Individual Project
Final Report

ID Number: B914293

Programme: BEng Sports Technology

Module Code: 23WSC500

Project Title: A Cost Effective Solution to Wheelchair Basketball Mobility Performance Monitoring

Abstract:

As interest and professionalism in wheelchair sports grow, so does the necessity to improve team dynamics and individual athlete performance. For that, tracking and analysing the athlete's data is a requisite. This study introduces a cost-effective sensor-based system tailored for wheelchair basketball athletes, addressing the need for accessible and relevant performance metrics. Leveraging accelerometer and gyroscope sensors, key performance metrics including forward acceleration, velocity, distance covered, rotational dynamics, and total rotation were derived. While exhibiting certain limitations in longer tests and complex movements, the system provides valuable tools for athletes to assess and enhance their performance. Comparative cost analysis demonstrates the affordability and practicality of the designed system, offering a viable option for casual athletes seeking accessible performance metrics. Future research directions include implementing rigorous validation methodologies, integrating real-time metrics capabilities, enhancing accessibility through user-friendly interfaces, and streamlining the system for greater compactness and portability. These advancements aim to further enhance the reliability and usability of the system, contributing to the advancement of wheelchair basketball performance monitoring and adaptive sports technology.

Keywords: Wheelchair, Inertial Measurement Unit, Sport, Cost-effective, Performance

Loughborough University

1 Table of Contents

2 Introduction

Wheelchair basketball (WB), wheelchair tennis and wheelchair rugby are amongst the most popular Paralympic sports, with increasing popularity and international competitions being held worldwide [1]. As interest and professionalism in these sports grow, so does the necessity to improve team dynamics and individual athlete performance. For that, tracking and analysing the athlete's data is a requisite.

Wheelchair related performance in court sports is strongly determined by linear and rotational accelerations [2, 3]. The most commonly adopted approach for evaluating wheelchair kinematics uses three Inertial Measurement Units (3IMU). In this setup, one sensor is positioned on each wheel hub, while another is situated at the centre of the chair frame. By combining the data from these three sensors, it becomes possible to derive measurements such as wheel speed, frame rotation speed, and subsequently determine the overall wheelchair speed. A new algorithm employing the Madgwick filter has been devised, utilizing a single IMU positioned on a wheel hub. This algorithm integrates data from accelerometers, magnetometers, and gyroscopes to calculate the spatial orientation of the IMU. However, since the IMU is fixed on the wheel hub, there are uncertainties regarding its accuracy in determining frame rotation speed, turning radius, and wheelchair speed [4].

While there are various methodologies and products that exist in the market for wheelchair athletes to assess their kinematics and mobility performance, many are characterized by high costs and complex installation requirements [4] that might not be suitable for the everyday, casual athlete. For this demographic, affordability, ease of use, and practicality take precedence over high precision and complexity. Hence, there is a growing demand for low-cost, user-friendly solutions that offer valuable performance metrics to aid in game improvement and training optimization.

Citation	# of IMU	Gyroscope	Accelerometer	Magnetometer	Sampling Frequency	Real-Time	GPS	RF/BT	Wheel	Chair	Wrist
[16]	2	+2000°/s	±3 g	x	30–50 Hz	√	x	20 m[†]	√	x	x
[6]	2	+1600°/s	x	x	100 Hz	x	x	x	√	x	x
[17]	1	x	±8 g	x	60 Hz	x	x	x	x	√	x
[18]	3	x	±6 g	x	128 Hz	x	x	x	x	√	√
[19]	1	+2000°/s	x	x	50 Hz	x	x	x	√	x	x
[13]	1	+6000°/s	x	x	100 Hz	√	x	30 m[†]	√	x	x
[14]	3	+2000°/s	±8 g	x	256 Hz	x	x	x	√	√	x
[20]	3	+2000°/s	±8 g	x	256 Hz	x	x	x	√	√	x
[7]	2	+1200°/s	x	x	100 Hz	√	√	x	√	x	x
[21]	1	+1200°/s	x	x	100 Hz	x	x	x	x	√	x
[22]	1	±6000°/s	x	x	64 Hz	√	x	√[‡]	√	x	x
[23]	3	+2000°/s	±8 g	x	200 Hz	x	x	√[‡]	√	√	x
[24]	3	+2000°/s	±16 g	±7 Gauss	250 Hz	x	x	x	√	√	x
[12]	1	x	±8 g	x	60 Hz	x	x	x	x	√	x
[25]	3	±2000°/s	±16 g	±8.1 Gauss	200 Hz	x	x	x	√	x	x

Number (#) of IMU, inertial measurement unit; RF, radiofrequency; BT, Bluetooth; GPS, global positioning system.

Table 1 - Comparison of the instrumentation used by different authors to measure wheelchair kinematics [5].

In the domain of wheelchair kinematic performance metrics, the cost considerations associated with IMU (Inertial Measurement Unit) devices are significant. A review of other studies reveals that IMUs utilized for similar purposes often exceed £250 each in cost. Consequently, achieving accurate and reliable wheelchair kinematic measurements may necessitate an investment exceeding £500. Notable examples of IMU units utilized in prior research include the Shimmer3 IMU Unit by Shimmer Sensing (€430), NGIMU by X-IO technologies (£250), and the Movella Dot Sensor (€132).

The aim of this study is to develop a cost-effective, sensor-based system tailored specifically for wheelchair basketball athletes. By leveraging advancements in sensor technology and data analytics, our goal is to provide athletes with accessible and relevant performance metrics to optimize their skills and strategic decision-making. With a focus on affordability, ease of use, and practicality, our system aims to democratize access to performance monitoring tools, making them more readily available to athletes of all levels.

Ultimately, we anticipate that such a system could not only enhance individual athlete performance but also contribute to the growth and popularity of wheelchair basketball by lowering barriers to entry and increasing participation in the sport.

3 Methods

For this study, 6 participants were recruited from Loughborough University. Among them, four individuals possessed prior experience in playing wheelchair basketball casually, while the remaining participants had limited familiarity with wheelchair movement, possessing only basic knowledge.

This study was approved by the ethics committee of the Wolfson School of Mechanical, Electrical and Manufacturing Engineering, Loughborough University. All participants provided signed informed consent forms after going through the debriefing document and explaining the aims and objectives of the test to them.

3.1 System Overview

During the study, participants utilized the Multisport Wheelchair, manufactured by Motivation, which features 26-inch rear wheels set at a 15-degree camber angle [6]. The wheelchair was equipped with one externally powered Inertial Measurement Unit (IMU) (MPU6050, TDK InvenSense) measuring 3D linear acceleration and angular velocity at a sampling rate of approximately 84 Hz. It was mounted on the chair such that the positive z-axis was pointed vertically upwards, the positive x-axis was pointed forward and the positive y-axis was pointed rightward. The IMU sensor was configured with a range of ±8 g for the accelerometer and ±500°/s for the gyroscope. The sensor was mounted on the frame's rear axis and aligned centrally between the two wheel axles (Fig. 1c). Data was recorded directly onto a laptop (Asus VivoBook) as a .csv file via Bluetooth using the HC-05 Bluetooth Module (DSD Tech) and a serial console software installed onto the laptop (PuTTY, Simon Tatham). To complete the system setup, additional hardware components were utilized, including an Arduino Nano microcontroller (Arduino.cc), a 52mm x 82mm breadboard, jumper wires for connectivity, and a custom 3D printed mount. This mount was engineered to securely fasten the system onto the wheelchair's frame rod, ensuring stability and reliability throughout the testing process.

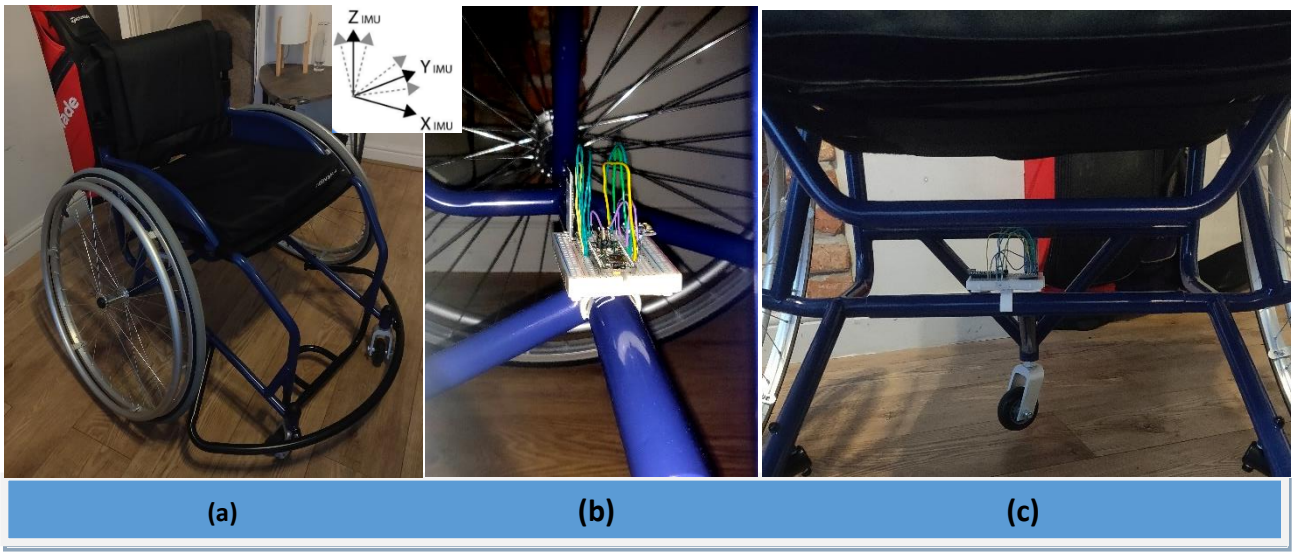

Figure 1 - (a) Multisport Wheelchair (Motivation) used in the experiment. Orthogonal axes indicate IMU orientation at starting position. (b/c) Lateral/Frontal view of wheelchair frame-mounted system

3.2 Mounting System Development

The design of the mount for fastening the system to the frame rod involved consideration of various requirements and factors to ensure optimal performance and compatibility.

