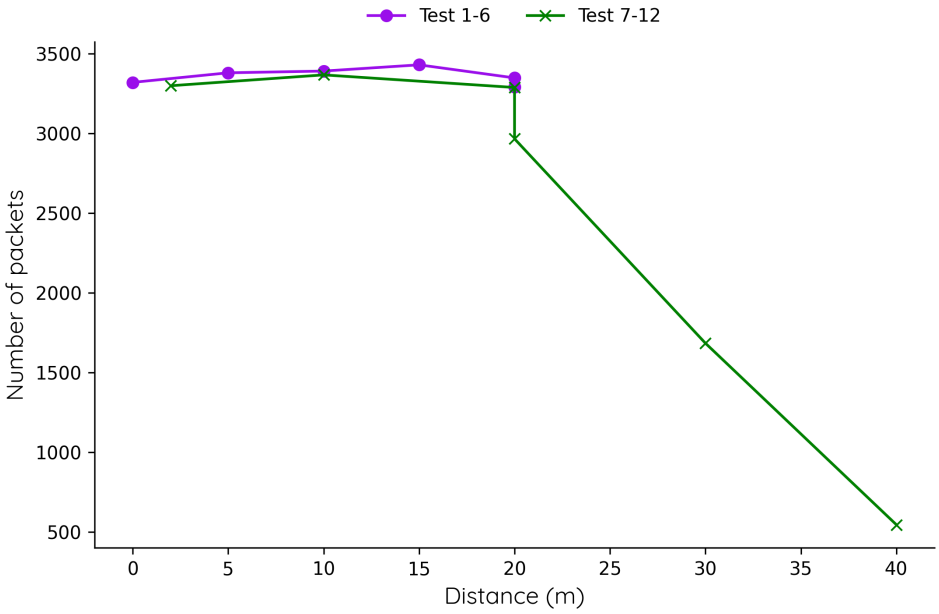


Figure 8.17: The variations between distance and the number of packets obtained in the various tests done to assess the range of the Action Tracer system. These are shown in two sets, the first, unobstructed, then the second with at least a single obstruction.

While the metric used in this testing is the packet count, a lower packet count is indicative of packets being lost in the network or being corrupted as they are being delivered to the monitoring device. In the first batch of tests (1 - 6), the results were collected with no obstructions, in an empty, open room. Test 6 was done with the monitoring device moving around this same room. Tests 7 - 12 were done with at least 1 wall between the access point and the monitoring device. Only Test 10 was done behind 2 walls, showing a noticeable drop in the number of packets received. Beyond 30 meters away from the access point, less than half the number of expected packets are arriving to the monitoring device.

When in the same room as the router, Action Tracer performs well and consistently for as far as could be tested, in this case 20 meters. With a relatively large packet count, the system was able to transmit the data captured well, even with motion as was done in test 6. This is an encouraging sign as one of the hopeful use cases of this system is with motions that do not require a user to be stationary. As for the tests where obstructions were present, the first 20 meters perform much the same as the tests conducted in the same room. Before going further, this leads to a question based around the distance to the router in this initial tests in the same room. A large enough space was not found to assess if distances greater than 20 meters would share the same ill effects of packet loss if the router was in the same room. What was noticeable was the packet drop between tests 9 and 10, which were done at the same distance away from the router with a different number of walls between Action Tracer and the router. This is a drop of roughly 200 packets which may not fully disrupt a recording session but would be noticeable in analysis. The further away the system from the router with a single wall obstructing the two object, the worse the system performed with less than 20% of the packets being received by the router.

To observe the effects of the network on the participants' data, first the data was sliced into repetitions, where it was found that the Action Tracer data was lagging the MediaPipe dataset. It was expected that this lag would be constant and due to network transfer rates, however Table 8.11 would shed light on the importance of network transfer on the data being collected from Action Tracer.

Table 8.11: A table that presents all the cumulative packet delays on the data collected by Action Tracer.

	Rep. 1 (s)	**Rep. 2 (s)**	**Rep. 3 (s)**	**Total delay (s)**	**Scaling Factor**
			Participant 1		
Motion 1	1.08	0.015	0.0	**1.095**	**1.018**
Motion 2	1.59	0.5	0.0	**2.090**	**1.034**
Motion 3	1.0	0.0	0.9	**1.900**	**1.032**
			Participant 2		
Motion 1	0.55	0.85	0.0	**1.400**	**1.023**
Motion 2	0.79	0.0	1.0	**1.790**	**1.030**
Motion 3	0.74	0.79	0.12	**1.650**	**1.026**
			Participant 3		
Motion 1	1.68	2.5	2.3	**6.480**	**1.101**
Motion 2	4.26	2.3	0.0	**6.560**	**1.109**
Motion 3	1.6	2.5	2.3	**6.400**	**1.107**

It was expected that data transfer would be consistent between tests, however the delay times when working with the primary machine were much lower (Participants 1 and 2) while those done on a secondary machine were much greater (Participant 3). The main difference between these machines was the operating system, with the primary machine running Ubuntu 22.04 and the secondary machine running Windows 10 Home leads to the conclusion that the operating system used to collect data matters as they have different ways of managing network packets internally. It must be noted that there were instances where there was little lag in the transmissions such as in repetition 3 of Participant 1's 1st motion. While not 0 seconds, the network quality can at times be good enough to allow this. This may also be affected by the router being used as its ability to handle packets is also of key importance in this test. On average, Table 8.11 shows that the data from Action Tracer was stretched as follows:

$$\delta = 1 + \frac{t_{\text{delay}}}{t_{\text{window}}}$$

where δ is the scaling factor, t_{window} is the length of the recording and t_{delay}. The scaling factor is used to correct for the delay in packet transmission and reception, where packets are slow to be received as a result of transmission delays in the network. The larger the scaling factor, the worse the delays in the network, This would lead to an scaling factor of $1.02 \leq \delta \leq 1.12$over a recording window of 60 seconds, when using the data found in Table 8.11. This translates to a 60 second recording being between 61.2 and 67.2 seconds long when using Action Tracer.

With regard to occlusion, the results obtained from the multi angle camera tests are presented in Figure 8.18. These were obtained by simultaneously capturing video form 3 different cameras and Action Tracer of Participant 1 performing their set of motions.

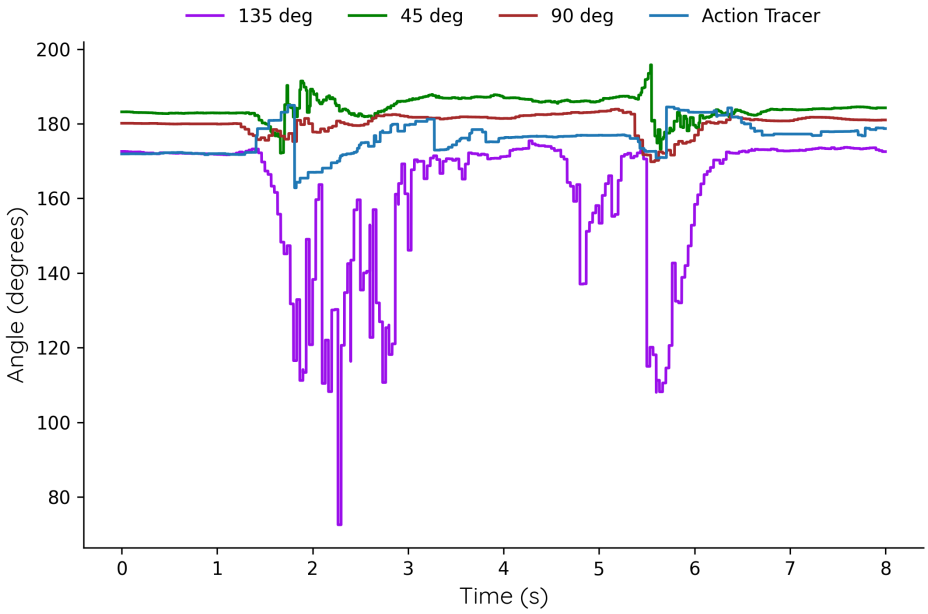


Figure 8.18: Graphs showing the elbow angle obtained from Action Tracer, as well as from various cameras, individually used with MediaPipe.

By first looking at the elbow angle estimations, it quickly be observed that the camera at 135 degrees produces bad estimations, even relative to it's companion cameras. This is because it was placed on the opposite side of the arm being monitored. In addition to this, the ability of MediaPipe to account for the angle distortion does not exist in MediaPipe. The camera to be most trusted was placed at 45 degrees, as it has a clearer view of the arm in motion compared to the rest of the cameras. When comparing this view to that of the camera placed directly in front of the participant, it can be seen that the estimations follow each other very well at the elbow, albeit with minor offset in the data of the 90 degree camera. Introducing Action Tracer into the discussion will show that it also does well to follow motion profile of the two cameras with relatively good views, showing similar forms of distortion over a single repetition. What is key to note is that Action Tracer, a marker based solution will not suffer from occlusion, no matter the motion the user attempts to perform.

When it came to looking at the wrist angle, shown in Figure 8.19, similar observations could be made as those from the elbow, The camera with the most obstructed view gave the most erratic readings supporting earlier claims about the occlusion. Action Tracer performed in much the same ways over the same repetition, following the same disturbance profile, however with readings higher than the estimates from MediaPipe.

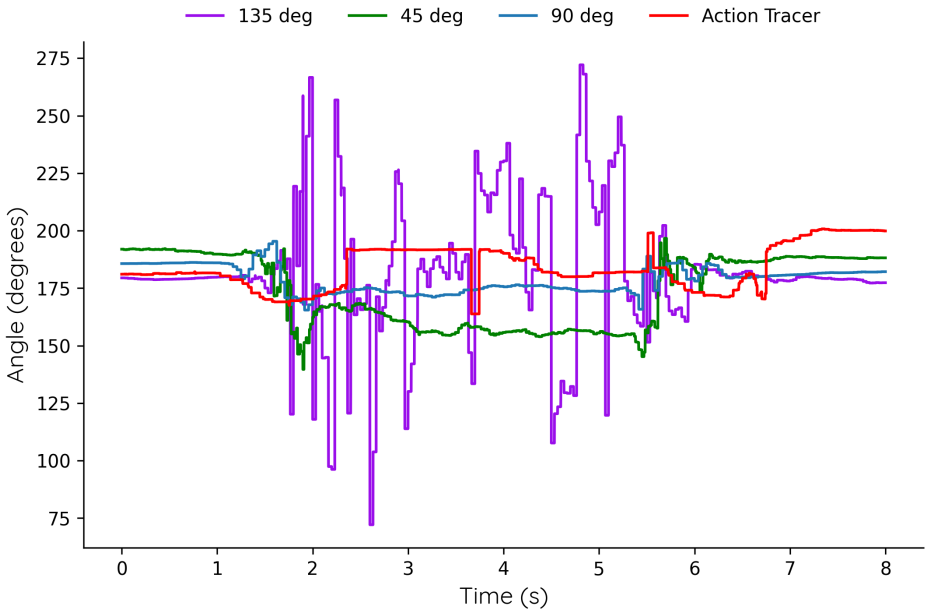


Figure 8.19: Graphs showing the wrist angle obtained from Action Tracer, as well as from various cameras, individually used with MediaPipe.

Overall, these tests showcased the effects of the network on Action Tracer. This revealed the grave importance of the networking equipment used as well as the distance from and number of obstructions between the router and the system. When it comes to the latency however, it was seen that Action Tracer would make recording sessions much longer due to transmission delays where as MediaPipe did not have these delays as each frame is delivered timeously. Finally, the effects of occlusion were tested by using various camera angles on the same motion, which showcased that with MediaPipe, camera placement is very important, while Action Tracer could work well without any markers being obstructed.

8.3.5 Profile Identification

In this section, experiments and tests from 7.3.5 will be addressed using data which was collected from the participant tests using the setup outlined in Section 7.3. The results presented in this section build up from the data collected from those presented in Section 8.3.3. The goal of the experiment was to see if Action Tracer could be used to identify individuals using the system performing specific motions. This would come about as a result of comparing the similarity and consistency of a user's motion. Of great note is the variation in participant heights, which would result in further variations in motion. In this vein, Figure 8.20 showcases Participant 1's 2D elbow data as estimated by Action Tracer during Motion 1.

