An Android-based system for frequent upper-limb functional workspace measurements to track neurological recovery

Afiq Asri 999 295 043
Cindy Bui 999 064 934
Joel Tan 999 367 046
Andrew Wong 999 066 816
Simeon Wong 998 972 615

Prepared for: Prof. José Zariffa

2015 April 29

Executive Summary

Spinal cord injuries cause a reduction in motor function, and regular limb functionality assessments are a crucial component of the rehabilitation process. Upper-limb functionality assessments are of particular interest, due to the high priority placed upon restoration of upper-limb function among tetraplegics. Currently, the frequency of these assessments is limited by the need for patients to meet with a medical professional to measure their range-of-motion and evaluate their ability to perform a standard series of upper-limb movements. The ability to acquire data from more frequent functional assessments is desired in order to personalize patient treatment. The increasingly widespread use of smartphone and smartwatch technologies and recent advances in their motion-sensing capabilities makes these devices ideal for this purpose. An Android-based self-administered system for frequent upper-limb functional workspace measurements to track neurological recovery in a non-clinical setting has been developed. The core smartwatch application interfaces with three main software and hardware modules: the device sensor module for capturing sensor input, the signals processing engine for analyzing the sensor data, and the information display library for visualizing the results of the measurement. Preliminary tests have shown that this system can be used to measure upper-limb functional workspace, but requires averaging of several trials for accurate data. Further work would involve fine tuning the signals processing algorithms and volume models to enhance the accuracy of the system. This proof of concept system demonstrates the ability for physicians to track upper-limb functional workspace recovery using commonly available consumer electronics.

Contents

Part I
Background

Clinical assessment of function is an essential aspect of neurorehabilitation following spinal cord injury (SCI). This assessment provides meaningful diagnostic information used to evaluate and guide patient treatment, and is also of particular interest in neurorehabilitation research and the development of novel treatment protocols.

SCI refers to damage to the spinal cord, which causes the loss or impairment of sensation or motor function. Common causes are trauma (car accidents, assaults, slips and falls) or disease (polio, meningitis, transverse myelitis, multiple sclerosis). There are approximately 130,000 new cases of traumatic SCI per year, and an estimated 2.5 million people are currently living with some form of the disease. Estimates for health care costs stemming from treating SCI in Canada, the United States, Australia, and the United Kingdom alone total approximately $10B a year [1]. Depending on the severity of the injury, some or all limb function can be lost. Consequences of a reduction in limb function includes a smaller range of motion, an inability to perform every day tasks, and a significant decrease in quality of life [2].

Spinal cord lesions typically result in the loss of function at and below the level of the injury, where injury at the thoracic and lumbar vertebrae levels result in paraplegia, and an injury at the cervical vertebrae level results in the more severe incidence of tetraplegia. The current project focuses on upper-limb rehabilitation, as restoration of upper-limb function is ranked as the highest priority among tetraplegics [3]. Critical to rehabilitative efforts is feedback for the clinicians on patient progress. Current standard tests for neurological status typically lack the temporal resolution to track functional recovery between physiotherapy sessions.

1 Traditional Neurorehabilitative Assessments

Currently, the recommended guidelines for clinical assessment after acute SCI involve a neurological assessment, functional outcome assessments, and an SCI-associated pain assessment [4].

1.1 Neurological Assessment

The gold standard neurological assessment is the International Standards for Neurological Classification of SCI (INSCSCI). In brief, the INSCSCI characterizes the level of the SCI based on the most caudal portion of the body with fully intact sensory and motor function in a two-step procedure [5]. A clinician assesses the patient's specific neurological injury using specific tools for that injury type, then performs a systematic neurological examination to assess sensory and motor function at the level of each spinal segment. Tactile sensation is evaluated for all 28 sensory dermatomes using both light touch and pinprick stimuli and is scored as 0 for absent, 1 for impaired, and 2 for normal. Motor function for 10 key muscles is scored on a scale ranging from 0 for total paralysis to 5 for normal strength [6]. However, the INSCSCI is not sensitive enough to detect small changes in neurological status to track progress during recovery [7].

1.2 Functional Assessments

Clinicians involved in rehabilitative therapy are often more concerned with changes in functional state as opposed to the change in neurological classification. The Spinal Cord Independence Measure (SCIM III) is the current recommended functional assessment after SCI [4], and tracks the patient's ability to independently perform 3 different categories of activities of daily living: self-care (feeding, bathing and grooming), respiration and sphincter management (independent respiration and use of toilet without assistance), and mobility (independent movement in bed, transfer to toilet) [8]. Each category is evaluated subjectively by a clinician and the sum gives the SCIM III functional score. A variant of the SCIM designed for patient self-reporting (SCIM-SR) is available and generally correlates well with the clinician-administered SCIM III [9]. Recently, the Graded Redefined Assessment of Strength, Sensibility and Prehension (GRASSP) test was developed to specifically measure upper-limb function in a highly sensitive manner to change over time [7]. Although the GRASSP has been found to correlate well with SCIM scores, the GRASSP test cannot be self-administered by the patient.

1.3 SCI-Associated Pain Assessment

Finally, pain associated with SCI is assessed using the International Spinal Cord Injury Pain Basic Data Set (ISCIPBDS). However, as pain is outside the scope of this project, we will not delve into the details of pain assessment. Interested readers may consult the ISCIPBDS document which provides information on SCI pain classifications [10].

1.4 Deficits in Current Metrics

Apart from the SCIM-SR, these evaluations must be administered by a medical professional, reducing the feasibility of frequent monitoring due to the cost to the healthcare system, medical professional availability, and transportation constraints. The patient must schedule and arrange transportation to the clinic—a non-trivial feat that is complicated by their neurological condition. Practically, professionally-administered neurorehabilitative performance inventories can only be conducted on a weekly basis, at the most frequent.

Accurate diagnosis of a patient's specific condition allows the physical therapist to tailor the therapeutic treatment to the patient. Krebs et al. found that stroke patients with similar clinical classifications can exhibit vastly different movement patterns [11], suggesting that the ideal therapeutic intervention for similarly classified SCI patients may not be identical. This lends credence to the importance of frequent data-driven diagnostics over the course of neurorehabilitation treatment. Furthermore, a widely available smartphone application (app) would enable patients to continue to monitor their improvements from at-home exercise therapy, and bring any signs of regression to the attention of their healthcare providers. Compliance with traditional prescribed exercise therapy regimens are poor, and have been found to be improved through the use of remotely monitored therapy [11], [12].