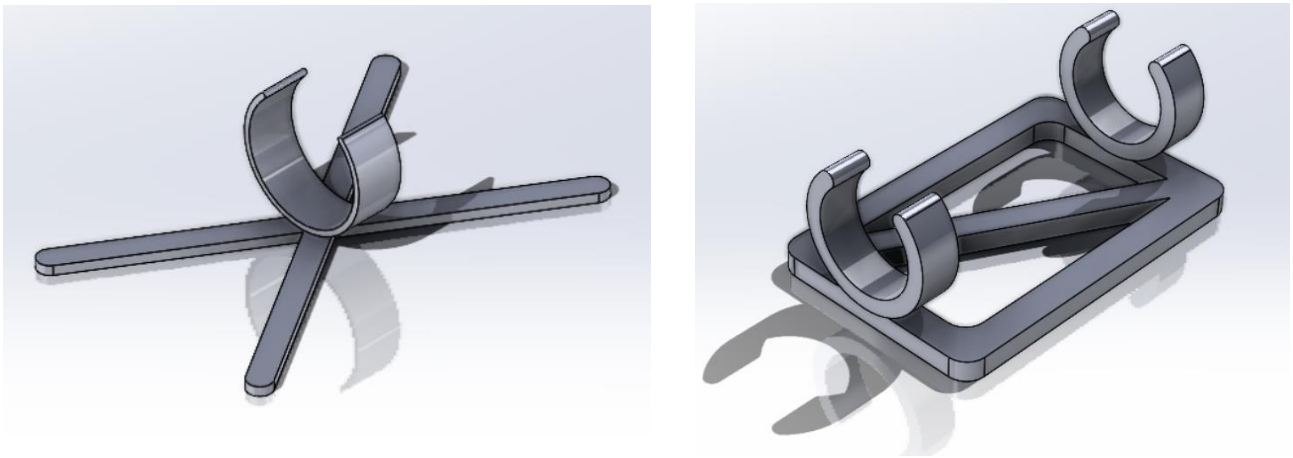

Figure 2 – Isometric view of both iterations of mount system. (1st iteration: left; 2nd iteration: right)

For example, the inclusion of two clips on either end of the mount's longer side provided additional support and minimized lateral instability compared to a single central clip (Figure. 3a). Additionally, a single diagonal bar was implemented across the platform to support the system, rather than two, thereby reducing weight while ensuring a stable surface for the breadboard (Figure. 3c).

Furthermore, to facilitate ease of use and adaptability, the mounts were designed with interchangeable clips (Figure. 3c) that could be swapped to accommodate different diameter rods or wheelchairs, thus enhancing the versatility of the mounting system.

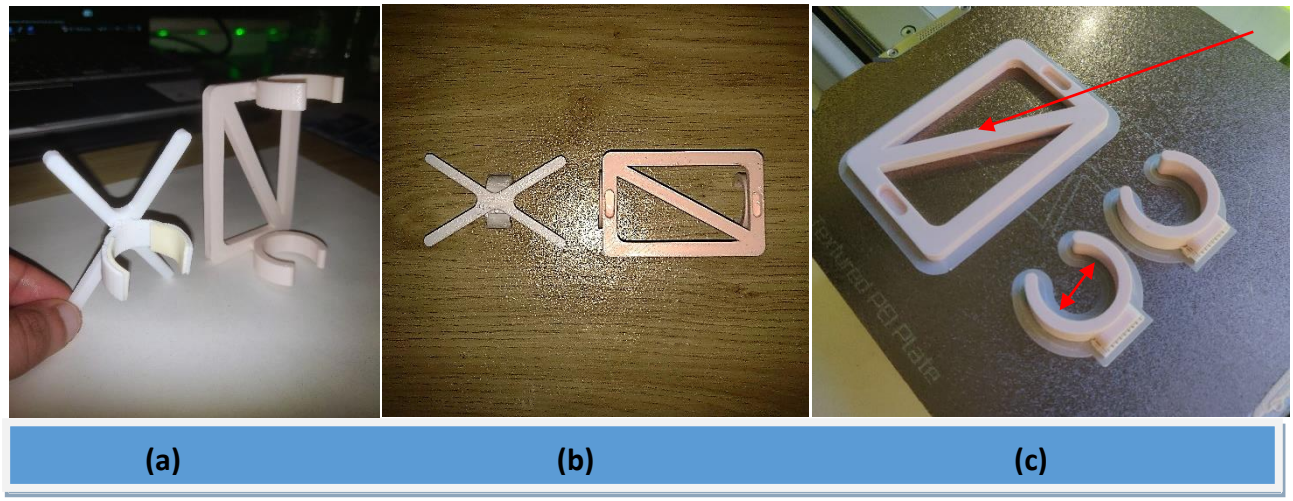

Figure 3 – (a) Inclusion of two clips on 2nd iteration print vs one clip on 1st iteration (b) Top-down view of both iterations (c) Interchangeable clips for adaptability

3.3 Testing Protocol

All testing took place at an indoor basketball court in the New Victory Hall (Holywell Fitness Centre, Loughborough University, UK).

Figure 4 - (Left) Indoor basketball court used for testing [7]; (Right) Cone setup for testing – arrows point to cone location

Participants performed a series of 7 tests consisting of different motion aspects of wheelchair basketball (Table 2) following a five minute free warmup. The majority of the tests mimic the tests conducted in [8]; the “Reverse-Straight” and “Distance Test” were added to the test by personal discretion.

Test	Name	Description
1	Straight - Normal	Straight forward movement at 70% effort over 5m, ending with a complete stop
2	Straight - Sprint	Straight forward sprint at 100% effort over 5m, ending with a complete stop
3	Reverse - Straight	Reverse movement for 5m, followed by a forward sprint over the same distance.
4	Slalom	Navigate around three cones spaced 2m apart and return to the starting position
5	U - Turn	Sprint towards a cone 5m away, execute a "U-turn" around the cone and sprint back to the starting cone
6	Pivot	Rotate 360° about one wheel, then repeat the rotation about the other wheel
7	Distance Test	Travel in a straight line between two cones spaced 5m apart, performing a 180° pivot turn at each cone for a minute and 50% effort.

Table 2 – Wheelchair Test Descriptions and Names

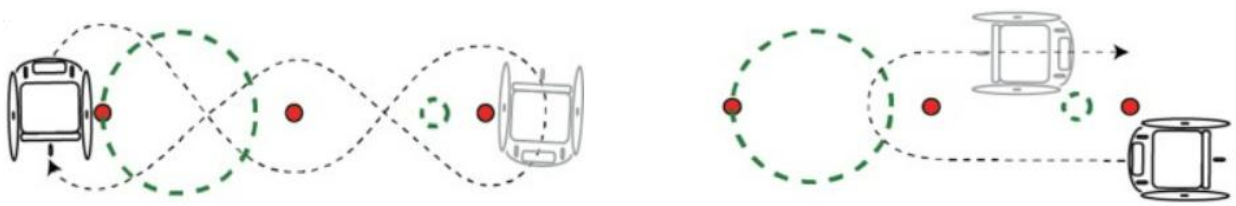

Figure 5 – Track layouts for (Left) Slalom drill and (Right) U - Turn drill [8]. Grey dashed line is the path taken by the participant and the red circles indicate cone positions

3.4 Data Collection

3.4.1 IMU Calibration

Prior to initiating data collection for a particular trial, the mounted IMU was first calibrated while the participant was asked to stay stationary with their hands on their lap for approximately 2-3 seconds.

The calibration process is inbuilt into the uploaded Arduino sketch [Appendix 1] and collects multiple samples (500 samples) of accelerometer data with a 10ms delay between each to determine the average offsets present in the sensor readings along the X, Y, and Z axes. It then adjusts these offsets during the trial to minimize inaccuracies and zero-errors, ensuring more precise accelerometer measurements in subsequent readings.

3.5 Data Analysis

All Inertial Measurement Unit (IMU) data collected during the trials were saved as comma-separated value (.csv) files and imported into RStudio (RStudio Team, 2020) for further processing and analysis.

3.5.1 Data Import, Cleaning, & Preprocessing

To ensure data integrity, consistency, and suitability for analysis and visualisation tasks, the .csv file imported into RStudio was first cleaned and transformed. This includes but is not limited to ensuring the data types for the variables are all correct, removal of unnecessary data, handling of missing data, etc [Appendix 2].

After importing and cleaning the wheelchair IMU data, it then underwent further processing to enhance its quality and extract meaningful insights. Techniques such as adding moving averages, low-pass filters, and complementary filters were applied to the data to remove noise, smooth out fluctuations, and improve signal clarity. These preprocessing steps are essential for refining the data and preparing it for subsequent analysis and visualization`.