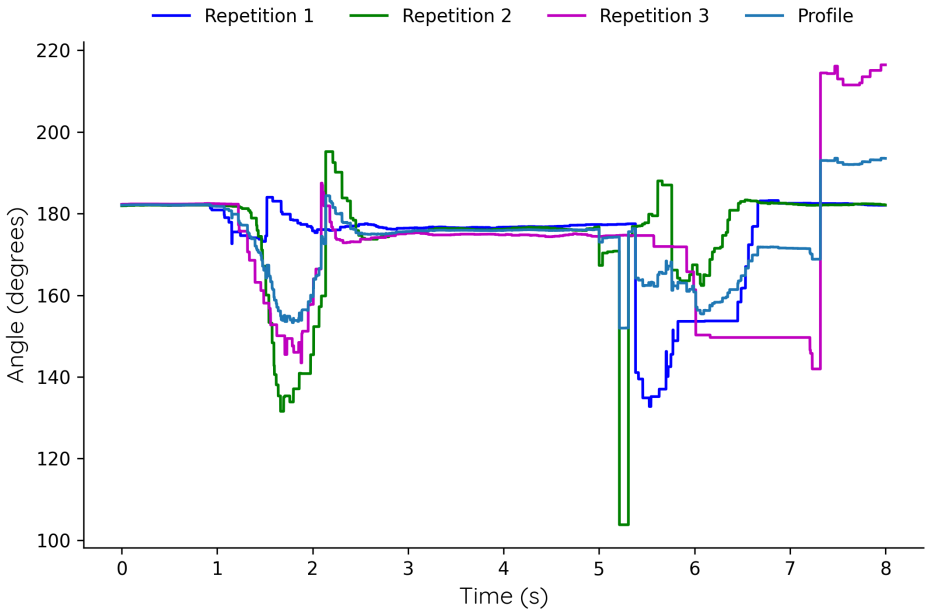
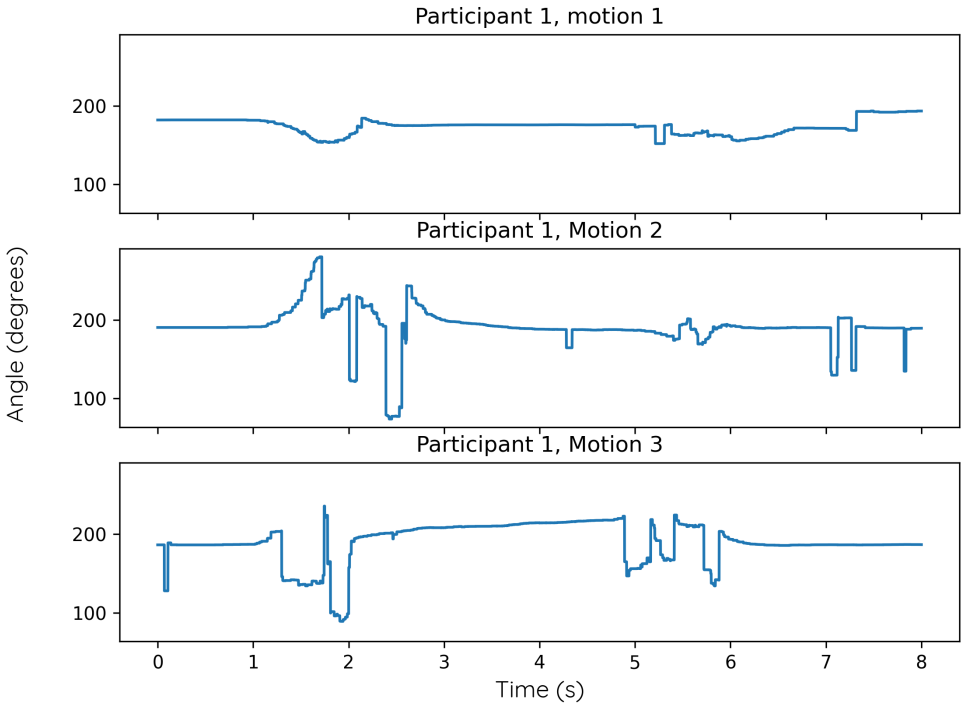


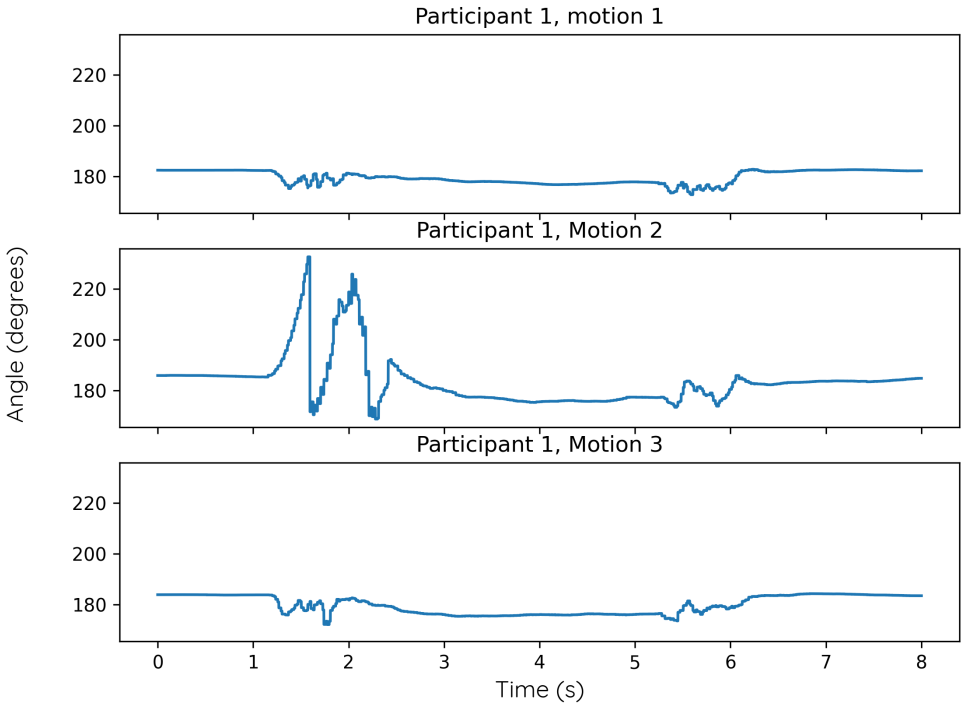
Figure 8.20: A graph showcasing all three repetitions of Motion 1 as well a profile based on their average elbow rotations as estimated by Action Tracer.

Looking at Figure 8.20, it can be seen that repetitions 2 and 3 have similar peaks and troughs around the 2 second mark, while later on in the motion, the second part of the distortion is much less consistent. With regard to the motion itself, this makes sense as participants were much more controlled in how they were lifting their arms to perform a motion (2 seconds) while lowering their arms produced mixed results which may have resulted form fatigue and discomfort(5-7 seconds). Of note is Repetition 1, which was not consistent with the other two repetitions with a lower deviation when lifting the arm but had a similar disturbance when the arm was being lowered.

With each motion being completed 3 times a recording, the various moment's of motion were condensed into eight second sections of motion. As a basic form of classifying a participant's repetition profile, the three data sets were averaged to produce a general profile which may be used to generalise what a normal repetition would be expected to look like for each participant. This is also shown in Figure 8.20. Using this principle, the same was done for each participant's elbow and wrist data for both Action Tracer and MediaPipe estimations, which would yield motion profiles such as those shown in Figure 8.21.



(a)

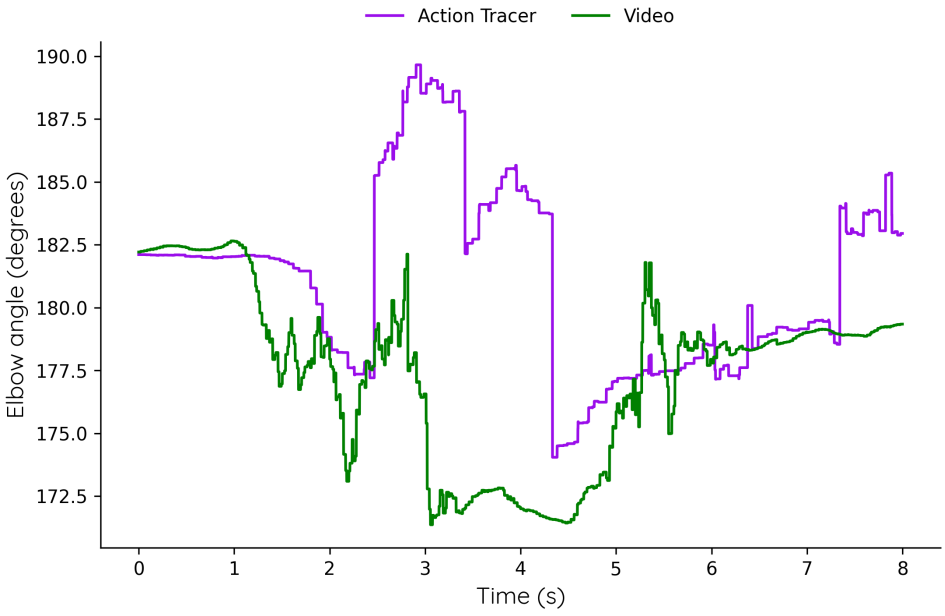


(b)

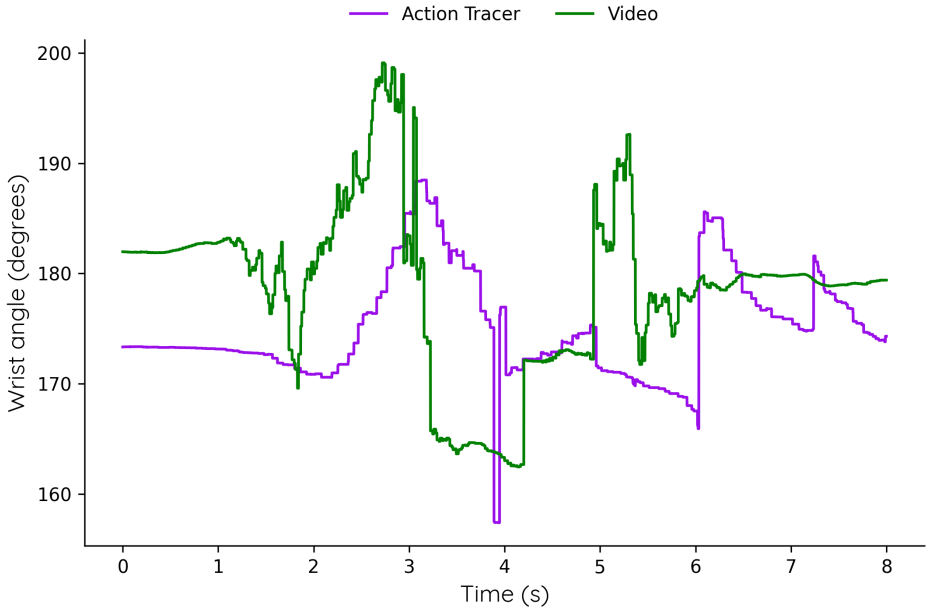
Figure 8.21: A view of Participant 1's motion profile based on the elbow rotation obtained from the 2D data estimated by Action Tracer (8.21a) and MediaPipe (8.21b).

Before moving forward, looking at Participant 1 motion profiles, many features are similar between the two profile sets. The fluctuations for all three motions appear to follow the same pattern. All profiles show two sections where the main action occurs between 1 and 3 seconds and between 5 and 7 seconds. The only exceptions to this are in Figure 8.21a, Motion 2, where there is a slight dip between 4 and 5 seconds. Apart from the presence of a disturbance, the nature and magnitude of the disturbance is also different. For instance, for Motion 2, 8.21a showcases a single two peaks 1 second into the motion as estimated by Action Tracer, while 8.21b showcases 3 more individual peaks in the same time period. This points to both systems noticing the same motion but with different levels of detection. Action Tracer seems to also be affected with a bit of noise, which led to the dip at around the 2 second mark of Motion 2 which would have greatly influenced the average motion profile at this time. Of great interest is the magnitude of Action Tracer's readings which could be attributed to the greater sensitivity of this system but also, looking back to the results from 8.2.2, the compounded error introduced when tracking motion may have played a roll in such large deviations.

Having created these profiles, this data could be used to begin the analysis described by the intended experiment. To begin, an example comparison between Action Tracer profile and MediaPipe is shown in Figure 8.22. The Action Tracer has not been corrected to account for the introduced lag.



(a)



(b)

Figure 8.22: A graph showcasing two profiles being compared to each other. Both profiles were obtained from Participant 1's elbow(8.22a) and wrist(8.22b) by Action Tracer and MediaPipe while they were conducting Motion 1.

Looking at the given profiles for elbow and wrist motion, there are multiple similarities that exist overall. Both feature two peaks for Motion 1 which fluctuate around 180 degrees, consistent with the expected results. Comparing both Action Tracer estimations, the same two peaks exist, but appear to have lagged the MediaPipe estimation, reasons to which were previously mentioned in Section 8.3.4. The differences must also be mentioned, which are the differences in the lower trough as well as the relatively stable

moment between 3 and 4.5 seconds in the MediaPipe data, which does not exist in the Action Tracer dataset when looking at Figure 8.22a. Similar notes can also be made around the same time period in Figure 8.22b where the MediaPipe estimation has a different deflection compared to that of Action Tracer, which will have peaks that lagged as well as had some noise affect its data.