2 Measuring Range-of-Motion Using Robotics

To address the lack of both self-administered and highly sensitive upper-limb measures of neurological function, various groups have been developing robotic-based automated measures of upper-limb function. Zariffa et al. found that patient Range-of-Motion (ROM), defined as the difference between the maximum and minimum position during a task, and grip strength collected by the Armeo® Spring (Hocoma, AG) rehabilitation robot can be used to predict clinical GRASSP, SCIM, and other functional measures to a high level of accuracy [13]. Although requiring less of a clinician's time, a \$10,000+ rehabilitation robot is still prohibitive for frequent daily or twice-daily measures of neurological function within a patient's home.

3 Smartphone-Based Opportunities in Neurorehabilitation

With the recent advances in mass-market consumer electronic devices and the ever-increasing sensitivity of on-board sensors such as the accelerometer and gyroscope (together, the inertial measurement unit, IMU), there has been renewed interest in exploiting the motion-sensing capabilities of these devices for cost-effective rehabilitative diagnostics. Research groups have evaluated the use of a smartphone's built-in clinometer application for the measurement of shoulder ROM [14], [15], and knee ROM and goiniometry [16], [17], among many others. Aside from the generic tools included with the device, groups have also evaluated apps designed specifically for medical rehabilitation, such as Simple Goniometer, a specialized and publicly available goiniometry app [18]. A group from the University of Bologna designed a custom smartphone app that uses data from the on-board IMU to measure performance on the Timed Up and Go mobility diagnostic test [19].

Beyond simply measuring patient state, smartphone applications have also been used to guide rehabilitative treatments. Spina et al. used a specialized smartphone app to measure and provide real-time feedback for Chronic Obstructive Pulmonary Disease patients [20]. Training data is collected while the patient performs prescribed exercises in the presence of a medical professional by measuring exercise motion parameters using the smartphone IMU. When the patient performs the exercises independently, the app monitors the accelerometry information and provides real-time feedback and guidance to the patient based on the training data [20]. These apps demonstrate the wealth of possibilities available in consumer smartphone platforms in measurement and diagnosis for neurorehabilitation. However, none of the current apps are designed as a cohesive system for the measurement of upper-limb function.

In the past several years, the consumer electronics market has been increasingly focused on “wearables” and health-related devices and applications. A fitness band or wearable activity tracker is a wearable device that uses accelerometry data to compile fitness and exercise reports throughout the day, and recently the usage of such a device has increased in popularity. Apps running on Google’s smartwatch platform have also used the sensors on-board many smartwatches to track similar health data [21], [22], [23].

Thus, the inertial motion sensors on-board a mass-market consumer smartphone or smartwatch provide the ideal platform for frequent, self-administered ROM measurements without requiring constant clinician oversight while using commonly available and affordable hardware.

Part II
Project Description

4 Project Goal

The goal of this project is to develop an Android-based system for frequent upper-limb functional workspace measurements to track neurological recovery of patients with SCI below the C4 level in a non-clinical setting. This level of SCI was chosen as it represents the level of injury where the patient still retains some upper-limb function.

The final product is an entire system for neurorehabilitative assessment: medical professionals can create an online account for the the user, help the user install the application on their smartphone and smartwatch, and select a series of prescribed exercises. In the clinic, the physician can demonstrate the use of the app and the selected exercises, which the patient can perform on their own, guided by the app. Results would be transmitted securely to the physician on a frequent basis for review and monitoring of treatment progress and compliance. The development of this entire integrated system is much beyond the scope of this course. Here, we focus on development of the application and a single exercise protocol for upper-body functional workspace measurement.

The app will acquire motion data from the accelerometer and gyroscope in the smartwatch IMU and process this motion data to reconstruct the device's position over time in a process known as dead reckoning. The user will be asked to perform a series of exercises designed to move the user through their upper-body functional workspace while wearing the watch. The position over time at specific points in the exercise will be used to approximately measure the user's functional workspace.

Here we note that although there are examples of smartphone IMUs being used to perform dead reckoning in the literature, most of these applications involve crude pedestrian location tracking, often in conjunction with other sources of location information. Accuracies on the 2-10% have been typically reported, and are not sufficient for accurate functional workspace measurements [24]. In this project, we will attempt to mitigate many of the factors that preclude accurate position reconstruction; however, dead reckoning is inherently inaccurate and similarly, we do not realistically expect to achieve precision and accuracy on the level of clinical measurements of functional workspace in this project.

The variability in different smartphone and smartwatch hardware, especially in older devices, increases the challenges of supporting multiple devices, although the Android API makes efforts to unify the performance of apps across compatible hardware. However, to reduce the burden of testing and debugging for multiple platforms, we choose to support and target only the LG Nexus 5 and the LG G Watch as they are both one of Google's selected reference hardware devices for developers.

The app and the related exercises should be intuitive to use, even for users who may not be familiar with smartphones and smartwatches. The app and exercises should not require extensive aftermarket modifications to hardware. The user interface (UI) and exercises should be accessible to users with motor difficulties.

5 Project Requirements

The project requirements are to create a modular smartwatch application and design a standard user protocol such that:

1. A smartwatch worn by the user while performing the test protocol records and transmits IMU data wirelessly to a smartphone to calculate the user's functional workspace, defined as the volumetric region in which the user is able to control their upper-limbs;

2. Changes in the user's functional workspace can be tracked over time based on daily measurements;
3. Graphical displays can be generated to visualize these changes;
4. Logs and summary-statistics can be created and communicated securely back to a medical professional.

6 Validation and Acceptance Tests

Since clinical trials are out of the scope of the project, we will use pre-clinical testing on healthy volunteers to test the functionality of the system. Firstly, 8 volunteers will be asked to perform the developed protocol, and their functional workspace will be measured both by our system and by a manual measurement for comparative validation. The aim of this is to test how well the system performs under ideal scenarios. The app will also be compared against a robotic rehabilitation device (Armeo® Spring, Hocoma AG). The idea is to test the app's accuracy compared to currently available methods to determine the functional workspace as a proof of concept.

The validation and acceptance tests involves experimentation on human volunteers. An application for human subject-related ethical review has been submitted to the University of Toronto Office of the Vice President (Research & Innovation), and has been reviewed and approved by the Delegated Ethics Review Committee for Undergraduate Student-Initiated Project. The specifics of the study and the role of the volunteer are fully explained prior to any tests. Volunteers are asked to sign a consent form (as attached in Appendix A) prior to participation.