To mitigate short-term fluctuations and noise inherent in the raw IMU data, a moving average technique was applied. Specifically, a simple moving average with a window size of 5 data points was employed using the SMA function from the TTR package (Technical Trading Rules, Joshua Ulrich). This process effectively smoothed out abrupt changes in the data, providing a clearer depiction of the underlying trends.

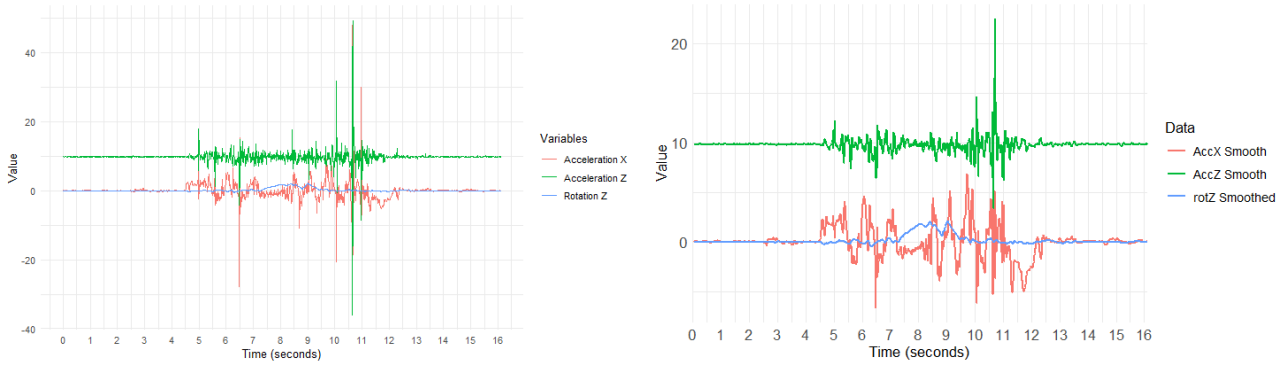

Figure 6 – (Left) Before adding moving average to trial (right) after adding moving average – AccZ peak reduced by about 20 m/s^2

Next, a Butterworth low-pass filter was introduced to further refine the data. The filter was designed with a cutoff frequency of 3Hz to attenuate high-frequency noise and fluctuations, ensuring that only the relevant components of the signal were preserved.

Finally, the data was cropped to accurately determine the trial's actual start and end times, corresponding to when the athlete commences movement and concludes the trial by coming to a halt. This involved identifying the starting and ending points where the signal exceeded predefined thresholds. Specifically, the indices where either the forward acceleration or rotational velocity about the z axis exceeded certain thresholds of ±1.2 m/s² and ±1.2 rad/s respectively were determined. After locating these points, the algorithm traversed backward from the start time and forward from the end time until the absolute value of the acceleration or rotational velocity became less than 0.2 m/s² or 0.2 rad/s, respectively. The data was then cropped to retain only the

portion of the signal corresponding to valid data points, excluding any erroneous sections [Appendix 2].

Finally, a new data frame, was created to store the cleaned and pre-processed data, and contained the relevant variables required for subsequent analysis and visualization tasks.

3.5.2 Performance Metrics Calculation

By analysing linear (acceleration) and rotational aspects of motion, we gain insights into the overall wheelchair-athlete's dynamic behaviour and performance during various manoeuvres. This includes metrics such as forward acceleration, forward velocity, distance covered, rotational acceleration, rotational velocity, and total rotation [Appendix 2].

To assess the wheelchair's forward acceleration, the collected x-axis accelerometer data was plotted against time and key metrics such as average and peak accelerations were extracted.

Velocity data was derived from the integrated acceleration measurements, enabling the visualization of the wheelchair's forward speed over time. Through this analysis, average and peak velocity values were identified.

By integrating the wheelchair's velocity data, the distance covered during the trial was calculated.

Rotational dynamics were assessed by computing rotational velocity about the z axis from the collected gyroscope data. Through a numerical differentiation process applied to the gyroscope outputs, rotational velocity was transformed into rotational acceleration. Rotational velocity data was processed to visualize the wheelchair's rotational velocity over time. Average and peak rotational velocity values were identified.

Total Rotation was computed by integrating the rotational velocity data. This metric, combined with the rotation direction indicator, offers insights into the directional aspects of wheelchair rotation [Appendix 2].

3.6 Data Validation

In order to ensure the integrity and reliability of the acquired IMU data, data validation techniques were employed.

Video data of the participants' trials was utilized to cross-reference and validate the IMU-derived metrics, offering a robust method to validate features in the plots, such as peaks, and identify any discrepancies.

Furthermore, instances of unusually high magnitude peaks or unexpected fluctuations in the data were flagged for further investigation. By visually inspecting the corresponding video segments, potential sources of anomalies, such as external disturbances or participant movements, were identified. Subsequent adjustments to the data preprocessing and filtering techniques were made accordingly to mitigate the impact of such anomalies and enhance the overall quality of the dataset.

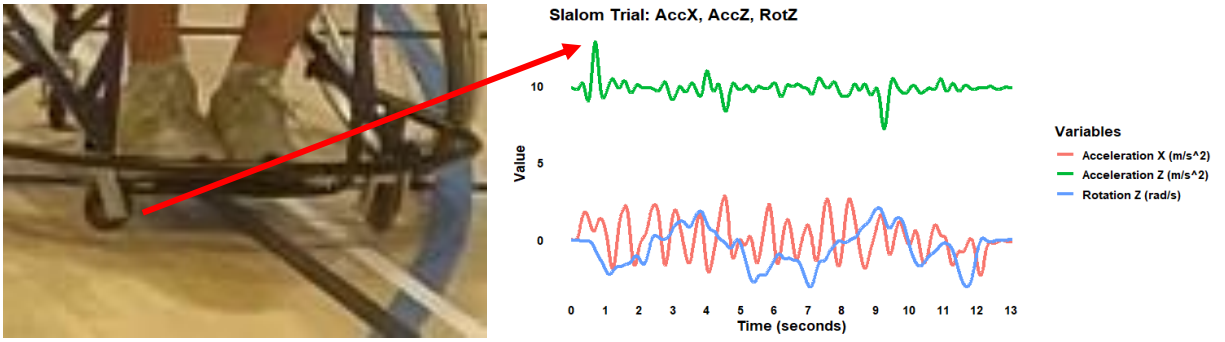

Figure 7 – Wheel rising and falling during acceleration causing sharp peak in vertical acceleration (attenuated by low pass filter)

The process of data validation and refinement was iterative in nature, with continuous feedback loops between the IMU data analysis and video verification stages. Feedback from the video analysis informed adjustments to the data processing pipelines, ensuring that the resulting metrics accurately reflected the underlying physical phenomena observed during the trials.

By leveraging complementary data sources, this approach facilitated a comprehensive understanding of the trial dynamics and ensured the validity of subsequent analyses and interpretations.

4 Results

4.1 Kinematic Outcomes Calculated

Table 2 provides a comprehensive overview of all main kinematic outcomes derived from the analysis of collected data, presenting key metrics and performance indicators.

Outcome Number	Description	Outcome Number	Description
1	Vertical Acceleration (m/s^2)	9	Rotational Acceleration (rad/s^2)
2	Forward Acceleration (m/s^2)	10	Peak Rotational Acceleration (rad/s^2)
3	Peak Forward Acceleration (m/s^2)	11	Average Rotational Acceleration (rad/s^2)
4	Average Forward Acceleration (m/s^2)	12	Rotational Velocity (rad/s)
5	Forward Speed (m/s)	13	Peak Velocity (rad/s)
6	Peak Forward Speed (m/s)	14	Average Velocity (rad/s)
7	Average Forward Speed (m/s)	15	Absolute Rotation (rad)
8	Distance Covered (m)	17	Average of best 5…