Based on these findings, various participants may be compared, and similar features should be visible across their datasets relative to each device. Figure 8.23 presents all three participant's profiles for Motion 1 compared against each other relative to Action Tracer, while Figure 8.24 shows the same motion's profiles based on MediaPipe estimations.

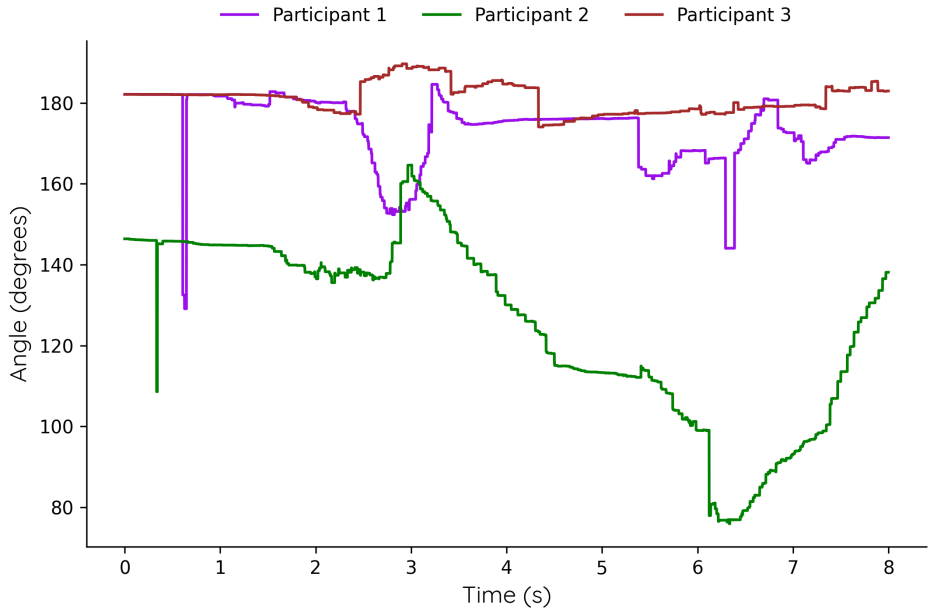


Figure 8.23: A graph showcasing all participant's Motion 1 profiles according to elbow estimations collected from the Action Tracer system.

To focus on the estimations obtained from Action Tracer, the two peaks are easily noted even in the case of Participant 2's erroneous reading at the elbow due to the drift that was observed in the recording session as was mentioned earlier in Section 8.3.3. Major differences arise when examining the magnitudes of the peaks at the elbow. Participant 2 has larger variations from the bases to the peaks, such as that between 2 and 4 seconds in Figure 8.23. This is highly consistent with the level of differences that the MediaPipe estimations in Figure 8.24 present, however, the nature of these differences is vastly different. Beyond these features, it remains to be seen if estimations could be made when motions from round 3 are compared to the baselines presented thus far which will be evaluated a bit later on in this section. Had Participant 2 had a dataset which was more consistent and not affected by drift, a conclusion may have been much more convincingly reached about how distinct the different profiles are. With regard to Participant 1 and 3 however, there is a level of distinction that visually exists as Participant 1 had a significant arm bend forward at about 3 seconds, while Participant 3 had s slight arm bend backwards.

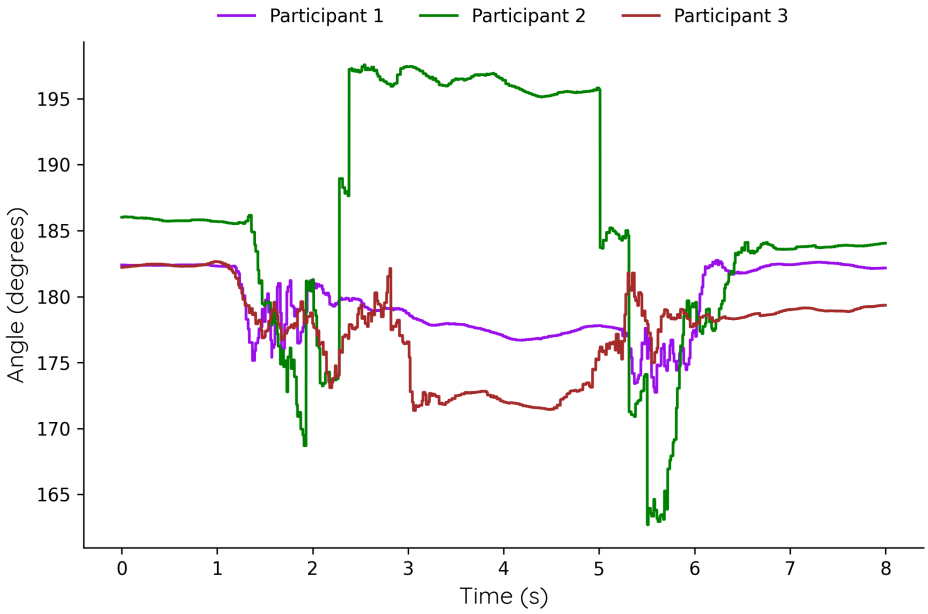


Figure 8.24: A graph showcasing all participant's Motion 1 profiles according to elbow estimations collected from MediaPipe.

Figure 8.24 showcases all 3 participant's general motion profile for Motion 1 at their elbows. What is consistent between profiles are the points at which the various disturbances occur, around the 2 second and the 5 second marks. What is however different are the different ways each participant's joint rotation changes after having raised their hand (between 2 and 5 seconds). Participant 2 appears to flex their hand backwards, hence the relatively high angle of roughly 195 degrees, while participant 1 keeps their hand fairly steady and close to 180 degrees. This is all different when compared to participant 3, who bend their arm forward slightly. Based on these observation, by inspection of the graphs alone, it can be seen that the three profiles indeed match three different individuals. When compared however to Action Tracer, these differences were not consistent with Participants 1 and 3 having inverse accounts of their estimations.

In order to dig deeper into these observations, multiple MSE evaluations were run between datasets from round 3 and the profiles previously presented from round 1. These findings are presented in Figure 8.25, various MSE matrices with the results from all the datasets.

Action Tracer Elbow

	Round 1 Participant 1	Round 1 Participant 2	Round 1 Participant 3
Round 3 Participant 1	989	5572	830
Round 3 Participant 2	11086	3777	12782
Round 3 Participant 3	245	3367	198

Action Tracer Wrist

	Round 1 Participant 1	Round 1 Participant 2	Round 1 Participant 3
Round 3 Participant 1	966	942	1287
Round 3 Participant 2	5529	6571	4870
Round 3 Participant 3	400	699	188

MediaPipe Elbow

	Round 1 Participant 1	Round 1 Participant 2	Round 1 Participant 3
Round 3 Participant 1	4	150	13
Round 3 Participant 2	47	278	27
Round 3 Participant 3	5	130	12

MediaPipe Wrist

	Round 1 Participant 1	Round 1 Participant 2	Round 1 Participant 3
Round 3 Participant 1	113	61	165
Round 3 Participant 2	132	298	281
Round 3 Participant 3	155	65	129

Figure 8.25: Four separate MSE matrices for the various MSEs run to view the similarities between participant profiles. The scale ranges from red (highest MSE) to green (lowest MSE).

By analysing the MSEs from Action Tracer's estimations, in most cases, it is clear to see that Action Tracer is able to somewhat accurately track the profiles of the various participants. The exception comes with Participant 2 who's recording was affected by drift. Despite this setback, there were still some of its estimations were decently close to the expected results as well as those presented by MediaPipe. Before getting to these estimations, it must be noted that the Action Tracer estimations feature very high MSE values as opposed to those from MediaPipe. This speaks to relative inaccuracy of Action Tracer to MediaPipe. While the magnitude of error may be greater, the predictions may still be correct, leading to good profile identification even though there is a great deal of noise in the datasets.

To understand these matrices, they should be analysed row by row then column by column. Should a given value in a row or column be the lowest, it will be considered the prediction from the analysis, this is a low confidence prediction. Where a value is the lowest value in both the row and column, it considered a high confidence prediction. It should also be noted that the values on the main diagonals are indicative of a match in motion being conducted while the main diagonal from the top left to the bottom right is indicative of a match in Participant and Motion type.

Using these definitions, the results from this test may be analysed as follows:

Action Tracer Elbow 3/6 predictions correct, with a pair of predictions having a high degree of confidence. (Participant 3 is identified in both Rounds 1 and 3 correctly.) The third prediction comes close to high degree of confidence, with the MSE difference being relatively small (Participant 2).

Action Tracer Wrist 2/6 predictions correct, with a pair of predictions having a high degree of confidence. (Participant 3 is identified in both Rounds 1 and 3 correctly) A third prediction is close to being correct but however has a slight difference in MSE (Participant 1)

MediaPipe Elbow 2/6 predictions correct, both of which are of low confidence.

MediaPipe Wrist 1/6 predictions correct, with low confidence. Only Participant 3's data was correctly predicted from round 1.

It must be noted that this test in not conclusive with regard to the abilities of Action Tracer and MediaPipe and only serves as a simplified comparison between the two system's being used for profile identification. For now, given the data analysis done, it can be concluded the MediaPipe does well with regard to participant profile identification with regard to the level of error, however, with regard to the number of correct matches, Action Tracer does better than MediaPipe despite its more erroneous data.

8.3.6 Motion Classification

In this section, experiments and tests from 7.3.6 will be analysed using data that was collected from the participant tests using the setup outlined in Section 7.3. The results presented in this section build up from the data collected from those presented in Section 8.3.5 where the method of collecting a profile is detailed. The aim of these experiments was to show if Action Tracer was able to aid in classifying different types of motion.

To begin, motion profiles were obtained from each of the participants in the same way as in Section 8.3.5 for all the rounds. The participants profiles are then combined by averaging them together to make a general motion profile. This average describes a particular motion based on multiple user's estimations for both MediaPipe and Action Tracer. Figure 8.26 presents this profiles for MediaPipe estimations and will later be followed by Action Tracer's estimations in Figure 8.27.

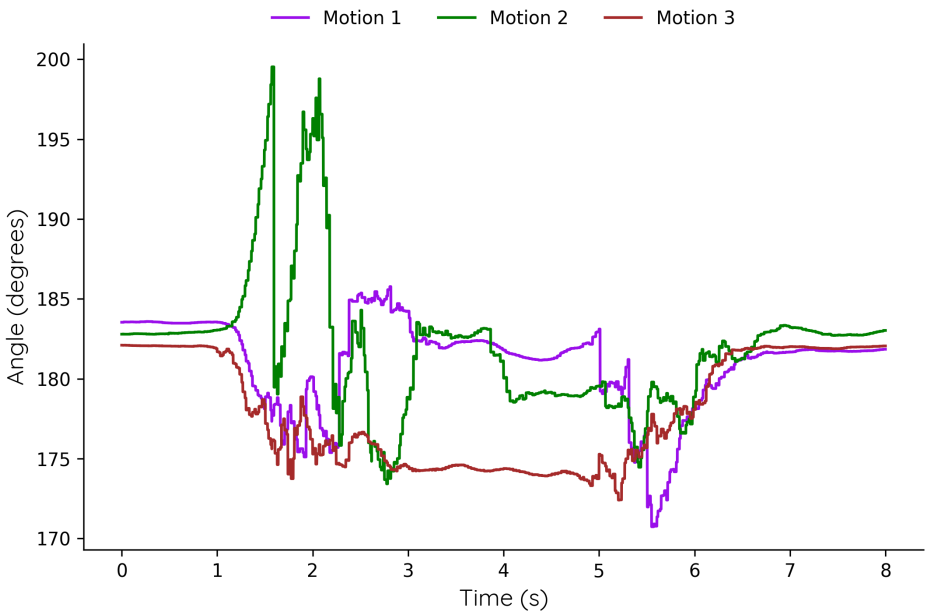


Figure 8.26: A graph showcasing the three motions and their profiles according to the MediaPipe estimations obtained at the elbow.

By first examining the MediaPipe estimations, it is clear that Motion 2 has a vastly different profile to

that of Motions 1 and 3, due to its peaks and fairly fluctuating middle region between 2 and 5 seconds. This is opposed to Motions 1 and 3, which are busy for short periods of time, with little various between these two moments. By visual observation, it is clear that Motion 1 and 3 are far more similar to each other than 2 to any other motion. Figure 8.27 will allow us to see if these observations may be carried forward.