Part III
Work Plan

7 Work Breakdown Structure

The following components will be required to complete this project:

- App architecture and core app
- Range-of-Motion protocol
- Sensor module
- Signals processing engine
- Information display library
- Beta-testing and improvements
- Documentation for module expansions
- Release candidate testing

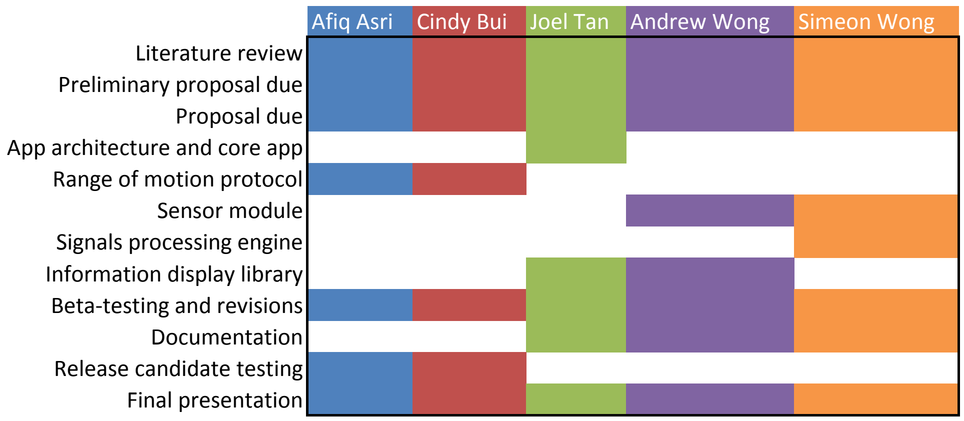
Each member of the project team will be responsible for a specific set of tasks, as per Figure 1

8 Gantt Chart

Please refer to the Gantt chart in Figure 2. Following the proposal submission in January 2015, the focus of the project will be on the sensor module and signals processing engine, which will be crucial for the function of the app. By March 2015, the only tasks remaining will be system testing, documentation drafting, and presenting the final product. The project is expected to be fully complete by April 2015.

9 Financial Plan

Please refer to the financial breakdown in Table 3. With the exception of the smartwatch, the materials needed for this project are already currently available for the team to access and use at no cost. The cost for the Hocoma Armeo® Spring is the estimated price based on [25].



	Afiq Asri	Cindy Bui	Joel Tan	Andrew Wong	Simeon Wong
Literature review	🟦	🟥	🟩	🟪	🟧
Preliminary proposal due	🟦	🟥	🟩	🟪	🟧
Proposal due	🟦	🟥	🟩	🟪	🟧
App architecture and core app			🟩		
Range of motion protocol	🟦	🟥			
Sensor module				🟪	🟧
Signals processing engine					🟧
Information display library			🟩	🟪	
Beta-testing and revisions	🟦	🟥	🟩	🟪	🟧
Documentation			🟩	🟪	🟧
Release candidate testing	🟦	🟥			
Final presentation	🟦	🟥	🟩	🟪	🟧

Figure 1. Individual team member contribution list

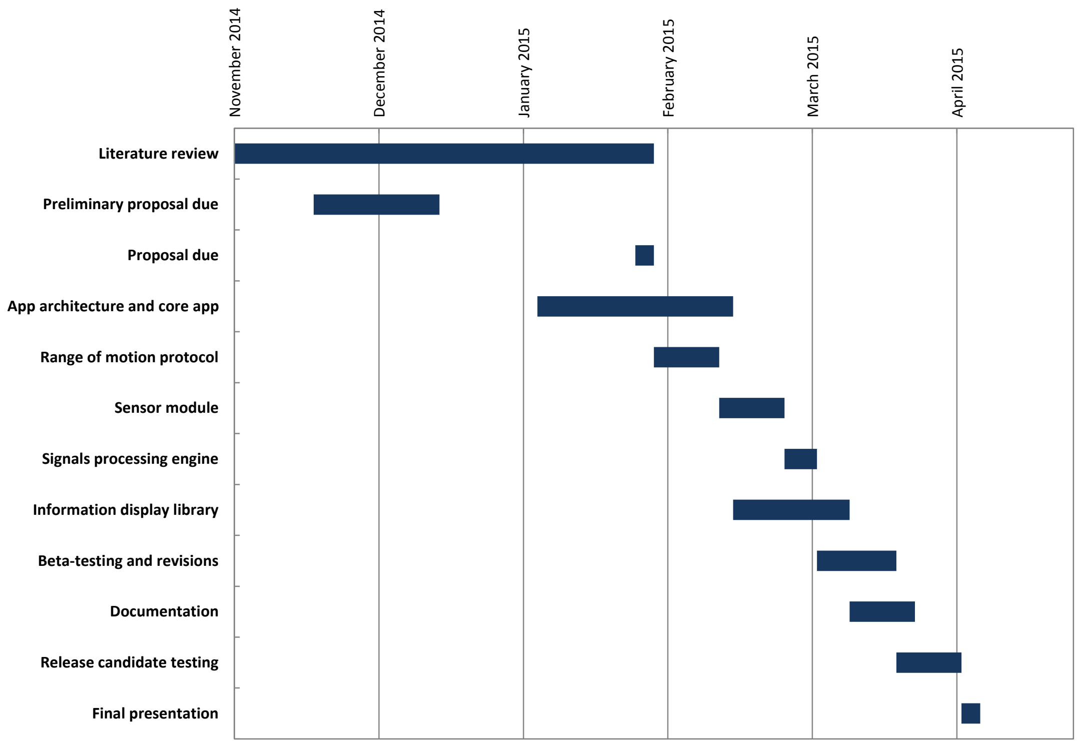
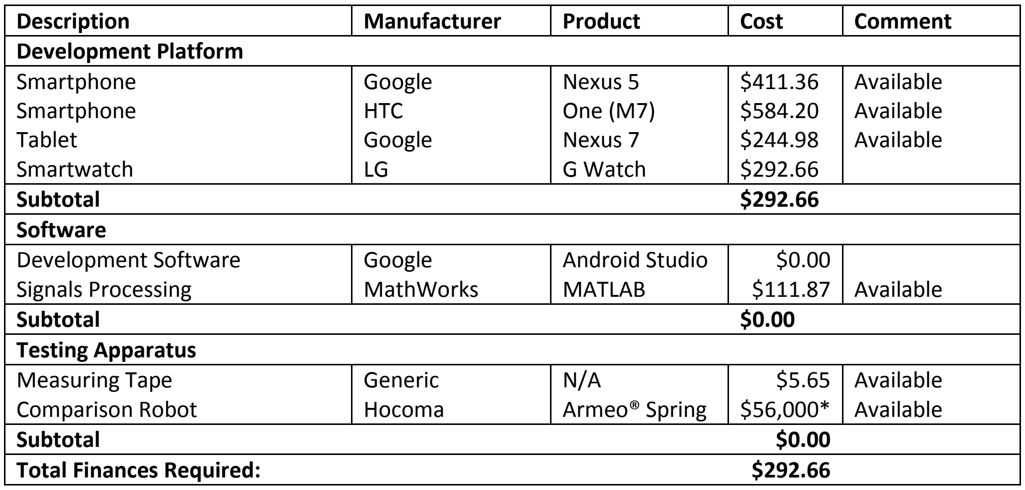


Figure 2. Gantt chart for project.