Table 3 – Overview of main kinematic outcomes calculated

4.2 Graphs of Kinematic Outcomes

4.2.1 Linear Kinematics

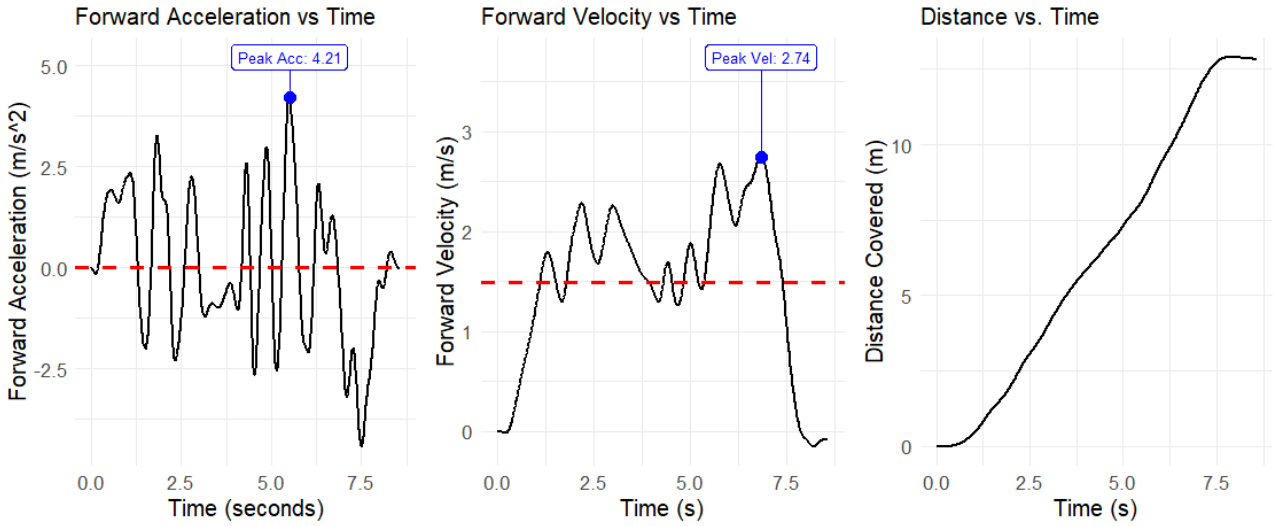

Figure 9 – Linear kinematics graphs of the "U-turn" drill

4.2.2 Rotational Kinematics

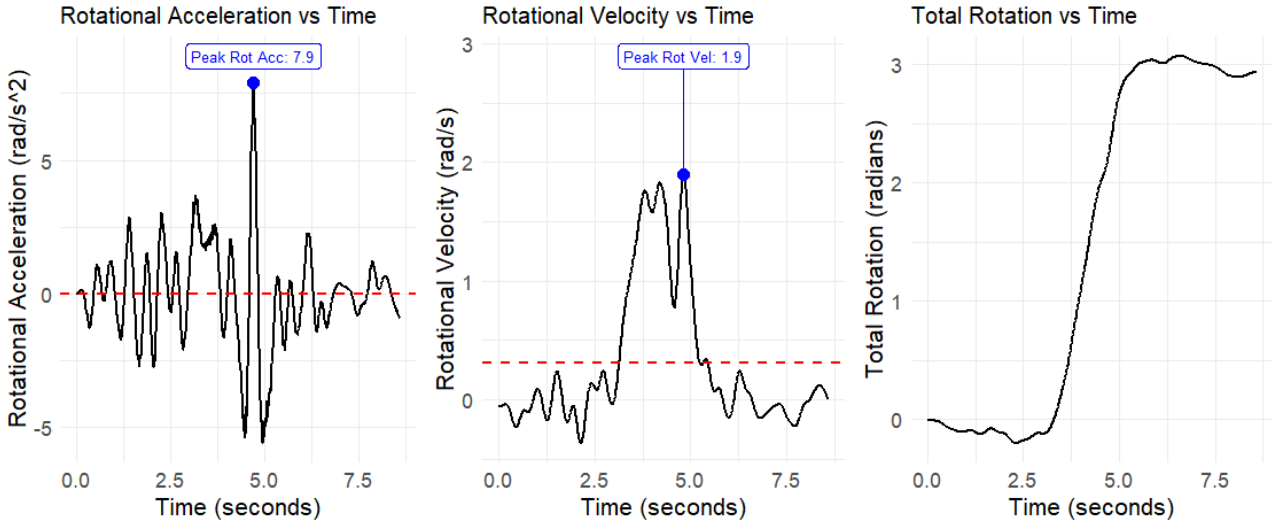

Figure 10 – Rotational kinematic graphs of the "U-turn" drill

4.3 Interactive Data Visualization

Shiny is an R package that enables the creation of web applications directly from R scripts, allowing for the integration of data analysis, visualization, and user interaction in a seamless manner. A Shiny app was developed as a means of a prototype of the user interface that would be used by the athlete or coach to import the athlete's wheelchair data and allow for them to view their statistics and performance metrics all in one place and displayed in a meaningful manner. The user can choose a specific time frame as well and it will recalculate all the metrics.

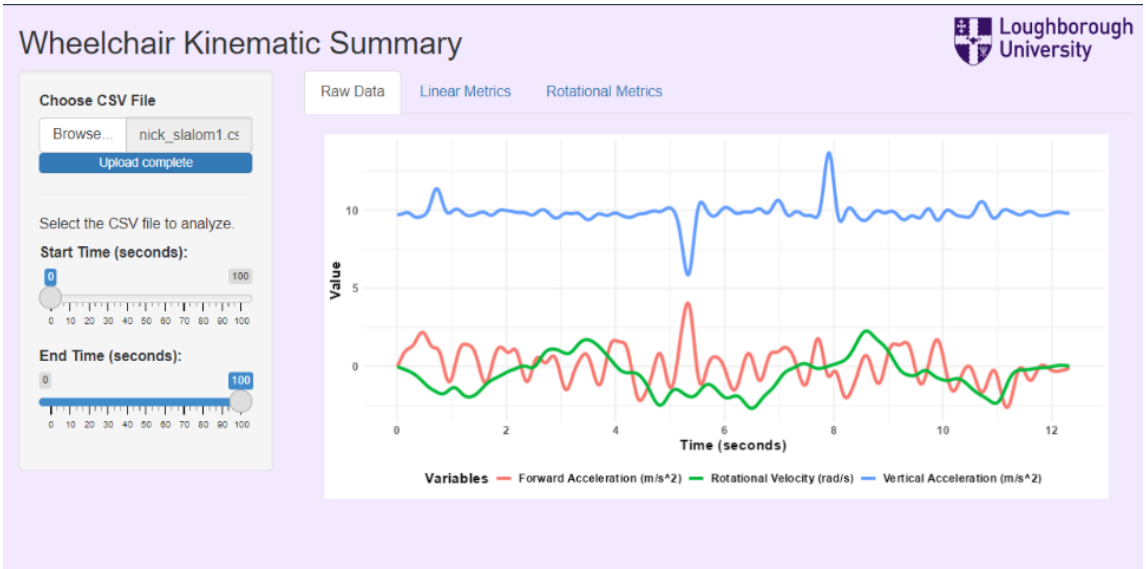

Figure 11 – "Raw Data" tab on developed UI

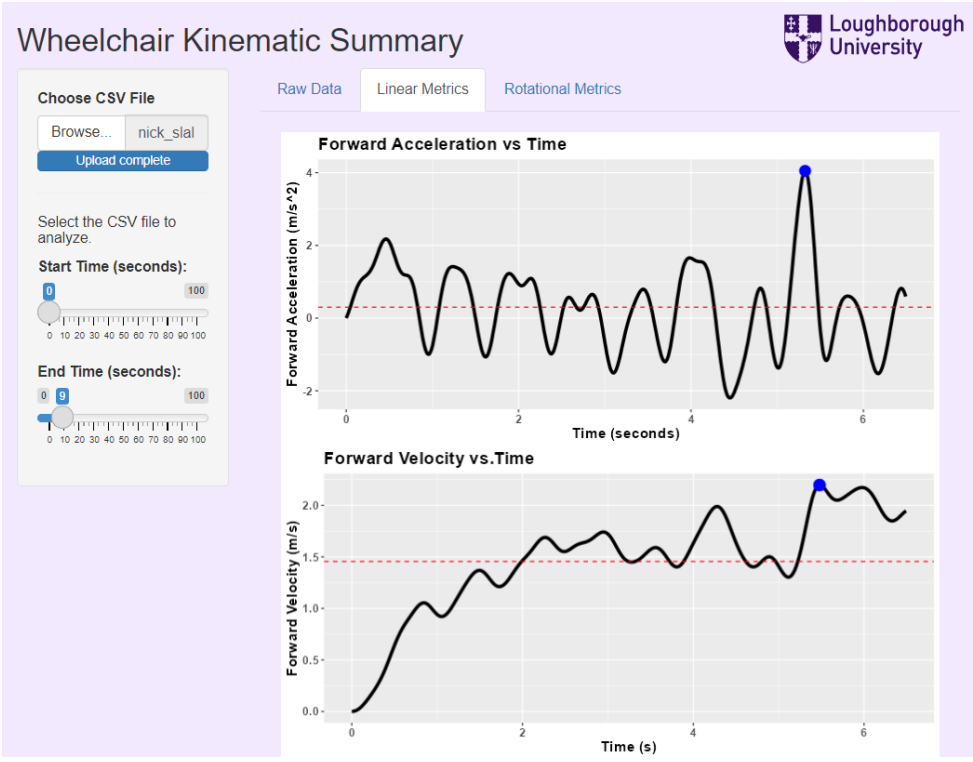

Figure 12 – Top of "Linear Metrics" tab on developed UI. Peak dictated by blue dot; average dictated by dashed red line