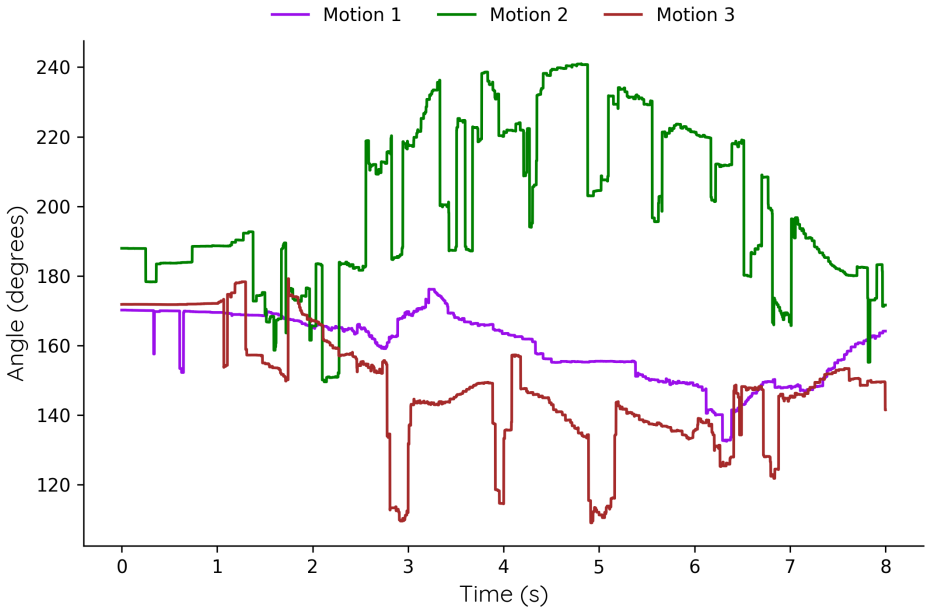


Figure 8.27: A graph showcasing the three motions and their profiles according to the Action Tracer estimations obtained at the elbow.

Much like with MediaPipe, the Action Tracer system showcases a vastly different profile for Motion 2 as opposed to Motions 1 and 3. Comparing Motions 1 and 3 however does not yield too similar a profile. Both Motions 1 and 3 follow a general divot, but there are a few more nuances with Motion 3 as opposed to Motion 1. To better assess the nuances between these motion profiles, Figure 8.28 presents an MSE matrix comparing Action Tracer and MediaPipe elbow and wrist motion profiles, as well as matrices comparing the individual system's predictions.

Action Tracer vs MediaPipe Elbow

	Motion 1 Action Tracer	Motion 2 Action Tracer	Motion 3 Action Tracer
Motion 1 MediaPipe	520	921	1318
Motion 2 MediaPipe	565	994	1343
Motion 3 MediaPipe	432	1130	1098

Action Tracer vs MediaPipe Wrist

	Motion 1 Action Tracer	Motion 2 Action Tracer	Motion 3 Action Tracer
Motion 1 MediaPipe	94	748	578
Motion 2 MediaPipe	121	887	586
Motion 3 MediaPipe	105	710	659

MediaPipe Elbow

	Motion 1 MediaPipe	Motion 2 MediaPipe	Motion 3 MediaPipe
Motion 1 MediaPipe	0	29	26
Motion 2 MediaPipe	29	0	39
Motion 3 MediaPipe	26	39	0

MediaPipe Wrist

	Motion 1 MediaPipe	Motion 2 MediaPipe	Motion 3 MediaPipe
Motion 1 MediaPipe	0	33	31
Motion 2 MediaPipe	33	0	62
Motion 3 MediaPipe	31	62	0

Action Tracer Elbow

	Motion 1 Action Tracer	Motion 2 Action Tracer	Motion 3 Action Tracer
Motion 1 Action Tracer	0	2361	327
Motion 2 Action Tracer	2361	0	3865
Motion 3 Action Tracer	327	3865	0

Action Tracer Wrist

	Motion 1 Action Tracer	Motion 2 Action Tracer	Motion 3 Action Tracer
Motion 1 Action Tracer	0	1000	599
Motion 2 Action Tracer	1000	0	1121
Motion 3 Action Tracer	599	1121	0

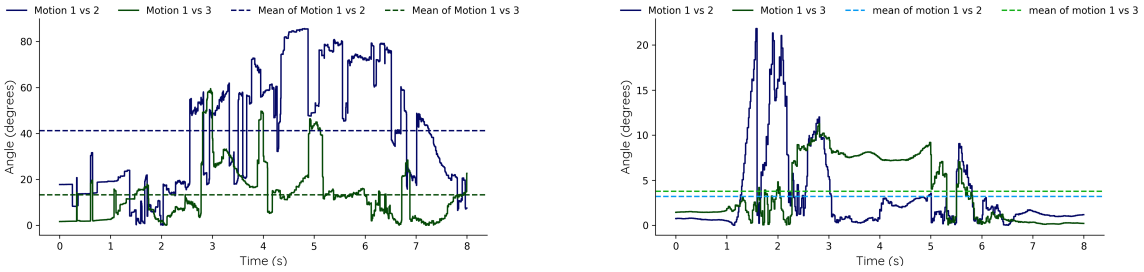
Figure 8.28: Six separate correlation matrices for the various correlations run to view the similarities between motion profiles. The scale ranges from red (highest MSE) to green (lowest MSE).

The same definitions of confidence will be used when comparing Action Tracer predictions to those from MediaPipe as were used in Section 8.3.5. By first looking at the results comparing the elbows between the two systems, only 2 motion profiles came close to matching up with low confidence, those being MediaPipe's Motion 1 and Action Tracer's Motion 3. With respect to the wrist angles, there was a single high confidence match with Motion 1 as well as a low confidence match with Motion 3 from MediaPipe.

When it comes to internal system motion comparisons, with regard to MediaPipe, the analysis shows that for both elbow and wrist comparisons, it correctly predicts the motion profiles presented. This is achieved by viewing the two lowest values in a row or column for Motions 1 and 3 or the lowest value for Motion 2. The difference between these MSEs however is quite small, ranging from 2 to 31.

When looking at Action Tracer, the same conclusions are reached as were discovered with MediaPipe. In all cases, the correct motions were predicted by the systems. With regard to the difference in MSEs, Action Tracer had a greater range of difference in angles ranging from 327 to 3538.

A different perspective to analyse this data by using Motion 1 to compare against Motion profiles for Rounds 2 and 3. Figure 8.29 shows these results for both Action Tracer and MediaPipe.



(a) (b)

Figure 8.29: Two profile error analysis graphs obtained by comparing Motions 2 and 3 to Motion 1. Figure 8.29a shows this for Action Tracer, and Figure 8.29b shows the same for MediaPipe.

One key similarity between the MSE matrices and this error analysis is the distance between the means and the distance between the MSEs for both Action Tracer and MediaPipe. In both cases, this distance is greater for Action Tracer comparisons as opposed to that from MediaPipe.

8.4 Summary

This section provided the results as well as discussion of the results. This was done to tie the project back to its core objectives and assess the Action Tracer system's capabilities, first against it's intended goals, then against an established system. This chapter serves as a backbone to the conclusion of this project by providing the key data points which were used to quantify the success of the system.

Diving into more detail, this chapter began by going over the results of the initial testing phase of both the hardware and the software subsystems in Section 8.1. Here it was found that the best way to continue the development and testing was to use the internal IMU position estimations. In addition to this, the decision to use a threaded application was also made as a result of the lower average time to transmission compared to a non-threaded solution. This testing also looked at the effects of processing data before transmission and added a serialisation method to make the data more accessible to other applications. This section was concluded by examining the effects of the number of IMUs connected to the system will have on the sampling frequency of the system, which showed a very sharp decline as the number of IMUs used increased.

The next phase of testing explored in Section 8.2 explored the first steps of the Action Tracer system being used. This was done by first working with the API developed to interact with the system successfully. Following this was an investigation on the systems ability to track basic rotations, which was the first area where the erroneous data output of the system was first observed. This section was then concluded by using the system as a whole on the left arm to see how well it worked to follow motion, which it performed well with.

The final stage of testing presented in Section 8.3 showcased a deeper look into the capabilities of the system as well as the potential hidden within the data which was collected. This was mainly done by running experiments with three different users wearing the system on their left arm and performing a given set of actions. This led to some initial comments around the system being collected which was followed by an investigation into the system's output data rate and level of data resolution available. The results of this investigation however revealed that the expected sampling frequency and that which was

observed were different, which pointed to a bug in the implementation of the software used for recording. The next test covered was relating to the accuracy and precision of the system where an error analysis was done over a 60 second interval as well as an 8 second interval where Action Tracer was found to follow MediaPipe well in some instances, while some tests showed that it did this tracking badly due to erroneous readings. Tests relating the the effects of range and occlusion were done next where it was found that Action Tracer performed well even when in a large room where the distance to the router was less than 20 meters, even with a barrier between the two. With regard to occlusion, Action Tracer did well as it was not interfered with in any way while the cameras filming the capture suffered bad readings as a result of not capturing the motions from the best angle.

The final two tests related to applications of the data which was collected. The first was to use the data to see if the data could be used to distinguish between individuals, In this test, it was found that both Action Tracer and MediaPipe could do this, however the ways in which this was seen in the presented graphs was not consistent. When looking further into this data by way of an MSE assessment, it was found that Action Tracer was able to get more profile identifications at a single joint compared to MediaPipe, The second application based test was found Action Tracer to be better at distinguishing between motions and providing a more clear estimation. This was shown through showcasing the higher level of resolution available as a result of Action Tracer's tracking.

Chapter 9

Conclusion

In this dissertation, the design and building of a low-cost motion capture system with API has been presented with the goal of being used in sport and rehabilitation. Such a system would be available in place of various other more expensive or difficult to replicate motion capture systems with options for feedback. Having successfully undergone an initial battery of tests showing that Action Tracer is functional as shown in Chapter 8, there are still areas of improvement that may still be explored and developed to better streamline the collection, processing, and transmission of motion data as well as its uses with other commercial applications.

This chapter will first present a summary of the dissertation, followed by an evaluation of the systems requirements. This will be followed by a report on some key findings with relation to Action Tracer and finally, some of the key areas that could be improved with regard to this system will be covered.

9.1 Summary of the Dissertation

The project contained in this dissertation was birthed out of a desire to build a motion capture system that could be built easily and be widely available at low costs to the user. This was done by first perusing the literature on the topic and presenting some key features and uses of such a system. Alongside this was the careful analysis of a few motion capture systems and their properties that were desirable to have. All this was contained in Chapter 2 of this dissertation and led to the development of the research objectives in Section 1.4.

Having prepared these objectives, the plan for building this system as well as design methodology was provided in Chapter 3. Following this, three chapters detailing the design of this system were provided, first with an overview of the system and how it will work in Chapter 4. The following two chapters detailed each of the two main subsystems that were outlined, those being Hardware and Software, in Chapters 5 and 6.

In order to validate the design outlined as well as assess the objectives of this project, Chapter 7 outlined the experiments and tests used to evaluate the Action Tracer system and the results from this testing were presented in Chapter 8. These results showed that the Action Tracer system met all the major requirements for a motion capture system that delivered data in real time to monitoring devices. This

system was able to attain data that was on average (without additional drift) within 6.3 degrees of the true value when compared to a markerless motion capture system while with the drift, the mean accuracy was 14.8 degrees.

9.2 Evaluation of requirements

This section will cover the degree to which the Action Tracer system met its intended requirements and specifications for all subsystems. Over and above this, the project's objectives will be revisited, and each objective will be addressed based on the findings of this project.

9.2.1 Review of requirements

This section will provide a review of the system requirements as provided in Chapter 4.

SR01: The system must be low cost

Given that the system was built using readily available off the shelf components, this requirement may be counted as being successfully met. As a result of this, the system may also be counted as easy to recreate. A bill of materials needed to build the system is provided in Table 9.1.

Table 9.1: The bill of materials required to reproduce the Action Tracer system.

Item	**Description**	**Quantity**
Raspberry Pi Zero W	The embedded platform that will run the motion capture and transmission software	1
MPU6050	IMU made by InvenSense	As required
3 mm Green LED	For feedback system	As required
470 Ω resistor	Current limiting LED	As required
10 strand ribbon cable	Connectors to the various IMUs	As required
Optional		
Action Node	A board that houses the feedback system and IMU	As required
Action Main Node	A board that houses a single IMU and attaches to the Raspberry Pi	1

SR02: The system must be able to capture human motion in real time

This requirement was successfully met as a result of the Action Tracer system's ability to collect rotation data from various IMUs and proceed to send this data over a Wi-Fi network. The system was able to track joint angles, with relation to MediaPipe, with an accuracy and precision of 14.8° and 30° respectively. In addition to this, the system was able to send packets to connected monitoring devices within 2ms of collecting the data.