Description	**Manufacturer**	**Product**	**Cost**	**Comment**
Development Platform				
Smartphone	Google	Nexus 5	$411.36	Available
Smartphone	HTC	One (M7)	$584.20	Available
Tablet	Google	Nexus 7	$244.98	Available
Smartwatch	LG	G Watch	$292.66	
Subtotal			**$292.66**	
Software				
Development Software	Google	Android Studio	$0.00	
Signals Processing	MathWorks	MATLAB	$111.87	Available
Subtotal			**$0.00**	
Testing Apparatus				
Measuring Tape	Generic	N/A	$5.65	Available
Comparison Robot	Hocoma	Armeo® Spring	$56,000*	Available
Subtotal			**$0.00**	
Total Finances Required:			**$292.66**	

Figure 3. Financial plan for project for materials only. Prices include taxes, shipping, and other fees where applicable. * estimated based on [25].

Part IV

Technical Design

10 Proposed Solution and Alternatives

The proposed solution involves designing a system that makes use of a smartphone in conjunction with a smartwatch, as both are widely available and include an IMU which can be utilized for the purposes of this project. The use of a smartphone alone would require the user to either grasp the smartphone, which would pose a challenge for SCI patients who have weak or no hand grip functionality, or have the device attached to their upper-limbs, the weight of which may also be problematic for SCI patients and may introduce artifacts in the sensor readings. The benefits of a smartwatch are that it is lighter and designed to be secured to a person's wrist, facilitating ease of motion and accuracy of measurements. However, the relatively low processing power, lack of communication systems, and other operating system limitations prevents its use in isolation.

Devices such as computers lack the capacity to perform the measurements that we require and must be combined with some other device. Although designing a device from scratch to perform the measurements has been considered, a custom device would have additional costs and low accessibility as it cannot be readily obtained. Theoretically, functional workspace could be determined from image processing of the video footage recorded by the camera embedded in many consumer electronic devices, but this method is complicated by the lack of depth recognition in many of these devices. For this reason, image processing of functional workspace measurement will not be used for this project. As such, smartphones together with smartwatches are still presently the best solution due to their widespread availability and features.

In particular, we have chosen to design our system using Google's Android mobile operating system (Google Inc., Mountain View, CA) because of its popularity, commanding 84.4% of the smartphone market as of the third quarter of 2014 [26]. In addition, Apple's iOS mobile operating system (Apple Inc., Cupertino, CA), which has the second highest market share at 11.7% in 2014, has a more restrictive software development kit (SDK), precluding its use in this project.

The use of the FitBit Inc. (San Francisco, CA), Misfit Wearables (San Francisco, CA), Jawbone (San Francisco, CA), Nike Fuel (Nike Inc., Beaverton, OR), and Withings (Issy-les-Moulineaux, France) line of activity trackers were also considered. These devices were designed with a combination of IMU and other motion sensors to track and log a user's fitness activities throughout the day. With a specific focus on activity tracking, the IMU sensors on these devices could possibly yield superior results to a generic smartwatch platform. However, privacy policies and proprietary data interfaces with the manufacturers prevent the direct export and use of data using these platforms without significant circumvention efforts. In consideration of the client's request of an easily deployable system, these platforms were thus excluded.

10.1 Metrics for Functional Workspace

While smartphone applications have been previously developed that use IMU data to track motion and measure ROM, these apps typically measure the difference between the minimal and maximal angle positions that users are able to achieve at different joints. For the purposes of this project, we will be looking at patient upper-limb ROM through the direct measurement of their functional workspace. This functional workspace is defined as the volumetric region in space in which a person is able to comfortably reach and use their upper-limbs. Measuring ROM in this manner is more intuitive and will allow the data to be more easily applicable to both patients and medical professionals. Furthermore, a measurement for upper-limb functional workspace has been specifically requested by the client.

11 Assessment of Proposed Design

Here we propose to design and build an Android smartphone application and a companion Android Wear smartwatch application that uses the data available from the device on-board IMU sensors to measure functional workspace as the user performs a series of exercises designed to move the user through their functional workspace.

The proposed smartphone application, in conjunction with these devices, will allow for frequent functional workspace measurements, which can be used to enhance neurorehabilitation therapy.

Portability and ease of deployment. The proposed solution is a smartphone application written for the Android mobile platform, currently the most common smartphone platform [26]. The application is a packaged executable, and includes all required binaries and resources, allowing it to be easily distributed. Furthermore, the Android platform includes a “Google Play” online app marketplace and repository where the app can be uploaded for even wider distribution. Installing the app will automatically install, detect and interface with a smartwatch connected to the device, simplfying the deployment process. The app does not require any custom hardware or aftermarket modifications to the required smartphone and smartwatch.

Limitations of the on-board IMU. As a result of the application portability, the app may be installed on a variety of different smartphone and smartwatch hardware. The acquired data is also limited by the sensors available on the user's device. We note that the on-board IMU may not be accurate enough for precise measurement of the workspace. Tracking of position using IMU data, a technique known as dead reckoning, relies on highly accurate motion information (See Section 13.3 for an in-depth discussion). Due to limitations in IMUs included with consumer electronic hardware, only a rough estimate of position is possible [27], [24]. This can be mitigated by observing trends in the change in functional workspace, rather than absolute measurements. This project prioritizes an easily accessible product that can be deployed as simply as installing an app, as opposed to specialized IMU hardware.

Limitations of the measurement scope. Another weakness of this design is the lack of measurements regarding the wrist and fingers, which limits the scope to measuring the functional workspace of the arms. Grip strength and other dexterous hand movements are too complex to be measured using this method and are out of the scope of this project.

12 System-Level Overview

Although this project deals only with a single ROM protocol, the smartphone application architecture must be modular to accommodate future additions, which may have different test routines and signal processing algorithms. Thus, the current project is focused on creating a flexible core app architecture. This core app interfaces with three other modules: 1) a sensor module which interfaces the core app with either the device sensors (IMU) or external sensors; 2) a signals processing engine which contains the algorithms used to analyze the data; and 3) the information display library to present the user with the data (See Figure 4). A user protocol for this system was also developed.