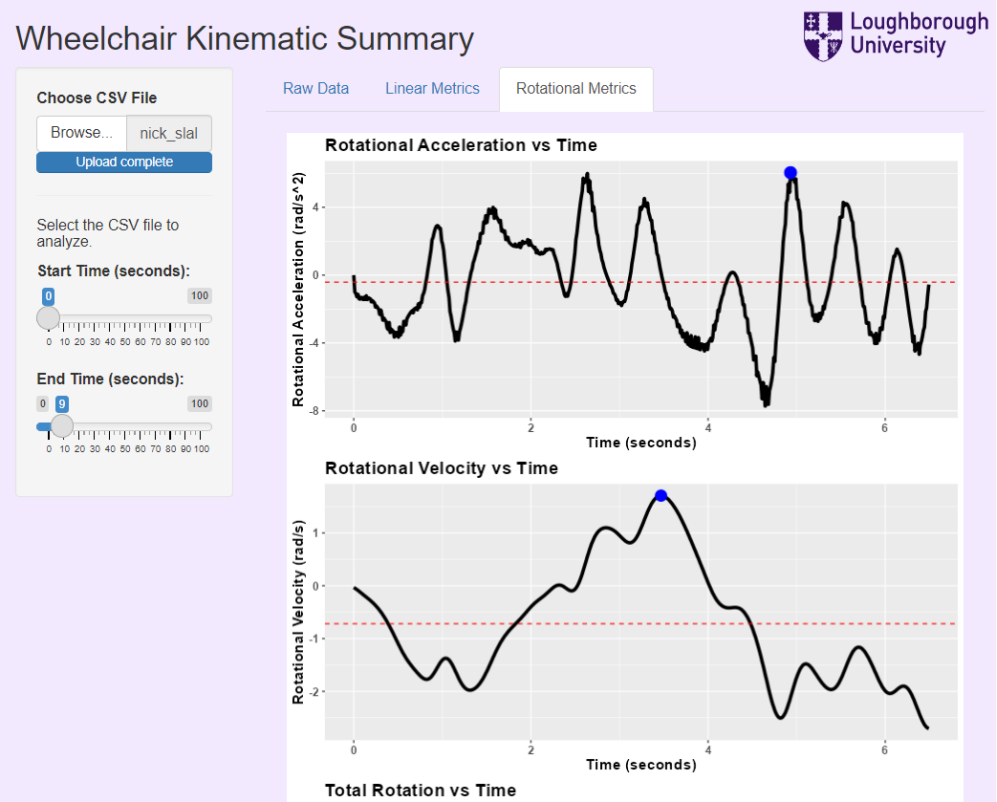

Figure 13 – Top of "Rotational Metrics" tab on developed UI

5 Discussion

Upon reviewing our analysis results, it's evident that the generated graphs offer utility for wheelchair basketball athletes, aiming to align with the standards outlined by the Wheelchair Mobility Plot [9]. While these visual representations provide valuable insights into performance metrics, it's important to acknowledge the system's limitations. The integration of data from accelerometer and gyroscope sensors offers valuable insights into wheelchair mobility; however, the system may exhibit deviations, particularly in longer tests like the "Distance test" or those involving complex movements.

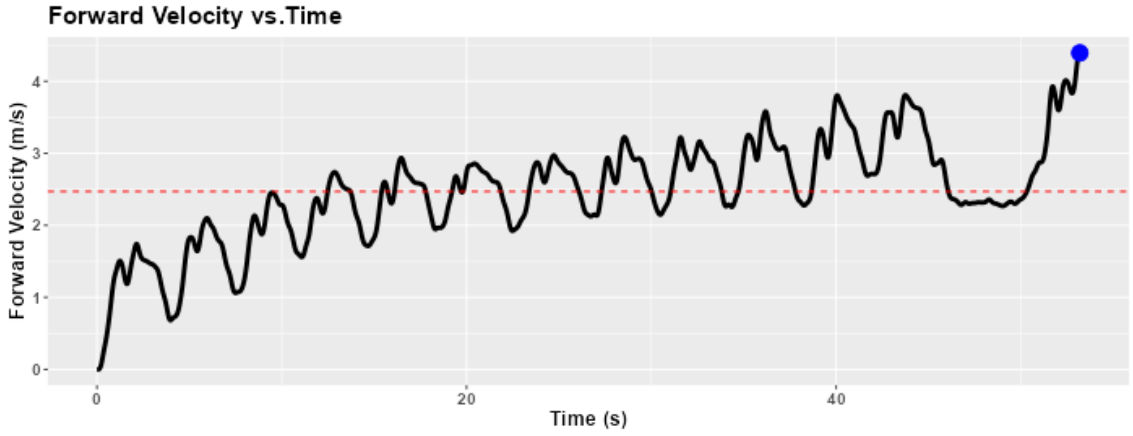

Figure 14 – Velocity vs Time graph for a participant "Distance Test". Note the upwards trend of forward velocity (m/s) - should be more level

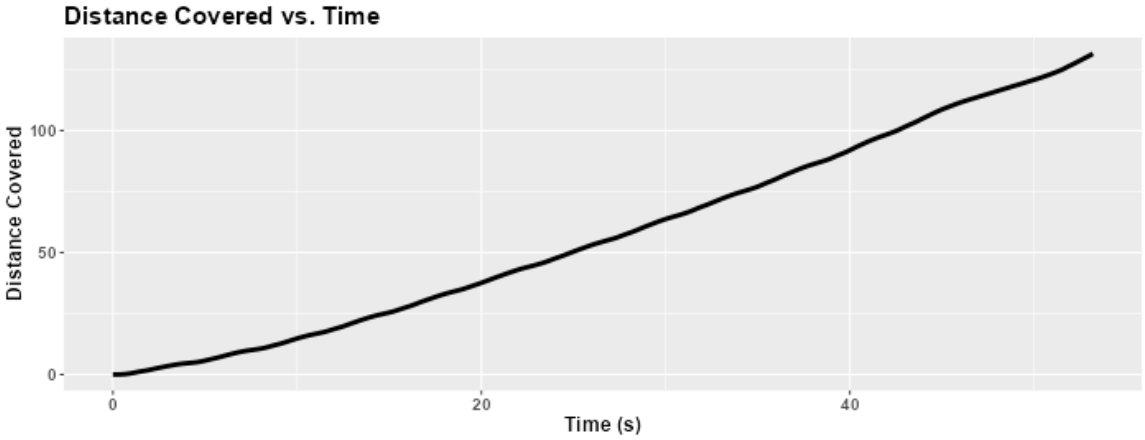

Figure 15 – Distance values from the same test as Figure 14 - far off from the true value (which should be around 70m)

Despite these challenges, the graphs remain valuable tools for athletes, providing a foundational understanding of their performance. As we continue to refine our methodologies and address limitations, we aim to enhance the reliability and accuracy of our system for more robust sports performance evaluation.

5.1 Limitations

In our exploration of the data, several limitations have come to light. Firstly, during the second testing session, we identified that the wheelchair tires were somewhat deflated. While seemingly minor, this observation could have introduced discrepancies or made it more challenging to achieve a consistent velocity. Such variations in tire pressure underline the importance of meticulous attention to equipment maintenance to ensure data accuracy and reliability.

Moreover, our validation methodology, which relies on time and video synchronization, presents another limitation. While effective to a certain extent, this approach lacks the precision afforded by a gold standard reference measurement system, such as a 24-camera 3D optical motion analysis system [2]. The inherent limitations of our validation techniques necessitate acknowledgment, emphasizing the need for continued exploration and refinement of validation methodologies to enhance data accuracy and robustness.

Additionally, our analysis overlooks the impact of wheel skid, a phenomenon common in wheelchair manoeuvres [10]. The omission of wheel skid from our data analysis could lead to significant variations in the data, potentially affecting the interpretation of results. However, it's essential to recognize that addressing wheel skid would likely entail additional costs, underscoring the inherent trade-offs between affordability and comprehensive data capture.

5.2 Cost of Designed System

Component Description	Quantity (no.)	Unit Price (£)	Total Price(£)
Arduino Nano	1	18.5	18.5
MPU6050	1	7	7
HC-05 Bluetooth Module	1	6.99	6.99
2200 mAh Power bank	1	6.9	6.9
Jumper cables	1 pack (40 pcs)	4.15	4.15
Breadboard	1	3.99	3.99
Custom 3D - Printed Wheelchair Mount	14g	0.015 per gram	0.22
		Total price:	47.75

Table 4 - The overall cost of the system is described

Shimmer3 IMU Unit (Shimmer Sensing)	NGIMU by X-IO technologies	Movella Dot Sensor	Designed System
€430	£250	€132	~£50

Table 5 - Cost comparison with other IMU's used in research

5.3 Future Directions

One key avenue for future exploration involves the implementation of more rigorous validation methodologies. While our current approach includes comparing data with video footage and referencing common values from existing literature, additional validation using more accurate methods is warranted. Incorporating gold standard reference measurement systems, such as 3D optical motion analysis systems, would offer a higher degree of precision and reliability in validating wheelchair kinematic performance metrics.

Moving forward, the integration of real-time metrics capabilities represents another crucial area for development in our research endeavours. Real-time data acquisition and analysis offer several advantages, including immediate feedback for athletes and coaches during training sessions and competitions. By enabling the real-time monitoring of key performance metrics, such as acceleration, velocity, and distance travelled, athletes can make timely adjustments to their techniques and strategies, optimizing their performance outcomes.

In addition to real-time metrics, enhancing accessibility to performance statistics is paramount for empowering athletes and coaches alike. By developing user-friendly interfaces that are accessible

via smartphones or personal computers, we can democratize access to critical performance data. Such interfaces would enable athletes to review their performance metrics at their convenience, facilitating self-directed learning and goal setting. Furthermore, coaches can leverage these platforms to track athletes' progress over time, identify areas for improvement, and tailor training programs to individual needs.

Furthermore, streamlining the overall system to achieve greater compactness and portability is a priority for future development efforts. The current system's size and form factor may pose logistical challenges, particularly in competitive or constrained environments. By miniaturizing components and optimizing the design, we can create a more compact and portable solution that offers greater flexibility and usability.