This is supported by the results obtained in Sections 8.1.2.1, 8.2.3, 8.3.2 and 8.3.3. In these tests, it was found that the system was able to send data to other devices with a decent frequency and follow the motion carefully.

SR03: The system must be usable from a distance

By conducting tests relating to the physical system's usability at various distances from a wireless router, it was shown that the Action Tracer system was capable of transmitting data reliably from at least as far as 20 meters while MediaPipe could only attain 10 meters. This was further assessed by assessing the impact of various camera angles on MediaPipe, where it was found that Action Tracer does not suffer from some common issues found with regard to occlusion. Having assessed these requirements, it was clear that the system built had successfully met this requirement based on the results from Section 8.3.4.

SR04: Achieve a configurable system

By creating a system that allowed for user defined processing to be programmed into the firmware, the usability of the system is greatly increased. While applications in this dissertation were limited to basic gym movement, the system is configurable to the extent of being used in animation, rehabilitation as well as in conjunction with artificial intelligence applications. This is all supported by the results from Section 8.2.1 as well as Section 8.3.1 which mentioned the wearability of the system.

9.2.2 Review of objectives

This section will provide a review of the project objectives as provided in Section 1.4 based on the findings of this dissertation.

Can a system be designed to help users learn a sport/activity?

Based on the Action Tracer system's ability to track motion as shown by the results in Sections 8.1.1.2, 8.2.2, 8.3.3 and 8.3.4. Given the level of detail attainable from the system, it is possible to use set points which will allow for the feedback system to be active to alert the user when a desired or undesired motion has to be performed. The results relating to the profile and motion identification are also indicative of being able to consistently identify not only an individual, but if a motion is following a particular pattern. This means that it can be identified using various computational methods which are not inherently in the system but could be added in future.

The system was capable of performing well even when not in good view of a monitoring camera. The ability to also be used at a distance may be vital for activities that require large amounts of space or isolation.

Can the data collected by such a system be used in assessing an athlete's risk of injury?

There were steps taken in the direction of this question, showing that the system would be usable for this purpose. Sections 8.3.5 and 8.3.6 presented a brief and simple analysis of identifying single participant motion profiles, which could identify how a participant normally performs a given motion. When asked to repeat this motion later, this system was shown to be able to pick up anomalies between two motions and two different participants. In addition to this, the quality of data produced by the system as is shown in Section 8.3.3 could be sufficient to conduct further testing and provide a general understanding of an athlete's current status.

This system is however extensible, allowing for much more effective motion analysis tools to be used to characterise a user's motion beyond elbow and wrist rotation angles. While not able to do this by the system on its own, the system also showed that it could be used with other applications that could be used to do this analysis in real-time.

Can this device interface with other commercial devices and protocols?

This question is addressed by the results from Section 8.1.2.2, where it was found that the system could be extended using a serialization framework. This allows other system to much more easily decipher data sent from the Action Tracer system. As a result of using a regular Wi-Fi network, communication with other devices is very simple and with the API available for use, custom applications may be written.

With regard to commercial protocols, the system in its current state does not support anything other than CSV files that contain the data as was recorded from the IMUs directly.

Will this model be sufficient for all sports, or would it be better suited to specific categories of activities?

It must be noted that this system was only tested for low speed and precise motions and further testing would need to be done for higher speed motions, leading to a limit in the activities this system may be limited to at this stage. Results from Sections 8.1.2.3 and 8.1.2.2 show that there is space for additional processing to do sport specific alterations to the data being collected.

Sections 8.3.3 - 8.3.6's results additionally showed that the data could be interpreted in various ways depending on the needs of the sport or activity, be it comparing by way of general profiles or of single moment actions and single motions.

9.2.2.1 Conclusion

The overall functionality of the Action Tracer system was tested and showed that it is indeed possible to design and build a motion capture system to address various problems that are faced both in the field of motion capture as well as in training and rehabilitation. When it came to answering the research questions, the work done in this dissertation to address these questions was sufficient to provide an initial

answer. These answers were based on the fact that a deeper level of testing was required to confirm the plausibility of these questions however the initial answers given were pointing in the direction of Action Tracer being able to solve the issues. These conclusions were all supported by various results from Chapter 8.

9.3 Future Work

This section will present some potential areas of development that were not covered in this dissertation that could be explored in future. These items were either out of the scope of this project or discovered during the assessment of the presented Action Tracer system.

9.3.1 True wireless motion capture

The comments received from participants in Section 8.3.1 helped to show that the system could in future be improved by making each node wireless. This would aid in reducing the restrictions of the cables with the disadvantage of increasing the cost of the system depending on the single node processing tools.

A wireless system would also be more preferable, as each IMU node would be capable of issuing interrupts which can be monitored. Extending the Action Tracer API to support both wired and wireless nodes may also have its advantages with regard to cost and processing.

9.3.2 Further benchmarking

Due to budget limitations, only MediaPipe could be used as a benchmark for the Action Tracer system. This limits the level of assessment that could be done between systems. Were a larger budget available, various other systems could be assessed alongside the designed system and would contribute to the results obtained in Section 8.3.

Ideally, a marker based system would be chosen to do this benchmarking so that data may be more deeply analysed. These systems would preferably be open source to allow a more direct comparison, such as OpenSense RT. [25]

9.3.3 Integration of AI

An additional field which the Action Tracer system could be used in would be Artificial Intelligence (AI). Due to the processing power and connectivity of the Raspberry Pi Zero W, it would be possible to integrate this system with various preprocessing stages done either on board the Pi or in the cloud, allowing for better data insights.

In addition to being used on the Pi, the data being collected by monitoring devices may be used to analyse a single athlete's data using multiple methods in order to get a more informed view of the motion. This may be empowered by AI techniques that are able to detect specific sequences in quick succession or deviations in motion profiles.

Having built and tested the Action Tracer system, the hope is for this system to be used in spaces where it would greatly benefit training and rehabilitation. With the relative ease in procuring the components required to build the system, the hope is to see new features added to the main repository, found at `https://github.com/chirambaht/Action-Tracer`.

Bibliography

[1] E. Hanani, "The study on value of recreational sports activity of urban communities," *KEMAS: Jurnal Kesehatan Masyarakat*, vol. 12, no. 2, pp. 286–291, 2017.

[2] A. Kos and A. Umek, "Wearable sensor devices for prevention and rehabilitation in healthcare: Swimming exercise with real-time therapist feedback," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1331–1341, 2019.

[3] R. W. Willy, "Innovations and pitfalls in the use of wearable devices in the prevention and rehabilitation of running related injuries," *Physical Therapy in Sport*, vol. 29, pp. 26–33, 2018.

[4] N. Hing, P. Vitartas, M. Lamont, and M. Amont, "International Gambling Studies Gambling sponsorship of sport: an exploratory study of links with gambling attitudes and intentions Gambling sponsorship of sport: an exploratory study of links with gambling attitudes and intentions," *International Gambling Studies*, vol. 19, no. 3, pp. 281–301, 2013.

[5] D. S. Teyhen, S. W. Shaffer, C. L. Lorenson, J. P. Halfpap, D. F. Donofry, M. J. Walker, J. L. Dugan, and J. D. Childs, "The functional movement screen: A reliability study," *Journal of Orthopaedic and Sports Physical Therapy*, vol. 42, no. 6, pp. 530–540, 2012.

[6] C. Armitano-Lago, D. Willoughby, and A. W. Kiefer, "A SWOT Analysis of Portable and Low-Cost Markerless Motion Capture Systems to Assess Lower-Limb Musculoskeletal Kinematics in Sport," *Frontiers in Sports and Active Living*, vol. 3, no. 1, pp. 1–14, 2022.

[7] R. Gruetzemacher, A. Gupta, G. B. Wilkerson, R. . Gruetzemacher, and A. . Gupta, "Sports Injury Prevention Screen (SIPS): Design and Architecture of an Internet of Things (IoT) Based Analytics Health App," in *International Conference on Information Resources Management*, p. 10, 2016.

[8] P. Nogueira, "Motion Capture Fundamentals A Critical and Comparative Analysis on Real-World Applications," in *Doctoral Symposium in Informatics Engineering*, (Porto), pp. 303–324, University of Porto, 2011.

[9] I. Takeda, A. Yamada, and H. Onodera, "Artificial Intelligence-Assisted motion capture for medical applications: a comparative study between markerless and passive marker motion capture," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 24, no. 8, pp. 864–873, 2020.

[10] E. Ceseracciu, Z. Sawacha, and C. Cobelli, "Comparison of Markerless and Marker-Based Motion Capture Technologies through Simultaneous Data Collection during Gait: Proof of Concept," *PLoS ONE*, vol. 9, no. 3, p. 9, 2014.

[11] L. Mündermann, S. Corazza, and T. P. Andriacchi, "The evolution of methods for the capture of human movement leading to markerless motion capture for biomechanical applications," *Journal of NeuroEngineering and Rehabilitation*, vol. 3, no. 6, 2006.

[12] W. Liu, Q. Bao, S. Yu, and T. Mei, "Recent Advances of Monocular 2D and 3D Human Pose Estimation: A Deep Learning Perspective," *Association for Computing Machinery*, 2022.

[13] S. Shimada, V. Golyanik, W. Xu, P. Perez, and C. Theobalt, "Neural Monocular 3D Human Motion Capture with Physical Awareness," *ACM Trans. Graph*, vol. 40, no. 4, 2021.

[14] R. Mohedano and N. Garcia, "Robust multi-camera 3D tracking from mono-camera 2d tracking using Bayesian Association," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 1, pp. 1–8, 2010.

[15] G. W. Jensen, P. van der Smagt, E. Heiss, H. Straka, and T. Kohl, "SnakeStrike: A Low-Cost Open-Source High-Speed Multi-Camera Motion Capture System," *Frontiers in Behavioral Neuroscience*, vol. 14, no. 116, p. 16, 2020.

[16] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *CVPR*, 2016.

[17] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," in *CVPR*, 2017.

[18] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand Keypoint Detection in Single Images using Multiview Bootstrapping," in *CVPR*, 2017.

[19] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[20] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann, "MediaPipe: A Framework for Building Perception Pipelines," *arXiv e-prints*, p. arXiv:1906.08172, 2019.

[21] J. L. J. Bascones, "Cloud point labelling in optical motion capture systems," 2019.

[22] T. H. Ribeiro, "Motion capture technology-benefits and challenges," *International Journal of Innovative Research in Technology & Science(IJIRTS)*, vol. 4, no. 1, pp. 48–51, 2016.

[23] M. Furniss, "Motion Capture," in *Media in Transition*, (Cambridge), MIT, 1999.

[24] B. R. Hindle, J. W. Keogh, and A. V. Lorimer, "Inertial-Based Human Motion Capture: A Technical Summary of Current Processing Methodologies for Spatiotemporal and Kinematic Measures," *Applied Bionics and Biomechanics*, vol. 2021, no. 6628320, p. 14, 2021.

[25] P. Slade, A. Habib, J. L. Hicks, and S. L. Delp, "An open-source and wearable system for measuring 3D human motion in real-time," *IEEE Transactions on Biomedical Engineering*, vol. XX, no. Xx, pp. 1–9, 2021.

[26] F. Gravenhorst, A. Muaremi, F. Kottmann, G. Troster, R. Sigrist, N. Gerig, and C. Draper, "Strap and row: Rowing technique analysis based on inertial measurement units implemented in mobile phones," *IEEE ISSNIP 2014 - 2014 IEEE 9th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Conference Proceedings*, no. 4, pp. 21–24, 2014.

[27] D. Holden, "Robust Solving of Optical Motion Capture Data by Denoising," *ACM Transactions on Graphics*, vol. 37, no. 4, 2018.

[28] A. Aristidou and J. Lasenby, "Real-time marker prediction and CoR estimation in optical motion capture," *Visual Computer*, vol. 29, no. 1, pp. 7–26, 2013.

[29] I. Stancic, T. G. Supuk, and A. Panjkota, "Design, development and evaluation of optical motion-tracking system based on active white light markers," *IET Science, Measurement and Technology*, vol. 7, no. 4, pp. 206–214, 2013.

[30] F. H. Raab, E. B. Blood, T. O. Steiner, and H. R. Jones, "Magnetic Position and Orientation Tracking System," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-15, no. 5, pp. 709–718, 1979.

[31] S. Hashi, M. Toyoda, S. Yabukami, K. Ishiyama, Y. Okazaki, and K. I. Arai, "Wireless Magnetic Motion Capture System for Multi-Marker Detection," *IEEE TRANSACTIONS ON MAGNETICS*, vol. 42, no. 10, p. 3279, 2006.

[32] J. F. O'Brien, J. B. Bodenheimer, Robert E., G. J. Brostow, and J. K. Hodgins, "Automatic Joint Parameter Estimation from Magnetic Motion Capture Data," tech. rep., Georgia Institute of Technology, 1999.