This application was developed in Google's Android Studio, targeting the Google Nexus 5 smartphone and the Google Nexus 7 tablet in conjunction with the LG G Watch smartwatch. The smartphone and tablet were selected since they are Google's official reference development platforms, and due to their availability. The LG G Watch was chosen because it was the most affordable Android Wear device that featured all the necessary sensors for the project. Due to the smartwatch compatibility requirement, the app will be restricted to users using Android 4.3 (Jelly Bean) or higher, the earliest Android operating system SDK with smartwatch support. Both the smartphone

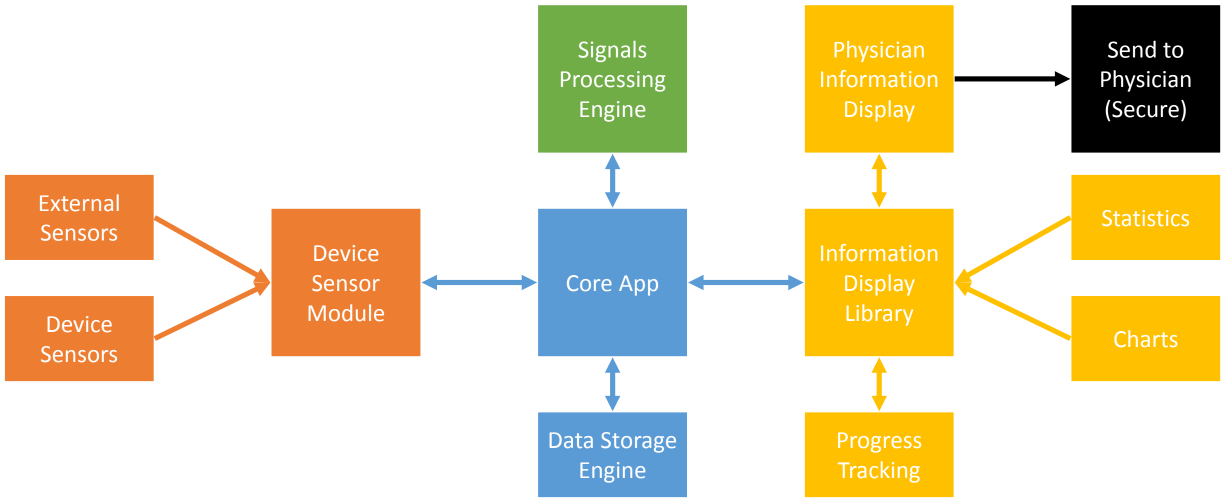


Figure 4. Flowchart of app. Different modules are labeled in different colors.

and tablet are running Android 5.0 (Lollipop) or higher. MATLAB and the Signal Processing Toolbox (Mathworks, Natick, MA) was used to design the algorithm for functional workspace measurement.

12.1 User Experience

The user interface and user experience were designed with usability in mind. Since this app was designed for users with mobility issues, the UI consists of large buttons with as few interactions as possible (see Figure 5). Tapping motions were favored over swiping motions, which are more complex and the user to move their finger across the screen in precise movements. Instead, all user interface elements are activated using taps, and actions generally have an undo button to revert accidental taps.

The protocol is shown on the screen using clear animations depicting the exercise motion. The user's health care practitioner is also expected to demonstrate the exercises in person, with the on-screen animations providing the user an intuitive prompt. The protocol is also available in textual format.

A smartwatch was chosen as a companion to the smartphone due to its size and weight. While smartphones themselves do contain the necessary IMUs required, attaching the smartphone to the user's hand may severely impair their motion. This still allows for a solution which can help enhance accuracy, while still maintaining an off-the-shelf solution.

13 Module-Level Descriptions

The role of the core application is to route information between the three system modules. The user will select the specific exercise to perform, which triggers the corresponding sensor routine, signals processing engine, and user instructions display. The app is built in a modular fashion to facilitate future expandability; here we only develop a single ROM protocol for measurement of upper-limb functional workspace.

13.1 ROM User Protocol

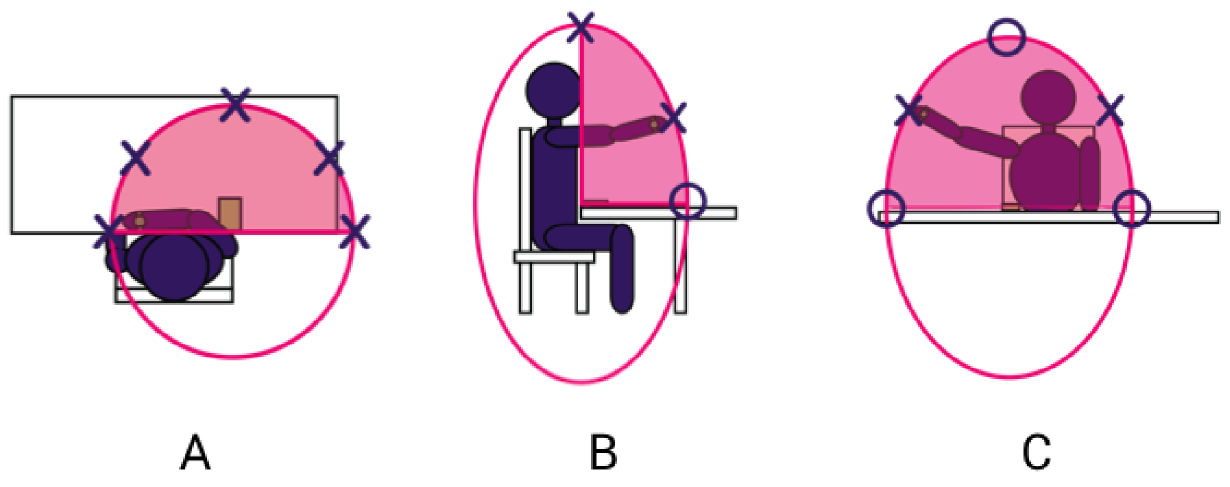


Figure 6. Views representing the different stages of the protocol. "X" represents new points on the plane which the user must reach, "O" represents points already measured from a previous step. (A) X-Y plane with 5 points. (B) Y-Z plane with 2 new points and one point from the X-Y plane. (C) X-Z plane with 2 new points, 2 points from the X-Y plane, and 1 point from the Y-Z plane. These points together give us the average radius used to calculate the volume of the functional workspace.

A user protocol was developed in conjunction with the smartphone application in order to guide the user through a 5-10min series of upper-arm reaching exercises, to be performed while wearing the smartwatch. These exercises will be done while seated at a desk or table to stabilize the torso and provide a flat surface for reference during the app measurement. The app will measure the user's reach distance as they reach for several points along the boundaries of their functional workspace. Due to the complexity of calculating the volume of an irregular surface, the distances measured are averaged to get a radius value that will be use to fit a sphere to the points reached by the user. Dividing the volume of this sphere by 4 will then represent the user's upper-limb functional workspace. Over time, the app will be able to track the changes in functional workspace based on these measurements.