6 Conclusion

This study presents a novel approach to wheelchair basketball kinematic performance monitoring, offering valuable insights and practical implications for athletes and coaches alike. By developing a cost-effective sensor-based system tailored specifically for wheelchair basketball, we have addressed the need for accessible and relevant performance metrics in this domain.

Our findings demonstrate the utility of integrating accelerometer and gyroscope sensors to derive key performance metrics, such as forward acceleration, velocity, distance covered, rotational dynamics, and total rotation. Despite certain limitations, such as deviations observed in longer tests and challenges in capturing complex movements accurately, our system provides valuable tools for athletes to assess and enhance their performance.

Moreover, our cost analysis reveals the affordability and practicality of our designed system compared to existing IMU solutions in the market. With a total system cost of approximately £50, our solution provides a cost-effective alternative that prioritizes affordability and practicality while offering basic functionality for performance monitoring. While it may not match the precision of higher-end systems, it serves as a viable option for casual athletes seeking accessible performance metrics. As such, its applicability may vary depending on the specific needs and expectations of individual users.

Looking ahead, future directions for research include implementing more rigorous validation methodologies, integrating real-time metrics capabilities, enhancing accessibility to performance statistics through user-friendly interfaces, and streamlining the overall system for greater compactness and portability. By addressing these areas of improvement, we aim to further enhance the reliability and usability of our system, ultimately contributing to the advancement of wheelchair basketball performance monitoring and the broader field of adaptive sports technology.

7 References

[1] R. M. A. van der Slikke, M. A. M. Berger, D. J. J. Bregman, and D. H. E. J. Veeger, "Wearable Wheelchair Mobility Performance Measurement in Basketball, Rugby, and Tennis: Lessons for Classification and Training," Sensors, vol. 20, no. 12, p. 3518, Jun. 2020, doi: https://doi.org/10.3390/s20123518.

[2] R. M. A. Van Der Slikke, M. Berger, D. J. J. Bregman, A. Lagerberg, and H. E. J. Veeger, "Opportunities for measuring wheelchair kinematics in match settings; reliability of a three inertial sensor configuration," Journal of Biomechanics, vol. 48, no. 12, pp. 3398–3405, Sep. 2015, doi: 10.1016/j.jbiomech.2015.06.001.

[3] R. M. A. Van Der Slikke, M. Berger, D. J. J. Bregman, and H. E. J. Veeger, "From big data to rich data: The key features of athlete wheelchair mobility performance," Journal of Biomechanics, vol. 49, no. 14, pp. 3340–3346, Oct. 2016, doi: 10.1016/j.jbiomech.2016.08.022.

[4] R. Rupf, M. Tsai, S. Thomas, and M. Klimstra, "Original article: Validity of measuring wheelchair kinematics using one inertial measurement unit during commonly used testing protocols in elite wheelchair court sports," Journal of Biomechanics, vol. 127, p. 110654, Oct. 2021, doi: 10.1016/j.jbiomech.2021.110654.

[5] J. Shepherd, D. James, H. Espinosa, D. Thiel, and D. Rowlands, "A Literature Review Informing an Operational Guideline for Inertial Sensor Propulsion Measurement in Wheelchair Court Sports," Sports, vol. 6, no. 2, p. 34, Apr. 2018, doi: https://doi.org/10.3390/sports6020034.

[6] "Sport's Wheelchairs," Motivation. https://motivation.org.uk/our-work/products/sports-wheelchairs/#multisport

[7] "Sports Halls | Sport | Loughborough University," www.lboro.ac.uk. https://www.lboro.ac.uk/sport/facilities/sports-facilities/sports-halls/

[8] M. P. van Dijk, R. M. A. van der Slikke, R. Rupf, M. J. M. Hoozemans, M. A. M. Berger, and D. H. E. J. Veeger, "Obtaining wheelchair kinematics with one sensor only? The trade-off between number of inertial sensors and accuracy for measuring wheelchair mobility performance in sports," Journal of Biomechanics, vol. 130, p. 110879, Jan. 2022, doi: https://doi.org/10.1016/j.jbiomech.2021.110879.

[9] R. M. A. van der Slikke, M. A. M. Berger, D. J. J. Bregman, and H. E. J. Veeger, "From big data to rich data: The key features of athlete wheelchair mobility performance," Journal of Biomechanics, vol. 49, no. 14, pp. 3340–3346, Oct. 2016, doi: https://doi.org/10.1016/j.jbiomech.2016.08.022.

[10] van, Monique A.M. Berger, D. J. J. Bregman, and H.E.J. Veeger, "Wheel Skid Correction is a Prerequisite to Reliably Measure Wheelchair Sports Kinematics Based on Inertial Sensors," Procedia Engineering, vol. 112, pp. 207–212, Jan. 2015, doi: https://doi.org/10.1016/j.proeng.2015.07.201.

8 Appendix

8.1 Appendix 1 – Arduino Sketch

```
#include <Wire.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <SoftwareSerial.h> // Include the SoftwareSerial library

Adafruit_MPU6050 mpu;

unsigned long startTime = 0; // Variable to store the start time
unsigned long elapsedTime = 0; // Variable to store the elapsed time
bool startReading = false; // Flag to indicate whether to start readings
bool readingInProgress = false; // Flag to indicate whether readings are
currently being taken
bool calibrationInProgress = false; // Flag to indicate whether calibration is
in progress
bool calibrationCompleted = false; // Flag to indicate whether calibration is
completed

// Calibration variables
float accelOffsets[3] = {0}; // Accelerometer offsets for x, y, z axes

SoftwareSerial BTSerial(0, 1); // RX, TX (connect Bluetooth module RX to pin 0,
and TX to pin 1)

void setup() {
  BTSerial.begin(115200); // Initialize Bluetooth serial communication

  if (!mpu.begin()) {
    BTSerial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }

  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);

  BTSerial.println("MPU6050 Acceleration and Rotation Test");
  BTSerial.println("Type 'C' to calibrate accelerometer and 'S' to start
readings.");
}

void loop() {
  if (BTSerial.available() > 0) {
```

```
    char receivedChar = BTSerial.read();
    if (receivedChar == 'C' && !readingInProgress && !calibrationInProgress) {
      calibrationCompleted = false; // Reset calibration flag
      calibrationInProgress = true; // Set calibration in progress flag
      calibrateAccelerometer();
    } else if (receivedChar == 'S' && calibrationCompleted) {
      if (!readingInProgress) {
        startReading = true;
        readingInProgress = true;
        startTime = 0;

        BTSerial.println();
        BTSerial.println("Time (ms), X (m/s^2), Y (m/s^2), Z (m/s^2), X (rad/s),
Y (rad/s), Z (rad/s)");
      }
    } else if (receivedChar == 'X' && readingInProgress) {
      readingInProgress = false;
      startReading = false;
    } else if (receivedChar == 'X' && calibrationInProgress) {
      calibrationInProgress = false;
    }
  }

  if (startReading && readingInProgress) {
    sensors_event_t accelEvent, gyroEvent, tempEvent;
    mpu.getEvent(&accelEvent, &gyroEvent, &tempEvent);

    accelEvent.acceleration.x += accelOffsets[0];
    accelEvent.acceleration.y += accelOffsets[1];
    accelEvent.acceleration.z += accelOffsets[2];

    if (startTime == 0) {
      startTime = millis();
    }
    elapsedTime = millis() - startTime;

    BTSerial.print(elapsedTime);
    BTSerial.print(", ");
    BTSerial.print(accelEvent.acceleration.x, 4);
    BTSerial.print(", ");
    BTSerial.print(accelEvent.acceleration.y, 4);
    BTSerial.print(", ");
    BTSerial.print(accelEvent.acceleration.z, 4);
    BTSerial.print(", ");
    BTSerial.print(gyroEvent.gyro.x, 4);
```

```
    BTSerial.print(", ");
    BTSerial.print(gyroEvent.gyro.y, 4);
    BTSerial.print(", ");
    BTSerial.println(gyroEvent.gyro.z, 4);
  }
}

void calibrateAccelerometer() {
  const int numSamples = 500;
  float accelX = 0, accelY = 0, accelZ = 0;

  BTSerial.println("Calibrating accelerometer...");

  for (int i = 0; i < numSamples; i++) {
    sensors_event_t accelEvent, gyroEvent, tempEvent;
    mpu.getEvent(&accelEvent, &gyroEvent, &tempEvent);

    accelX += accelEvent.acceleration.x;
    accelY += accelEvent.acceleration.y;
    accelZ += accelEvent.acceleration.z;

    delay(10);
  }

  accelOffsets[0] = -accelX / numSamples;
  accelOffsets[1] = -accelY / numSamples;
  accelOffsets[2] = 9.81 - (accelZ / numSamples);

  BTSerial.println("Calibration Complete");
  BTSerial.print("Accel Offsets (X, Y, Z): ");
  BTSerial.print(accelOffsets[0]);
  BTSerial.print(", ");
  BTSerial.print(accelOffsets[1]);
  BTSerial.print(", ");
  BTSerial.println(accelOffsets[2]);

  //BTSerial.println("Type 'S' to start readings.");
  calibrationCompleted = true;
  calibrationInProgress = false;
}
```