[33] C. Oz and M. C. Leu, "American Sign Language word recognition with a sensory glove using artificial neural networks," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 7, pp. 1204–1213, 2011.

[34] D. Rajendran, R. Da, V. Torres, Z. Hu, R. Ramalingame, A. Al-Hamry, and O. Kanoun, "Electronic Motion Capture Glove based on Highly Sensitive Nanocomposite Sensors; Electronic Motion Capture Glove based on Highly Sensitive Nanocomposite Sensors," in *2019 16th International Multi-Conference on Systems, Signals & Devices (SSD)*, pp. 494–497, 2019.

[35] A. Das, L. Yadav, M. Singhal, R. Sachan, H. Goyal, K. Taparia, R. Gulati, A. Singh, and G. Trivedi, "Smart glove for Sign Language communications; Smart glove for Sign Language communications," in *2016 International Conference on Accessibility to Digital World (ICADW)*, IEEE, 2016.

[36] F. J. Perez-Duarte, M. Lucas-Hernandez, A. Matos-Azevedo, J. A. Sanchez-Margallo, I. Diaz-Gü, and F. M. Sanchez-Margallo, "Objective analysis of surgeons' ergonomy during laparoendoscopic single-site surgery through the use of surface electromyography and a motion capture data glove," *Surgical Endoscopy*, vol. 28, pp. 1314–1320, 2014.

[37] G. N. Pradhan, N. Engineer, M. Nadin, and B. Prabhakaran, "Integration of motion capture and EMG data for classifying the human motions," in *Proceedings - International Conference on Data Engineering*, pp. 56–63, 2007.

[38] H. W. Ng, X. Jiang, L. K. Merhi, and C. Menon, "Investigation of the feasibility of strain gages as pressure sensors for force myography," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10208 LNCS, pp. 261–270, 2017.

[39] E. Cho, R. Chen, L. K. Merhi, Z. Xiao, B. Pousett, and C. Menon, "Force myography to control robotic upper extremity prostheses: A feasibility study," *Frontiers in Bioengineering and Biotechnology*, vol. 4, no. MAR, p. 18, 2016.

[40] Z. G. Xiao, A. M. Elnady, and C. Menon, "Control an exoskeleton for forearm rotation using FMG," in *Proceedings of the IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics*, pp. 591–596, IEEE Computer Society, 2014.

[41] N. Praveen and N. Karanth, "Sign Language Interpreter Using A Smart Glove," in *2014 International Conference on Advances in Electronics Computers and Communications*, 2014.

[42] G. Merino, L. da Silva, D. Mattos, B. Guimarães, and E. Merino, "Ergonomic evaluation of the musculoskeletal risks in a banana harvesting activity through qualitative and quantitative measures, with emphasis on motion capture (Xsens) and EMG," *International Journal of Industrial Ergonomics*, vol. 69, no. 2019, pp. 80–89, 2019.

[43] B. Fang, D. Guo, F. Sun, H. Liu, and Y. Wu, "A robotic hand-arm teleoperation system using human arm/hand with a novel data glove," in *IEEE Conference on Robotics and Biomimetics*, pp. 2483–2488, 2015.

[44] S. D. Senturia, *Microsystem design*. Kluwer Academic Publishers, 1 ed., 2001.

[45] TE Connectrivity, "Choosing the right type of accelerometer," tech. rep., TE Connectivity, Schaffhausen, 2017.

[46] G. Plasqui, A. G. Bonomi, and K. R. Westerterp, "Daily physical activity assessment with accelerometers: new insights and validation studies," *Obesity Reviews*, vol. 14, no. 6, pp. 451–462, 2013.

[47] K. Liu, W. Zhang, W. Chen, K. Li, F. Dai, F. Cui, X. Wu, G. Ma, and Q. Xiao, "The development of micro-gyroscope technology," *Journal of Micromechanics and Microengineering*, vol. 19, no. 11, p. 113001, 2009.

[48] D. Xia, C. Yu, and L. Kong, "The Development of Micromachined Gyroscope Structure and Circuitry Technology," *Sensors*, vol. 14, no. 1, pp. 1394–1473, 2014.

[49] E. A. Donley, "Nuclear magnetic resonance gyroscopes," in *Proceedings of IEEE Sensors*, pp. 17–22, 2010.

[50] Z. You, "Chapter 9 - Magnetometer Technology," in *Space Microsystems and Micro/nano Satellites* (Z. You, ed.), Micro and Nano Technologies, pp. 341–360, Butterworth-Heinemann, 2018.

[51] C. Coillot, P. Leroy, and K. Kuang, "Induction magnetometers principle, modeling and ways of improvement," *Magnetic Sensors—Principles and Applications*, pp. 45–64, 2012.

[52] M. Li, S. Nitzan, and D. A. Horsley, "Frequency-modulated lorentz force magnetometer with enhanced sensitivity via mechanical amplification," *IEEE Electron Device Letters*, vol. 36, no. 1, pp. 62–64, 2015.

[53] D. Budker, D. F. Kimball, S. M. Rochester, V. V. Yashchuk, and M. Zolotorev, "Sensitive magnetometry based on nonlinear magneto-optical rotation," *Physical Review A - Atomic, Molecular, and Optical Physics*, vol. 62, no. 4, p. 7, 2000.

[54] J. Clarke, "Principles and Applications of SQUIDs," *Proceedings of the IEEE*, vol. 77, no. 8, pp. 1208–1223, 1989.

[55] Adafruit Industries and Mouser Electronics, "Adafruit LSM303AGR," 2021.

[56] A. Nurhakim, M. R. Effendi, H. M. Saputra, R. Mardiati, T. Priatna, and N. Ismail, "A Novel Approach to Calculating Yaw Angles Using an Accelerometer Sensor," in *Proceeding of 2019 5th International Conference on Wireless and Telematics, ICWT 2019*, Institute of Electrical and Electronics Engineers Inc., 2019.

[57] A. Bayat, M. Pomplun, and D. A. Tran, "A Study on Human Activity Recognition Using Accelerometer Data from Smartphones," *Procedia Computer Science*, vol. 34, no. 2014, pp. 450–457, 2014.

[58] C. Pao and CEVA, "The Optimal Number of Axes for Motion Sensors - CEVA's Experts blog." `https://www.ceva-dsp.com/ourblog/motion-sensors-how-many-axes-do-you-need/`, 2019.

[59] N. Ahmad, A. R. Ghazilla, N. M. Khairi, and V. Kasi, "Reviews on Various Inertial Measurement Unit (IMU) Sensor Applications," *International Journal of Signal Processing Systems*, vol. 1, no. 2, pp. 256–262, 2013.

[60] W. S. Flenniken IV, J. H. Wall, and D. M. Bevly, "Characterization of various IMU error sources and the effect on navigation performance," *Proceedings of the 18th International Technical Meeting of the Satellite Division of The Institute of Navigation, ION GNSS 2005*, vol. 2005, pp. 967–978, 2005.

[61] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm," tech. rep., University of Bristol, Bristol, 2011.

[62] N. Abbate, A. Basile, C. Brigante, A. Faulisi, and F. L. Rosa, "Modern Breakthrough Technologies Enable New Applications Based on IMU Systems," *Journal of Sensors*, vol. 2011, p. 7, 2011.

[63] A. Ghio, S. Escalante, and J. Tarrillo, "Analysis of Moving Average Filter for IMU Measurements on an 8-bit Microcontroller," in *Proceedings of the 2018 IEEE 25th International Conference on Electronics, Electrical Engineering and Computing, INTERCON 2018*, Institute of Electrical and Electronics Engineers Inc., 2018.

[64] J. Z. Sasiadek, "Sensor fusion," *Annual Reviews in Control*, vol. 26, no. 2, pp. 203–228, 2002.

[65] A. Kos and A. Umek, *Biomechanical Biofeedback Systems and Applications*. Human–Computer Interaction Series, Cham: Springer International Publishing, 1st ed. 20 ed., 2018.

[66] S. Ilovar, D. Zolger, E. Castrillon, J. Car, and K. Huckvale, "Biofeedback for treatment of awake and sleep bruxism in adults: Systematic review protocol," *Systematic Reviews*, vol. 3, no. 1, pp. 1–9, 2014.

[67] J. Z. Goldenberg, M. Brignall, M. Hamilton, J. Beardsley, R. D. Batson, J. Hawrelak, B. Lichtenstein, and B. C. Johnston, "Biofeedback for treatment of irritable bowel syndrome," *Cochrane Database of Systematic Reviews*, vol. 2019, no. 11, 2019.

[68] D. L. Frank, L. Khorshid, J. F. Kiffer, C. S. Moravec, and M. G. McKee, "Biofeedback in medicine: Who, when, why and how?," *Mental Health in Family Medicine*, vol. 7, no. 2, pp. 85–91, 2010.

[69] A. Hermanis, K. Nesenbergs, R. Cacurs, and M. Greitans, "Wearable Posture Monitoring System with Biofeedback via Smartphone," *Journal of Medical and Bioengineering*, vol. 2, no. 1, pp. 40–44, 2013.

[70] R. Li, Z. Cai, W. Lee, and D. T. Lai, "A wearable biofeedback control system based body area network for freestyle swimming," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, vol. 2016-Octob, pp. 1866–1869, 2016.

[71] P. P. Ray, "Internet of Things for Sports (IoTSport): An architectural framework for sports and recreational activity," *International Conference on Electrical, Electronics, Signals, Communication and Optimization, EESCO 2015*, 2015.

[72] W. Kong, S. Sessa, S. Cosentino, M. Zecca, K. Saito, C. Wang, U. Imtiaz, Z. Lin, L. Bartolomeo, H. Ishii, T. Ikai, and A. Takanishi, "Development of a real-time IMU-based motion capture system for gait rehabilitation," in *2013 IEEE International Conference on Robotics and Biomimetics, ROBIO 2013*, pp. 2100–2105, IEEE Computer Society, 2013.

[73] J. González-Alonso, D. Oviedo-Pastor, H. J. Aguado, F. J. Díaz-Pernas, D. González-Ortega, and M. Martínez-Zarzuela, "Custom imu-based wearable system for robust 2.4 ghz wireless human body parts orientation tracking and 3d movement visualization on an avatar," *Sensors*, vol. 21, no. 19, 2021.

[74] L. Buthe, U. Blanke, H. Capkevics, and G. Troster, "A wearable sensing system for timing analysis in tennis," in *BSN 2016 - 13th Annual Body Sensor Networks Conference*, pp. 43–48, Institute of Electrical and Electronics Engineers Inc., 2016.

[75] N. Abbate, A. Basile, C. Brigante, and A. Faulisi, "Development of a MEMS based wearable motion capture system," in *Proceedings - 2009 2nd Conference on Human System Interactions, HSI '09*, (Catania), pp. 255–259, 2009.

[76] E. Wade and M. J. Matarić, "Design and testing of lightweight inexpensive motion-capture devices with application to clinical gait analysis," in *2009 3rd International Conference on Pervasive Computing Technologies for Healthcare - Pervasive Health 2009, PCTHealth 2009*, (Los Angeles), 2009.

[77] Q. Jin, Z. Zhang, and W. Jin, "Design and implementation of the IMU human body motion tracking system," *Advances in Intelligent Systems and Computing*, vol. 1037, no. 2022, pp. 1034–1043, 2020.

[78] A. Seth, J. L. Hicks, T. K. Uchida, A. Habib, C. L. Dembia, J. J. Dunne, C. F. Ong, M. S. DeMers, A. Rajagopal, M. Millard, S. R. Hamner, E. M. Arnold, J. R. Yong, S. K. Lakshmikanth, M. A. Sherman, J. P. Ku, and S. L. Delp, "OpenSim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement," *PLOS Computational Biology*, vol. 14, no. 7, p. e1006223, 2018.

[79] Xsens Technologies B.V, "MTi 1-series Datasheet," 2019.

[80] W. Wei, K. Kurita, J. Kuang, and A. Gao, "Real-Time Limb Motion Tracking with a Single IMU Sensor for Physical Therapy Exercises," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, pp. 7152–7157, Institute of Electrical and Electronics Engineers Inc., 2021.

[81] ANT, "ANT Message Protocol and Usage," 2013.

[82] M. Sama, V. Pacella, E. Farella, L. Benini, and B. Ricco, *3dID: a low-power, low-cost hand motion capture device*. Paper, University of Bologna, 2008.

[83] J. Stylos, *Making APIs more usable with improved API Designs, Documentation and Tools*. Phd thesis, Carnegie Mellon University, 2009.

[84] Microsoft, "Programming reference for the Win32 API - Win32 apps | Microsoft Learn." `https://learn.microsoft.com/en-us/windows/win32/api/`, 1984.