The user protocol features three stages, corresponding to the three planes of motion: X-Y, Y-Z, and X-Z. In the first stage, the user reaches for 5 points in the X-Y plane (See Figure 6A). In the second and third stages, the user reaches for 2 points in the Y-Z and X-Z planes, respectively (See Figure 6B and 6C). These points map circular areas in each plane. Written instructions for the user protocol are included in Appendix B.

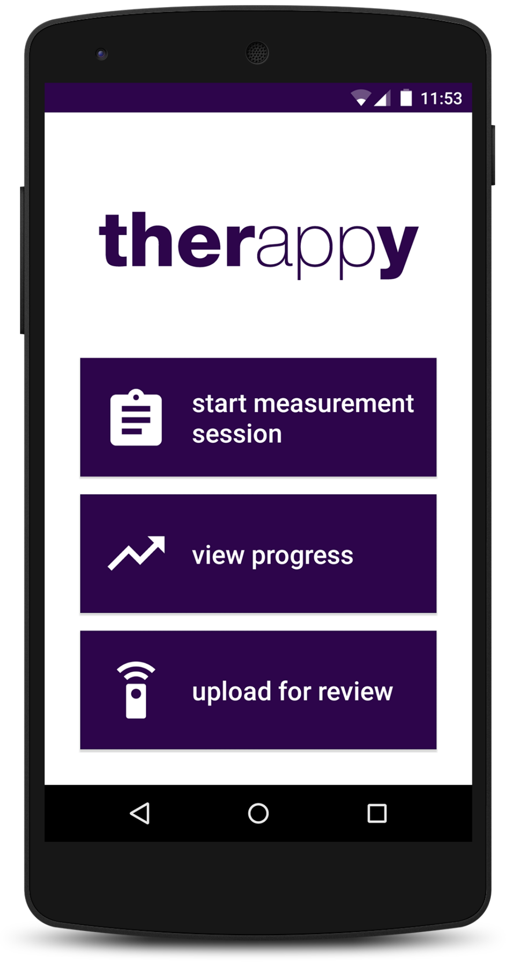


Figure 5. Main menu for the app. This screen consists of 3 large buttons for the user to interact with.

13.1.1 Protocol Assumptions

This protocol assumes:

- the rotation of each joint is independent of the other joints

- differences in reach distance for different points will adequately modify the radius value used to calculate the functional workspace
- the functional workspace includes all the points bounded by the mapped volume
- the same table and chair set-up will be used for all measurements

This protocol focuses on the upper-limb functional workspace accessible when seated at a desk since this space is important for every day tasks such as feeding, self-grooming, and interpersonal interactions.

13.2 Sensor Module

The role of this module is to interface the app with the smartwatch and/or other sensors. A variety of different sensors, including external sensors can be incorporated by reconfiguring the sensor module. The focus of this project will be on using the IMUs available on the LG G Watch. In addition, this module will be responsible for receiving IMU data, formatting the data for system cross-compatibility, and saving the formatted data. The formatted data used by the app consists of 5 entries stored in a comma-separated value (.csv) format: time (ns), type (‘a’ for acceleration, ‘n’ for next step, in its current implementation), x value, y value, and z value (units dependent on type of sensor). For example, an entry that reads: `12345,a,0.1,0.2,0.3`, describes an accelerometer reading taken at time 12345 ns, with values $\{x, y, z\} = \{0.1, 0.2, 0.3\}$ ms^{-2}. This common format is used in the signals processing engine to ensure cross-compatibility across all sensor types and signals processing algorithms.

The sensor module consists of two parts: the sensor module class in the smartphone app, and the companion app on the smartwatch.

13.2.1 Sensor Module Class

The sensor module class (SMC) contains the UI elements, as well as background I/O commands to receive and write IMU data. When the class is called from the “start measurement session” button, a loading screen appears which waits for the smartwatch to be connected. Once the smartwatch is connected, the screen will indicate to the user that the app is ready to record data. The user will begin the session with a screen tap. The SMC will signal the watch to begin collecting and sending data. Communication with the smartwatch is done using Android’s `MessageApi`. At the same time, a UI will be present on the smartwatch screen, containing information and an animation detailing the current step of the protocol. When the user has completed a step, they will tap the screen to advance to the next step. The SMC will indicate to the smartwatch that the user is proceeding to the next step. The smartwatch will then send the remaining sensor data from that step to the smartphone, and the UI will switch to the next step. At the end of the protocol, the user taps the screen, and the SMC will finish, saving any remaining data and calling the signals processing engine. See Figure 7 for a visual depiction.

13.2.2 Smartwatch Companion App

The smartwatch companion app (SCA) contains commands to get IMU data from the device sensors and send it to the smartphone. Due to the tight integration of motion data in the smartwatch for system functions, our app is not able to obtain exclusive and direct access to the onboard hardware IMU. Instead, data was acquired using the Android Sensor API. Motion data was requested from the system by registering a `SensorEventListener` for each sensor. The operating system returns data by calling our `onSensorChanged` callback method.