8.2 Appendix 2 – R Code

```
library(tidyverse)


# Define the file path (add code for what to do if file is not found)
file_path <- "C:/Users/nicks/Desktop/IDP R
Scripts/real_trials2/nick_slalom1.csv"


# Read the CSV file, skipping lines containing "PuTTY log" and enforcing 7
columns
data <- read.csv(file_path, header = FALSE, col.names = paste0("V", 1:7))


# Remove rows containing the word "putty" (case insensitive) in the first column
data <- data %>%
  dplyr::filter(!grepl("putty", V1, ignore.case = TRUE))


# Find the index of the last row with specified column names
last_row_index <- max(which(
  data$V1 == "Time (ms)"
))


# Remove rows before the last occurrence
data <- data[(last_row_index):nrow(data), ]



# Assign the first row as column names
colnames(data) <- unlist(data[1, ])


# Remove the first row
data <- data[-1, ]


# Reset row names to be consecutive (1st row is row 1)
rownames(data) <- NULL


# Remove units from column names
```

```
colnames(data) <- gsub("\\s*\\(.*\\)", "", colnames(data))

# Ensure column names are unique
colnames(data) <- make.unique(colnames(data))

# Convert all variables in the dataframe individually to numeric, handling non-numeric values
data <- mutate_all(data, ~as.numeric(as.character(.)))

# Rename columns directly
colnames(data) <- c("time", "accX", "accY", "accZ", "rotX", "rotY", "rotZ")

data1 <- data

# Convert "time" variable from milliseconds to seconds
data1$time_s <- data1$time / 1000

# Reorder the columns to place "time_s" before "time"
data1 <- data1[, c("time_s", names(data1)[names(data1) != "time_s"])]

# Plotting final "Raw" graph
ggplot(data1, aes(x = time_s)) +
  geom_line(aes(y = accX, color = "Acceleration X")) +
  geom_line(aes(y = accZ, color = "Acceleration Z")) +
  geom_line(aes(y = rotZ, color = "Rotation Z")) +
  labs(x = "Time (seconds)", y = "Value", color = "Variables") +
  scale_x_continuous(breaks = seq(0, max(data1$time_s), by = 2)) +
  theme_minimal()

############ Moving Avg (Change window size individually later #################

library(TTR)
```

```
# Define the window size for the moving average
window_size <- 5

# Calculate the moving average using SMA function from TTR package
data1$accX_smoothed <- SMA(data1$accX, n = window_size)
data1$accY_smoothed <- SMA(data1$accY, n = window_size)
data1$accZ_smoothed <- SMA(data1$accZ, n = window_size)
data1$rotX_smoothed <- SMA(data1$rotX, n = window_size)
data1$rotY_smoothed <- SMA(data1$rotY, n = window_size)
data1$rotZ_smoothed <- SMA(data1$rotZ, n = window_size)



######################### Butterworth low pass filter ############################

library(signal)

# Define the sampling frequency(Hz)
sampling_frequency <- 84

# Define the cutoff frequency for the low-pass filter (3 Hz)
cutoff_frequency <- 3

# Calculate the normalized cutoff frequency (0 to 1, where 1 represents Nyquist
frequency)
normalized_cutoff <- cutoff_frequency / (sampling_frequency / 2)

# Design a low-pass Butterworth filter
butterworth_filter <- butter(4, normalized_cutoff, type = "low")

# Apply the filter to the "rotZ" values
data1$accX_adj <- filtfilt(butterworth_filter, data1$accX_smoothed)
data1$accY_adj <- filtfilt(butterworth_filter, data1$accY_smoothed)
data1$accZ_adj <- filtfilt(butterworth_filter, data1$accZ_smoothed)
```

```
data1$rotX_adj <- filtfilt(butterworth_filter, data1$rotX_smoothed)

data1$rotY_adj <- filtfilt(butterworth_filter, data1$rotY_smoothed)

data1$rotZ_adj <- filtfilt(butterworth_filter, data1$rotZ_smoothed)


# Convert all variables in the dataframe individually to numeric, handling non-
numeric values

data1 <- mutate_all(data1, ~as.numeric(as.character(.)))



# Filtering the dataframe to remove start and end error from low pass filter

data2 <- subset(data1, time_s >= 0.5 & time_s < max(data1$time_s) - 0.5)


################################## Crop Data ####################################


# Find the index where either accX_adj or rotZ_adj goes above the absolute value
of 1

start_index <- which(abs(data2$accX_adj) > 1.2 | abs(data2$rotZ_adj) > 1.2)[1]


if (!is.na(start_index)) {

  while (start_index > 1) {

    if (abs(data2$accX_adj[start_index]) < 0.002 |
abs(data2$rotZ_adj[start_index]) < 0.002) break

    start_index <- start_index - 1

  }


  cat("Starting time:", data2$time_s[start_index], "seconds\n")

} else {

  cat("No valid starting point found.\n")

}


# Find the index where either accX_adj or rotZ_adj exceeds the absolute value of
1.2, starting from the end of the dataframe

end_index <- length(data2$accX_adj) - which(abs(rev(data2$accX_adj)) > 1.2 |
abs(rev(data2$rotZ_adj)) > 1.2)[1] + 1
```

```
if (!is.na(end_index)) {
  # Find the index where either accX_adj or rotZ_adj goes below the absolute value of 0.0075, moving forward in time from the end_index
  while (end_index < nrow(data2)) {
    if (abs(data2$accX_adj[end_index]) < 0.002 | abs(data2$rotZ_adj[end_index]) < 0.002) break
    end_index <- end_index + 1
  }

  cat("Ending time:", data2$time_s[end_index], "seconds\n")
} else {
  cat("No valid ending point found.\n")
}



############################## Make Final Dataframe ##############################

# Create a new dataframe containing data from the starting point till the end (this can be imported into shiny?)
sensor_data <- data2[start_index:end_index, ]

# Remove variables
sensor_data <- subset(sensor_data, select = -c(accX, accY, accZ, rotX, rotY, rotZ, accX_smoothed, accY_smoothed, accZ_smoothed, rotX_smoothed, rotY_smoothed, rotZ_smoothed))

# Rename columns
names(sensor_data) <- gsub("_adj", "", names(sensor_data))

# Reset row names to start from 1
rownames(sensor_data) <- NULL

sensor_data <- mutate(sensor_data, time_s = time_s - min(time_s))
```

```
# Create a column to indicate the direction of rotation
sensor_data$rotation_direction <- ifelse(sensor_data$rotZ > 0, "Anticlockwise", "Clockwise")


# Plotting final "Raw" graph
ggplot(sensor_data, aes(x = time_s)) +
  geom_line(aes(y = accX, color = "Acceleration X (m/s^2)"), size = 1.5) +
  geom_line(aes(y = accZ, color = "Acceleration Z (m/s^2)"), size = 1.5) +
  geom_line(aes(y = rotZ, color = "Rotation Z (rad/s)"), size = 1.5) +
  labs(x = "Time (seconds)", y = "Value", color = "Variables", title = "Slalom Trial: AccX, AccZ, RotZ") +
  scale_x_continuous(breaks = seq(0, max(sensor_data$time_s), by = 1)) +
  theme_minimal() +
  theme(
    axis.text = element_text(size = 10, color = "black", face = "bold"),
    axis.title = element_text(size = 12, color = "black", face = "bold"),
    legend.text = element_text(size = 10, color = "black", face = "bold"),
    legend.title = element_text(size = 12, color = "black", face = "bold"),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    plot.title = element_text(size = 14, color = "black", face = "bold"),
    plot.subtitle = element_text(size = 12, color = "black", face = "bold"),
  )
```