[85] J. Shippen and B. May, "BoB – Biomechanics in MATLAB," in *Proceedings of 11th International Conference BIOMDLORE 2016*, vol. 2016-Octob, (DRUSKININKAI), pp. 11–13, 2016.

[86] P. Raghavendra, M. Sachin, P. S. Srinivas, and V. Talasila, "Design and Development of a Real-Time, Low-Cost IMU Based Human Motion Capture System," *Lecture Notes in Networks and Systems*, vol. 12, no. 5, pp. 155–165, 2017.

[87] A. Kos and A. Umek, "Selection of sensor networks for biofeedback systems in healthcare and sports," in *7th International Conference on Information Society and Technology ICIST 2017 Selection*, vol. 1, pp. 312–315, 2017.

[88] B. Boehm, "A Spiral model of software development and enhancement," *Software Management, Seventh Edition*, vol. 21, no. 5, pp. 37–48, 1988.

[89] J. A. N. Bartoníček, "Programming Language Paradigms & The Main Principles of Object-Oriented Programming," *CRIS - Bulletin of the Centre for Research and Interdisciplinary Study*, vol. 2014, no. 1, pp. 93–99, 2014.

[90] Rokoko, "Intuitive and affordable motion capture tools for character animation." https://www.rokoko.com/, 2022.

[91] InvenSense, "MPU-6000 and MPU-6050 Product Specification," Tech. Rep. 4, TDK, California, 2013.

[92] Raspberry Pi Foundation, "Raspberry Pi Zero W." https://www.raspberrypi.com/products/raspberry-pi-zero-w/, 2017.

[93] K. Varda, "Protocol Buffers: Google's Data Interchange Format," tech. rep., Google, 2008.

[94] J. Rowberg, "I2C Device Library." http://www.i2cdevlib.com/, 2010.

[95] E. Guimarães, "MPU6050 Motion Tracking." https://github.com/Edubgr/MPU6050-MotionTracking, 2021.

Appendix A

Additional documents

ETHICS APPLICATION FORM

Please Note:
Any person planning to undertake research in the Faculty of Engineering and the Built Environment (EBE) at the University of Cape Town is required to complete this form **before** collecting or analysing data. The objective of submitting this application *prior* to embarking on research is to ensure that the highest ethical standards in research, conducted under the auspices of the EBE Faculty, are met. Please ensure that you have read, and understood the **EBE Ethics in Research Handbook** (available from the UCT EBE, Research Ethics website) prior to completing this application form: http://www.ebe.uct.ac.za/ebe/research/ethics1

APPLICANT'S DETAILS		
Name of principal researcher, student or external applicant		Humphrey Chiramba
Department		Electrical Engineering
Preferred email address of applicant:		chirambaht@gmail.com
If Student	Your Degree: e.g., MSc, PhD, etc.	MSc Electrical Engineering
	Credit Value of Research: e.g., 60/120/180/360 etc.	180
	Name of Supervisor (if supervised):	Dr Simon Winberg
If this is a researchcontract, indicate the source of funding/sponsorship		
Project Title		Action Tracer with an Android API

I hereby undertake to carry out my research in such a way that:

- there is no apparent legal objection to the nature or the method of research; and
- the research will not compromise staff or students or the other responsibilities of the University;
- the stated objective will be achieved, and the findings will have a high degree of validity;
- limitations and alternative interpretations will be considered;
- the findings could be subject to peer review and publicly available; and
- I will comply with the conventions of copyright and avoid any practice that would constitute plagiarism.

APPLICATION BY	Full name	Signature	Date
Principal Researcher/ Student/External applicant	Humphrey Chiramba	Signed by candidate	22/05/2021
SUPPORTED BY	Full name	Signature	Date
Supervisor (where applicable)	Simon Winberg		22/05/2021

APPROVED BY	Full name	Signature	Date
HOD (or delegated nominee) Final authority for all applicants who have answered NO to all questions in Section 1; and for all Undergraduate research (Including Honours).	Amit Mishra		24-6-21
Chair: Faculty EIR Committee For applicants other than undergraduate students who have answered YES to any of the questions in Section 1.			

Information Sheet

Project Title: Action Tracer with Android API
Researcher: Humphrey Chiramba

Details of the study

The reason that data is required for this study is to verify the designed system's functionality. This system will work as follows:

Sensors will be attached to your body in a few locations. They will be firmly secured to the body to capture either the rotation of the body, to capture a pulse or to capture a muscle contraction. This data will simply be captured to be compared to a more commercial data capture device and compared to verify that the collected data is both accurate and precise.

The final goal of the study is to create a low cost and open source system that could potentially be used in training athletes in a specific discipline. There is a large gap in the market for low cost yet intuitive data capture devices in this field, and the results of this study may lead to a system that could be used in training the next generation of athletes or in rehabilitation centres to help patients return to full or partial mobility sooner and more accurately.

Expectations

The following are the expectations for you as a participant in this study, which will be done in stages. Each of these stages will not be strenuous or tiring, but will be extremely repetitive with minor changes. Please note that these stages will be time sensitive, meaning that you will perform actions for a period of time and not to achieve a specific goal. They can be broken down into 5 stages:

1. **Blind performance**

   You will be required to perform some specific actions without the devices attached to your body. You will need to remember some of the actions you performed and how you performed them while being filmed.

2. **Verification**

   The sensors will be strapped on to your body. The researcher will then ask you to perform a few actions to verify that the capturing device is getting your data correctly and that the devices have all been correctly attached.

3. **Observed performance**

   You will repeat the exact procedure that you performed in the blind performance stage with the sensors attached.

Consent Form

Project Title: Action Tracer with Android API
Researcher: Humphrey Chiramba

I volunteer to participate in a research project conducted by **Humphrey Chiramba** from the University of Cape Town. I understand that this research project is designed to study a method to observe the motion and interaction of the human body while performing a specific action in a given sporting activity. Furthermore, I also understand that my data will only be used to verify the functionality of the system. I also understand that I will have sensors attached to my body for the purposes of this study and will have to perform a few actions and motions with and without them attached.

As a _____ at the University of Cape Town, I understand that I am being invited to take part in an interview and practical exercise. I understand that in agreeing to participate:

- My participation is voluntary. I understand that I will not be paid for my participation.
- The interview will last approximately 30 minutes. Notes will be written during the interview by the researcher and my data will be recorded.
- I understand that if I feel uncomfortable in any way during the interview, I have the right to decline to perform any of the required actions and leave without giving explanation, as well as have all my data removed from the study.
- I understand that the researcher will not identify me by name in any reports using the information obtained from the interview. My confidentiality as a participant will remain secure. Subsequent uses of recordings and data will be subject to standard data use policies, which protect anonymity of individuals and institutions.
- Administrative and other teaching staff at the University of Cape Town may be present during the interview, but will not have access to raw notes or transcripts of either the survey or the interview. This precaution will prevent any of the findings having personal negative repercussions for me.
- If I choose to be interviewed, I have the right to view and comment on the recorded interview data before the findings are analysed.
- I have read and understand the participant information sheet provided to me. I have had all my questions answered to my satisfaction, and I voluntarily agree to participate in this study.
- I have been given a copy of this consent form.

I hereby agree / disagree (circle the applicable option) to participate in the **interview** for this study.
I hereby agree / disagree (circle the applicable option) to the **data recording of my interview** for this study.

Name _____

Signed _____

Date _____

Appendix B

Design and Implementation

B.1 Software: Accessing Action Tracer's source code

Table B.1: An overview of the accelerometer FSRs from all the systems, presented in Table 2.1

Accelerometer	**1g**	**2g**	**4g**	**5g**	**6g**	**8g**	**16g**
ISM330DHCX		✓	✓			✓	✓
BNO080						✓	
EXLs3		✓	✓			✓	✓
MPU-9250		✓	✓			✓	✓
LIS331DLH		✓	✓			✓	
LIS3LV02DL		✓			✓		
MXR9150				✓			

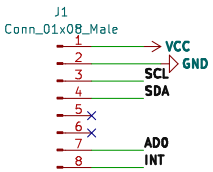
Table B.2: An overview of the gyroscope FSRs from all the systems, presented in Table 2.1

Gyroscope	**125dps**	**250dps**	**300dps**	**400dps**	**500dps**	**1000dps**	**2000dps**
ISM330DHCX	✓	✓			✓	✓	✓
BNO080							✓
EXLs3		✓			✓	✓	✓
MPU-9250		✓			✓	✓	✓
LYPR540AH				✓			
LY530AL			✓				
ADXRS300							✓

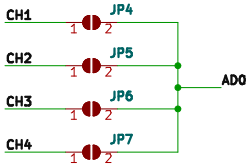
B.2 Hardware

B.2.1 Schematics

MPU6050

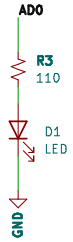


Channel Selector



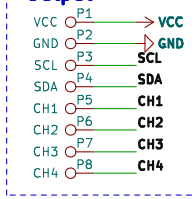
Power routing

LED Indicator

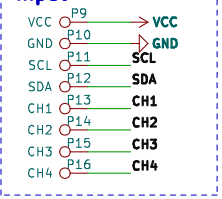


I/O

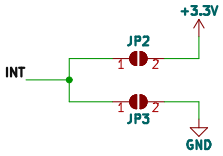
Output



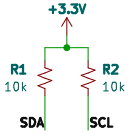
Input



Interrupt Manager



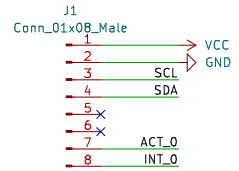
I2C Pull-Up



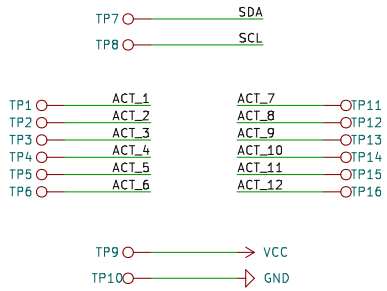
The Action Node accompanies the Action Tracer project that aims to capture motion on a Raspberry Pi and transmit that data to a server.

Sheet: /		
File: ActionPCB.kicad_sch		
Title: Action Node		
Size: A4	Date: 5 March 2022	**Rev:** 0.3
KiCad E.D.A. kicad 6.0.2+dfsg-1		Id: 1/1

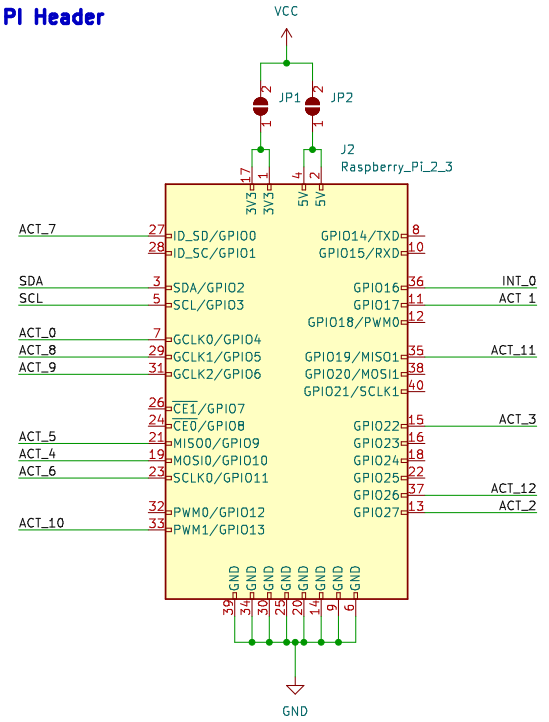
MPU6050 Board



Connector Pads



PI Header



The Action Main Node accompanies the Action Tracer project that aims to capture motion on a Raspberry Pi and transmit that data to a server. This board connects the Pi to the nodes.