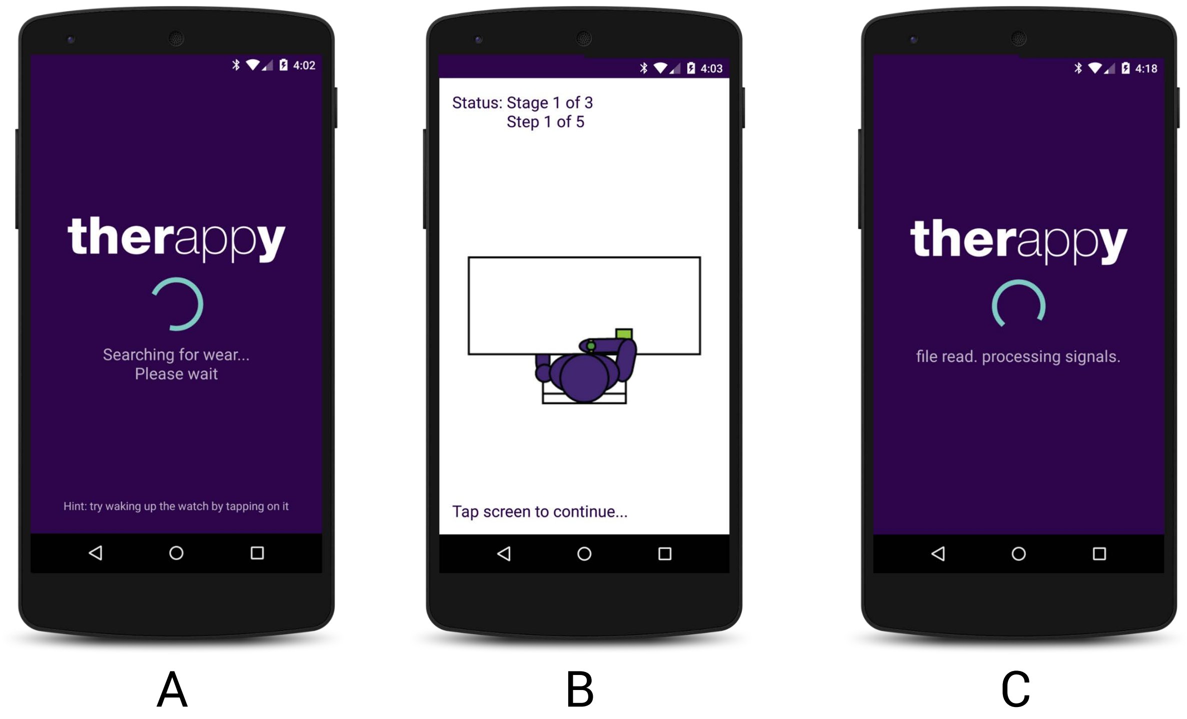

therappy
Searching for wear...
Please wait
Hint: try waking up the watch by tapping on it
A
Status: Stage 1 of 3
Step 1 of 5
Tap screen to continue...
B
therappy
file read. processing signals.
C

Figure 7. Screenshots as the user progresses through the protocol. (A) Primary loading screen to pair the smartwatch with the smartphone. (B) UI displaying the current step with animation during the protocol. (C) Screen indicating to the user that the signals processing engine is processing the results.

The SCA registers listeners for the device's linear acceleration (`TYPE_LINEAR_ACCELERATION`) and rotation vector (`TYPE_ROTATION_VECTOR`) sensors. These two sensors are virtual sensors with values constructed from a combination of hardware sensors. The linear acceleration sensor separates the measured acceleration into components due to gravity, and components due to device motion, and returns only the device motion. The rotation vector combines gyroscope, magnetometer (similar to a magnetic compass), and gravitational acceleration to determine the orientation of the device relative to the Earth. The implementation of these virtual sensors are device-dependent. On the LG G Watch, based on available information, they are implemented in hardware on the InvenSense MPU-6515 (InvenSense Inc., San Jose, CA) using proprietary MotionFusion algorithms implemented on the chip [28].

The `TYPE_LINEAR_ACCELERATION` virtual sensor returns a 3D acceleration vector in units of ms^{-2} corresponding to the Android device axes [29]. The `TYPE_ROTATION_VECTOR` virtual sensor returns the three vector components of the unit quarternion corresponding to the device rotation relative to the Earth. The reference Earth axis is defined such that the +Z axis points to the sky and is perpendicular to the ground, the +Y axis is tangential to the ground and points towards magnetic north, and the +X axis is the vector product of the Y and Z axes (the result is tangential to the ground and points east). See Figure 8 for a diagram.

However, the acceleration data received in this fashion has several issues: 1) the acceleration data is relative to the internal frame of reference, 2) the frequency in which the sensor data is accessed is not constant. The SCA corrects for change in the internal frame of reference using the rotation matrix as follows:

$$\vec{a}_{\text{real world frame}} = [\text{rotation matrix}]^{-1} \times \vec{a}_{\text{internal frame of reference}} \tag{1}$$

The rotation matrix is derived using the built-in `SensorManager.getRotationMatrixFromVector` function using the rotation vector as its argument. This sensor draws data from the accelerometer and geomagnetic physical sensors. As a result, the acceleration data sent is corrected for any rotation in the watch and all relative to the real world.

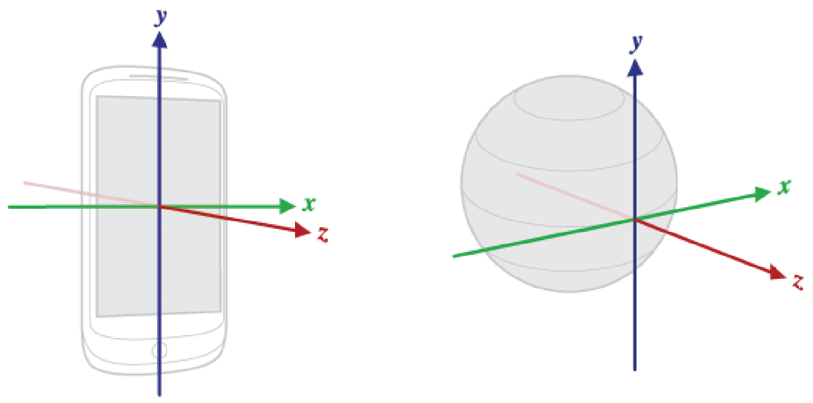


Figure 8. Schematic diagram of the device axes (left) and the Earth axes (right), where the top of the sphere is North. Images used with permission from [29].

However, SCA does not correct for data frequency. Since `onSensorChanged` only allows access of one sensor at a time, the acceleration data is corrected using the last known rotation vector. This will induce some error in the acceleration data. The signals processing engine will correct for the acceleration data frequency using upsampling and linear interpolation. The data is temporarily stored in a 64-event `ByteBuffer` before being sent to the phone using the `MessageApi`.

13.3 Signals Processing Engine

The role of the signal processing engine is to analyze and process the formatted data saved by the sensor module to output the calculated functional workspace volume and the associated 2D area components. These outputs will be used by the information display library to produce graphs and summary statistics. The algorithms employed by the signal processing engine are specific to each test protocol and measurement to be performed. Furthermore, it allows for future expansion, including external processing of the sensor data for more complex analyses. Although outside the scope of this project, this module can be expanded to calculate other metrics aside from upper-limb functional workspace, such as lower-limb functional workspace and stability.

13.3.1 Theory

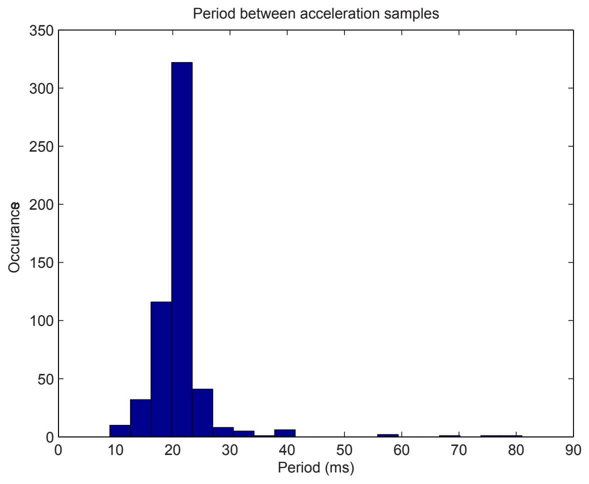
For the measurement of functional workspace, the signals processing engine will calculate the volume of the user's functional workspace based on the distance traveled by the smartwatch using a process known as dead reckoning. This involves calculating the current position based on double integration of acceleration data relative to an initial position. The calculation is as follows:

$$\vec{x}(t) = \iint_t \vec{a}(t)\mathrm{d}t \tag{2}$$

However, given that the data comes in at discrete time points, the values which are recorded are the instantaneous acceleration in the period $[t, t + \Delta t]$. As such, the equation becomes:

$$\vec{x}(t_i) = \sum \Big(\sum \vec{a}(t_i) \cdot \Delta t_i \Big) \Delta t_j \tag{3}$$