8.3 Appendix 3 – R Shiny App Code

```
library(shiny)
library(tidyverse)
library(TTR)
library(signal)

# Define UI
ui <- fluidPage(
  titlePanel("Wheelchair Kinematic Summary"),
```

```
  # Add background color to the entire app
  tags$style(type = "text/css", "body {background-color: #f2e9ff;}"),

  # Add logo image to the top right corner
  tags$img(src = "lu_logo.png", height = 50, width = 175, style = "position:
absolute; top: 10px; right: 10px;"),

  sidebarLayout(
    sidebarPanel(
      width = 3,  # Adjust the width of the sidebar panel
      fileInput("file", "Choose CSV File",
                accept = c(".csv")),
      tags$hr(),
      h5("Select the CSV file to analyze."),

      # Slider inputs for start and end times
      sliderInput("start_time", "Start Time (seconds):", min = 0, max = 100,
value = 0),
      sliderInput("end_time", "End Time (seconds):", min = 0, max = 100, value =
100)
    ),

    mainPanel(
      width = 9,  # Adjust the width of the main panel
      tabsetPanel(
        id = "tabs",
        tabPanel("Raw Data", plotOutput("raw_plot", height = "300px")),  #
Adjust height here
        tabPanel("Linear Metrics",
                 plotOutput("forward_acceleration", height = "300px"),  # Adjust
height here
                 plotOutput("forward_velocity", height = "300px"),  # Adjust
height here
```

```
                plotOutput("distance_covered", height = "300px")),  # Adjust
height here
        tabPanel("Rotational Metrics",
                plotOutput("rotational_acceleration", height = "300px"),  #
Adjust height here
                plotOutput("rotational_velocity", height = "300px"),  # Adjust
height here
                plotOutput("total_rotation", height = "300px"))  # Adjust
height here
      )
    )
  ),

  # Adjust UI style
  tags$head(
    tags$style(HTML("
                    .tab-content {
                      padding: 20px;
                    }
                    .navbar-default {
                      background-color: #f2f2f7;
                      border-color: #e7e7e7;
                    }
                    .sidebar-panel {
                      border-right: 1px solid #ddd;
                      padding-top: 20px;
                    }
                    "))
  )
)

# Define server logic
server <- function(input, output) {
```

```
  data <- reactive({
    req(input$file)

    # Read the CSV file
    data <- read.csv(input$file$datapath, header = FALSE, col.names =
paste0("V", 1:7))

    # Remove unwanted rows
    data <- data %>%
      dplyr::filter(!grepl("putty", V1, ignore.case = TRUE)) %>%
      slice((which(V1 == "Time (ms)") + 1):n())

    # Assign column names
    colnames(data) <- unlist(data[1, ])

    # Remove first row
    data <- data[-1, ]

    # Reset row names
    rownames(data) <- NULL

    # Remove units from column names
    colnames(data) <- gsub("\\s*\\(.*\\)", "", colnames(data))

    # Ensure column names are unique
    colnames(data) <- make.unique(colnames(data))

    # Convert variables to numeric
    data <- mutate_all(data, ~as.numeric(as.character(.)))

    # Rename columns
    colnames(data) <- c("time", "accX", "accY", "accZ", "rotX", "rotY", "rotZ")
```

```
# Convert time variable from milliseconds to seconds
data$time_s <- data$time / 1000

# Calculate moving average
window_size <- 5
data$accX_smoothed <- SMA(data$accX, n = window_size)
data$accY_smoothed <- SMA(data$accY, n = window_size)
data$accZ_smoothed <- SMA(data$accZ, n = window_size)
data$rotX_smoothed <- SMA(data$rotX, n = window_size)
data$rotY_smoothed <- SMA(data$rotY, n = window_size)
data$rotZ_smoothed <- SMA(data$rotZ, n = window_size)

# Define cutoff frequency for low-pass filter
sampling_frequency <- 84
cutoff_frequency <- 3
normalized_cutoff <- cutoff_frequency / (sampling_frequency / 2)

# Design Butterworth low-pass filter
butterworth_filter <- butter(4, normalized_cutoff, type = "low")

# Apply filter
data$accX_adj <- filtfilt(butterworth_filter, data$accX_smoothed)
data$accY_adj <- filtfilt(butterworth_filter, data$accY_smoothed)
data$accZ_adj <- filtfilt(butterworth_filter, data$accZ_smoothed)
data$rotX_adj <- filtfilt(butterworth_filter, data$rotX_smoothed)
data$rotY_adj <- filtfilt(butterworth_filter, data$rotY_smoothed)
data$rotZ_adj <- filtfilt(butterworth_filter, data$rotZ_smoothed)

# Filter data to remove start and end errors
data <- subset(data, time_s >= input$start_time & time_s <= input$end_time)

# Find start index
start_index <- which(abs(data$accX_adj) > 1.2 | abs(data$rotZ_adj) > 1.2)[1]
```

```
    while (start_index > 1) {
      if (abs(data$accX_adj[start_index]) < 0.002 |
abs(data$rotZ_adj[start_index]) < 0.002) break
      start_index <- start_index - 1
    }

    # Find end index
    end_index <- length(data$accX_adj) - which(abs(rev(data$accX_adj)) > 1.2 |
abs(rev(data$rotZ_adj)) > 1.2)[1] + 1
    while (end_index < nrow(data)) {
      if (abs(data$accX_adj[end_index]) < 0.002 | abs(data$rotZ_adj[end_index])
< 0.002) break
      end_index <- end_index + 1
    }

    # Create final dataframe
    sensor_data <- data[start_index:end_index, ]

    # Remove unnecessary variables
    sensor_data <- subset(sensor_data, select = -c(accX, accY, accZ, rotX, rotY,
rotZ, accX_smoothed, accY_smoothed, accZ_smoothed, rotX_smoothed, rotY_smoothed,
rotZ_smoothed))

    # Rename columns
    names(sensor_data) <- gsub("_adj", "", names(sensor_data))

    # Reset row names
    rownames(sensor_data) <- NULL

    # Adjust time
    sensor_data <- mutate(sensor_data, time_s = time_s - min(time_s))

    # Add rotation direction column
    sensor_data$rotation_direction <- ifelse(sensor_data$rotZ > 0,
"Anticlockwise", "Clockwise")
```

```
    return(sensor_data)
  })

  output$raw_plot <- renderPlot({
    req(data())
    ggplot(data(), aes(x = time_s)) +
      geom_line(aes(y = accX, color = "Forward Acceleration (m/s^2)"), size =
1.5) +
      geom_line(aes(y = accZ, color = "Vertical Acceleration (m/s^2)"), size =
1.5) +
      geom_line(aes(y = rotZ, color = "Rotational Velocity (rad/s)"), size =
1.5) +
      labs(x = "Time (seconds)", y = "Value", color = "Variables") +
      scale_x_continuous(breaks = seq(0, max(data()$time_s), by = 2)) +
      theme_minimal() +
      theme(
        text = element_text(size = 14, face = "bold"),
        legend.position = "bottom"
      )
  })

  output$forward_acceleration <- renderPlot({
    req(data())
    ggplot(data(), aes(x = time_s, y = accX)) +
      geom_line(size = 1.5) +
      geom_hline(yintercept = mean(data()$accX), linetype = "dashed", color =
"red") +
      geom_point(data = data()[data()$accX == max(data()$accX), ], aes(x =
time_s, y = accX), color = "blue", size = 5, shape = 16) +
      xlab("Time (seconds)") +
      ylab("Forward Acceleration (m/s^2)") +
      ggtitle("Forward Acceleration vs Time") +
      theme(
        text = element_text(size = 14, face = "bold")
```

```
    )
  })

  output$forward_velocity <- renderPlot({
    req(data())
    data <- data()
    data <- data %>%
      mutate(velocity_x = cumsum(accX * c(0, diff(time_s))))
    ggplot(data, aes(x = time_s, y = velocity_x)) +
      geom_line(size = 1.5) +
      geom_hline(yintercept = mean(data$velocity_x), linetype = "dashed", color
= "red") +
      geom_point(data = data[data$velocity_x == max(data$velocity_x),], aes(x =
time_s, y = velocity_x), color = "blue", size = 5) +
      labs(x = "Time (s)", y = "Forward Velocity (m/s)") +
      ggtitle("Forward Velocity vs.Time") +
      theme(
        text = element_text(size = 14, face = "bold")
      )
  })

  output$distance_covered <- renderPlot({
    req(data())
    data <- data()
    data <- data %>%
      mutate(velocity_x = cumsum(accX * c(0, diff(time_s))),
             distance_covered = cumsum(velocity_x * c(0, diff(time_s))))
    ggplot(data, aes(x = time_s, y = distance_covered)) +
      geom_line(size = 1.5) +
      labs(x = "Time (s)", y = "Distance Covered") +
      ggtitle("Distance Covered vs. Time") +
      theme(
        text = element_text(size = 14, face = "bold")
      )
```

```
  })


  output$rotational_acceleration <- renderPlot({
    req(data())
    data <- data()
    data <- data %>%
      mutate(rotational_acceleration = c(0, diff(rotZ) / diff(time_s)))
    peak_index <- which.max(data$rotational_acceleration)
    peak_value <- ifelse(length(peak_index) > 0,
data$rotational_acceleration[peak_index], NA)
    peak_time <- ifelse(length(peak_index) > 0, data$time_s[peak_index], NA)
    ggplot(data, aes(x = time_s, y = rotational_acceleration)) +
      geom_line(size = 1.5) +
      geom_hline(yintercept = mean(data$rotational_acceleration, na.rm = TRUE),
linetype = "dashed", color = "red") +
      geom_point(data = data.frame(time_s = peak_time, rotational_acceleration =
peak_value), aes(x = time_s, y = rotational_acceleration), color = "blue", size
= 5) +
      # annotate("text", x = peak_time, y = peak_value, label = paste0("Peak: ",
round(peak_value, 2)), vjust = -0.5, hjust = -0.5, color = "blue") +
      xlab("Time (seconds)") +
      ylab("Rotational Acceleration (rad/s^2)") +
      ggtitle("Rotational Acceleration vs Time") +
      theme(
        text = element_text(size = 14, face = "bold")
      )
  })


  output$rotational_velocity <- renderPlot({
    req(data())
    ggplot(data(), aes(x = time_s, y = rotZ)) +
      geom_line(size = 1.5) +
      geom_hline(yintercept = mean(data()$rotZ), linetype = "dashed", color =
"red") +
```

```
      geom_point(data = data()[data()$rotZ == max(data()$rotZ), ], aes(x =
time_s, y = rotZ), color = "blue", size = 5, shape = 16) +
      xlab("Time (seconds)") +
      ylab("Rotational Velocity (rad/s)") +
      ggtitle("Rotational Velocity vs Time") +
      theme(
        text = element_text(size = 14, face = "bold")
      )
  })

  output$total_rotation <- renderPlot({
    req(data())
    ggplot(data(), aes(x = time_s, y = cumsum(rotZ) * (time_s[2] - time_s[1])))
+
      geom_line(size = 1.5) +
      xlab("Time (seconds)") +
      ylab("Total Rotation (radians)") +
      ggtitle("Total Rotation vs Time") +
      theme(
        text = element_text(size = 14, face = "bold")
      )
  })
}
# Run the application
shinyApp(ui = ui, server = server)
```