Sheet: /		
File: ConnectorPCB.kicad_sch		
Title: Action Main Node		
Size: A4	Date: 5 March 2022	Rev: 0.1
KiCad E.D.A. eeschema 6.0.2+dfsg-1		Id: 1/1

B.3 Test circuitry

B.4 Machine Specifications

B.4.1 Raspberry Pi Zero W

OS Raspbian GNU/Linux

Kernel 5.10.17+

CPU BCM2835 (1) @ 1GHz

Memory 430 MiB

B.4.2 Machine 1

OS Ubuntu 22.04.1 LTS x86_64

Kernel 5.15.0-53-generic

CPU Intel i7-4790 (8) @ 4.000GHz

GPU NVIDIA GeForce GT 730

Memory 15841 MiB

B.4.3 Machine 2

OS Windows 10 Home

CPU Intel i3-7020U (4) @ 2.3GHz

GPU Intel HD Graphics 620

Memory 7430 MiB

B.4.4 Machine 3

OS EMUI 9.0

Chipset Kirin 659

CPU Cortex-A53 (8) @ 1.7GHz and 2.36GHz

GPU Mali-T830 MP2

Memory 4 GB

B.5 Software: Accessing Action Tracer's source code

The source code used to develop Action Tracer is split into many online repositories due to the various use cases that exist. Below is a list of the core code and design repositories:

Main Drivers Used to manage network connections and data collection from IMUs connected to a Raspberry Pi: GitHub – Action-Tracer

Action Tracer Documentation The main software documentation of the Action Tracer system will be found at: GitHub Pages: Action Tracer

Unity implementaion Used as a client to a host Pi to test the functionality of the Action Tracer system on an android phone or desktop: GitHub – Action Figure

B.5.1 Sample Python code to find angles between two vectors

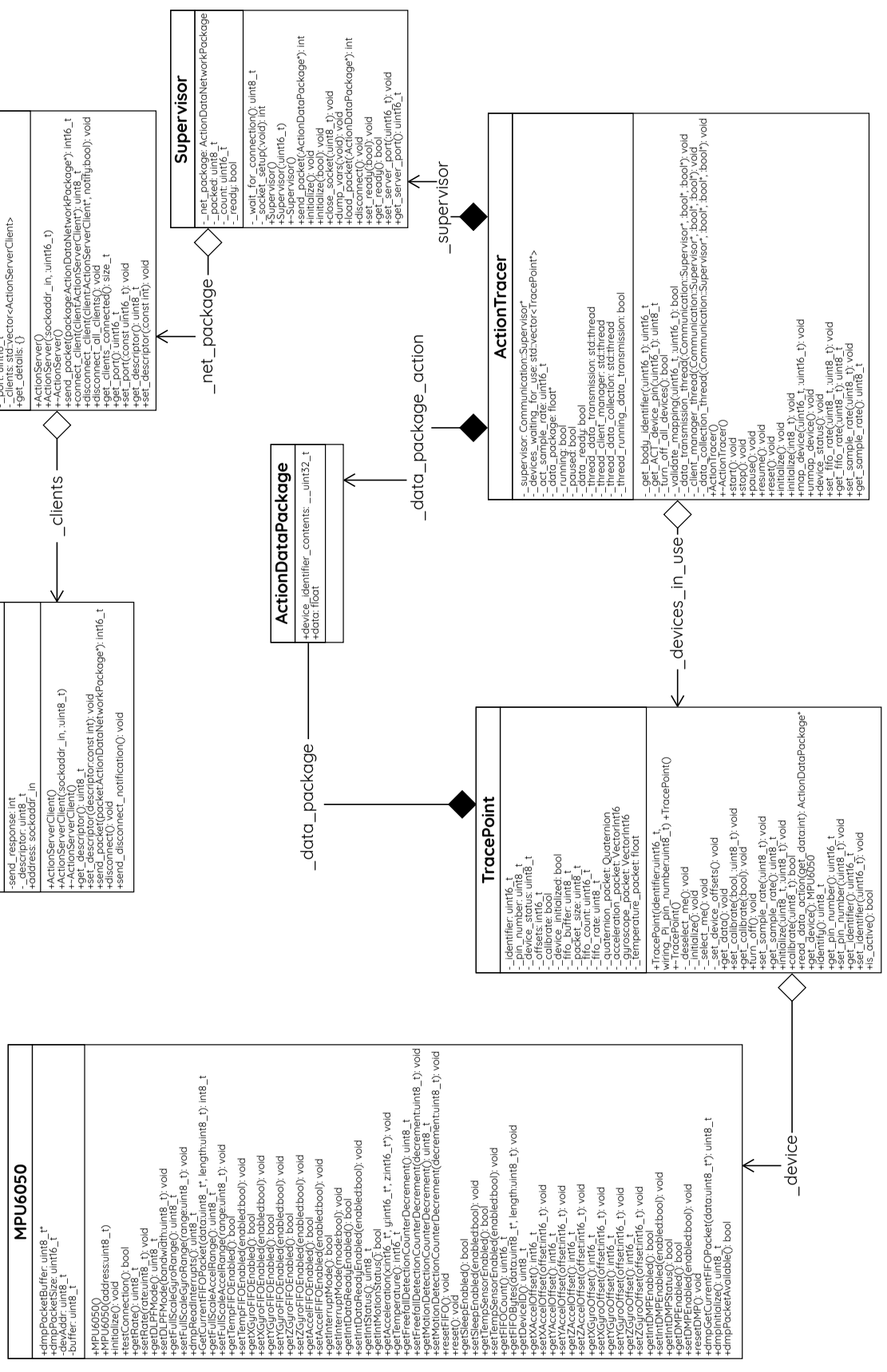
```
import numpy as np

def calculate_angle_2d(a, b, c):
# Each point: [ x , y ]

angle = np.arctan2(c[1] - b[1], c[0] - b[0]) - np.arctan2(a[1] - b[1], a[0] -
   b[0])

return np.degrees(angle).round(3)
```

Listing B.1: A python method used to obtain an angle between 3 points in degrees each with x and y data

```
import numpy as np

def calculate_angle_3d(a, b, c):
# Each point: [ x , y , z ]

ba = a - b
bc = c - b

cos_angle = np.dot(ba, bc) / (np.linalg.norm(ba) * np.linalg.norm(bc))
angle = np.arccos(cos_angle)

return np.degrees(angle).round(3)
```

Listing B.2: A python method used to obtain an angle between 3 points in degrees each with x, y and z data

B.5.2 UML Class Diagram


ActionServer
_descriptor: uint8_t
_client_count: int
_clients: std::vector<ActionServerClient>
+get_details: {}
+ActionServer()
+ActionServer(sockaddr_in, uint16_t)
+~ActionServer()
+send_packet(package:ActionDataNetworkPackage*): int16_t
+connect_client(client:ActionServerClient*, uint8_t
+disconnect_client(client:ActionServerClient*, notify:bool): void
+disconnect_all_clients(): void
+get_clients_connected(): size_t
+get_port(): uint16_t
+set_port(const uint16_t): void
+get_descriptor(): uint8_t
+set_descriptor(const int): void
Supervisor
_net_package: ActionDataNetworkPackage
_packed: uint8_t
_count: uint16_t
_ready: bool
-_wait_for_connection(): uint8_t
-_socket_setup(void): int
+Supervisor()
+Supervisor(uint16_t)
+~Supervisor()
+send_packet(package:ActionDataPackage*): int
+initialize(): void
+initialize(bool): void
+close_socket(uint8_t): void
+dump_vars(void): void
+load_packet(ActionDataPackage*): int
+disconnect(): void
+get_ready(): bool
+set_server_port(uint16_t): void
+get_server_port(): uint16_t
ActionServerClient
-send_response: int
-_descriptor: uint8_t
+address: sockaddr_in
+ActionServerClient()
+ActionServerClient(sockaddr_in, uint8_t)
+~ActionServerClient()
+get_descriptor(): uint8_t
+set_descriptor(descriptor:const int): void
+send_packet(package:ActionDataNetworkPackage*): int16_t
+disconnect(): void
+send_disconnect_notification(): void
ActionDataPackage
+device_identifier_contents: __uint32_t
+data: float
ActionTracer
-_supervisor: Communication::Supervisor*
-_devices_waiting_for_use: std::vector<TracePoint*>
-_act_sample_rate: uint16_t
-_data_package: float*
-_running: bool
-_paused: bool
-_data_ready: bool
-_thread_data_transmission: std::thread
-_thread_client_manager: std::thread
-_thread_data_collection: std::thread
-_thread_running_data_transmission: bool
-_get_body_identifier(uint16_t): uint16_t
-_get_ACT_device_pin(uint16_t): uint8_t
-_turn_off_all_devices(): bool
-_validate_mapping(uint16_t, uint16_t): bool
-_data_transmission_thread(Communication::Supervisor*, bool*, bool*): void
-_client_manager_thread(Communication::Supervisor*, bool*, bool*): void
-_data_collection_thread(Communication::Supervisor*, bool*, bool*): void
+ActionTracer()
+~ActionTracer()
+start(): void
+stop(): void
+pause(): void
+resume(): void
+reset(): void
+initialize(): void
+initialize(int8_t): void
+map_device(uint16_t, uint16_t): void
+unmap_device(): void
+device_status(): void
+set_fifo_rate(uint8_t, uint8_t): void
+get_fifo_rate(uint8_t): uint8_t
+set_sample_rate(uint8_t): void
+get_sample_rate(): uint8_t
TracePoint
-identifier: uint16_t
-pin_number: uint8_t
-device_status: uint8_t
-offsets: int16_t
-calibrated: bool
-device_initialized: bool
-fifo_buffer: uint8_t
-packet_size: uint8_t
-fifo_count: uint16_t
-fifo_rate: uint8_t
-quaternion_packet: Quaternion
-acceleration_packet: VectorInt16
-gyroscope_packet: VectorInt16
-temperature_packet: float
+TracePoint(identifier:uint16_t,
wiring_PI_pin_number:uint8_t) +TracePoint()
+~TracePoint()
-_deselect_me(): void
-_initialize(): void
-_select_me(): void
-_set_device_offsets(): void
+get_data(): void
+set_calibrate(bool, uint8_t): void
+get_calibrate():bool): void
+turn_off(): void
+set_sample_rate(uint8_t): void
+get_sample_rate(): uint8_t
+initialize(uint8_t): void
+calibrate(uint8_t): bool
+read_data_action(get_dataint): ActionDataPackage*
+get_device(): MPU6050
+identify(): uint8_t
+get_pin_number(): uint16_t
+set_pin_number(uint8_t): void
+get_identifier(): uint16_t
+set_identifier(uint16_t): void
+is_active(): bool
MPU6050
+dmpPacketBuffer: uint8_t*
+dmpPacketSize: uint16_t
-devAddr: uint8_t
-buffer: uint8_t
+MPU6050()
+MPU6050(address:uint8_t)
+initialize(): void
+testConnection(): bool
+getRate(): uint8_t
+setRate(rate:uint8_t): void
+getDLPFMode(): uint8_t
+setDLPFMode(bandwidth:uint8_t): void
+getFullScaleGyroRange(): uint8_t
+setFullScaleGyroRange(range:uint8_t): void
+GetCurrentFIFOPacket(data:uint8_t*, length:uint8_t): int8_t
+getFullScaleAccelRange(): uint8_t
+setFullScaleAccelRange(range:uint8_t): void
+getTempFIFOEnabled(): bool
+setTempFIFOEnabled(enabled:bool): void
+getXGyroFIFOEnabled(): bool
+setXGyroFIFOEnabled(enabled:bool): void
+getYGyroFIFOEnabled(): bool
+setYGyroFIFOEnabled(enabled:bool): void
+getZGyroFIFOEnabled(): bool
+setZGyroFIFOEnabled(enabled:bool): void
+getAccelFIFOEnabled(): bool
+setAccelFIFOEnabled(enabled:bool): void
+getInterruptMode(): bool
+setInterruptMode(mode:bool): void
+getIntDataReadyEnabled(): bool
+setIntDataReadyEnabled(enabled:bool): void
+getIntMotionStatus(): bool
+getAcceleration(xint16_t*, yint16_t*, zint16_t*): void
+getTemperature(): int16_t
+getFreefallDetectionCounterDecrement(): uint8_t
+setFreefallDetectionCounterDecrement(decrement:uint8_t): void
+getMotionDetectionCounterDecrement(): uint8_t
+setMotionDetectionCounterDecrement(decrement:uint8_t): void
+resetFIFO(): void
+reset(): void
+getSleepEnabled(): bool
+setSleepEnabled(enabled:bool): void
+getTempSensorEnabled(): bool
+setTempSensorEnabled(enabled:bool): void
+getFIFOCount(): uint16_t
+getFIFOBytes(data:uint8_t*, length:uint8_t): void
+getDeviceID(): uint8_t
+getXAccelOffset(): int16_t
+setXAccelOffset(offset:int16_t): void
+getYAccelOffset(): int16_t
+setYAccelOffset(offset:int16_t): void
+getZAccelOffset(): int16_t
+setZAccelOffset(offset:int16_t): void
+getXGyroOffset(): int16_t
+setXGyroOffset(offset:int16_t): void
+getYGyroOffset(): int16_t
+setYGyroOffset(offset:int16_t): void
+getZGyroOffset(): int16_t
+setZGyroOffset(offset:int16_t): void
+getIntDMPEnabled(): bool
+setIntDMPEnabled(enabled:bool): void
+getIntDMPStatus(): bool
+getDMPEnabled(): bool
+setDMPEnabled(enabled:bool): void
+resetDMP(): void
+dmpGetCurrentFIFOPacket(data:uint8_t*): uint8_t
+dmpInitialize(): uint8_t
+dmpPacketAvailable(): bool
_clients
_net_package
_data_package_action
_supervisor
_data_package
_devices_in_use
_device