The issue with this method is the accumulation of error, also known as integration drift. Dead reckoning uses the previously calculated position to estimate the new position. At each time point, the calculated change in position comes with some error, $\Delta(t)$. Since each new position is based on the previous position, which itself comes with some error, $\Delta'(t)$, this error is accumulated over time. With accelerometry data, this error is integrated twice: once



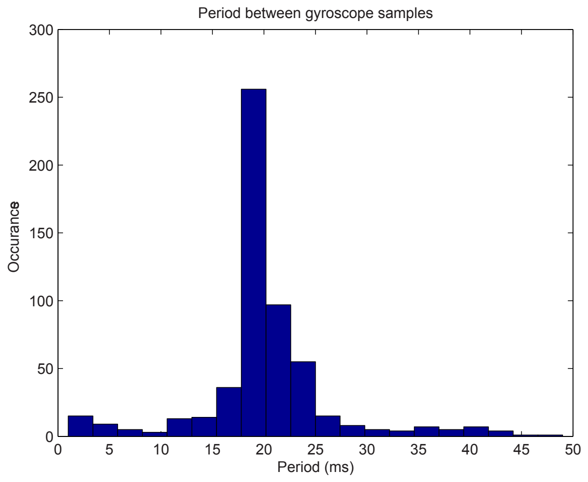


Figure 9. Histogram showing distribution of periods between subsequent acceleration and gyroscope data samples returned by Android Sensor API in a representative recording session. The standard deviation of the periods is typically around 20 ms.

to calculate the change in velocity, and once again to calculate the new position. This is a known limitation of dead reckoning using IMUs.

13.3.2 Confounding Factors

In practice, there are multiple confounding variables that complicate the dead reckoning algorithm and processes. In the following, we describe the major confounders we encountered and discuss our methods for mitigation.

The methods are not perfect. The app will be unable to precisely calculate the functional workspace with single measurements. Current dead reckoning systems that are implemented on smartphones are primarily used in pedestrian localization systems that do not require the same level of accuracy [27], [24]. These systems also draw information from other localization methods such as indoor WiFi signal strengths, Bluetooth beacons, and other fixed markers. More complex systems, such as the ones used in aircrafts or satellites, use more precise and costly IMUs, and additional, more accurate sensors such as GPS to correct for the drift. However, using methods for mitigating the following errors, and by looking at trends in the data rather than the absolute functional workspace volume, the accuracy can be improved greatly.

The sensor is imprecise and noisy. The data received from the Android sensor API is noisy even when the device is at rest. The IMU in the device uses microelectromechanical systems (MEMS) based hardware sensors, and due to their size and associated circuitry, are non-trivially affected by thermal noise. Although high-frequency noise is reduced during the integration step, the noise results in slight accumulated errors in the integrated values that have a large impact over time due to integration drift.

The returned data has jitter. Data is returned from the Android sensor API to a function of our choosing asynchronously when the requested sensor has recorded a new value. The approximate interval between returned values can be set when initializing the IMU sensors, but the interval is far from consistent. A histogram of the period between sampled acceleration and gyroscope values is presented in Figure 9, with standard deviations of approximately 20 ms. This is unacceptably large if the data is to be filtered. Data may also be delayed, returned out of order, or multiple values returned in quick succession depending on current processor load and other scheduling factors.

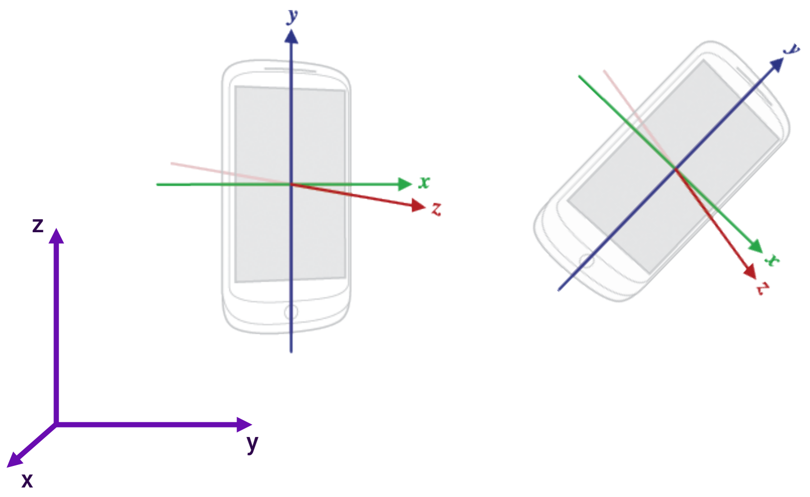


Figure 10. Illustration of phone rotation changing the relationship between the measurement axis and the fixed external reference frame of interest. Device axis images used with permission from [29].

Acceleration due to gravity. Any accelerometer, by definition, measures the net acceleration it experiences. It is impossible, physically, to distinguish between the acceleration due to gravity and the acceleration from device movement relative to the Earth's reference frame. Gravity is a tremendous issue for accurate localization using double integration of measured acceleration. Gravity has magnitude 9.81 ms^{-2}, compared to acceleration from sudden motion peaking at approximately 5 ms^{-2}.

Methods for accounting for gravity include high-pass filtering, estimating device orientation and subtracting the gravity vector, and independent component analysis (ICA). If the device is kept at a constant orientation, and moved only in a single plane orthogonal to the gravitational acceleration, all three of these methods work acceptably well. [30] details another possible method to account for this error. However, this method does not completely remove the gravitational error to the level of accuracy demanded for our purposes.

Our selected solution to removing the gravitational component of acceleration uses the `TYPE_LINEAR_ACCELERATION` virtual sensor, where gravity is approximately removed at a hardware level on the LG G Watch's IMU chip. A 3 sec moving average is subtracted from the data to remove the remaining bias. A more in-depth discussion on the hardware virtual sensors is presented in 16.4.1.

Device rotation. It is difficult, if not impossible, to design a functional workspace measurement protocol that both keeps the hand in the same orientation, and moves the user through the extent of their functional workspace. However, the rotation of the acceleration sensor poses some problems. The sensor is only capable of measuring acceleration in its own intrinsic frame of reference, but the motion we concern ourselves with is the Earth's external reference frame. If the device begins motion in a particular orientation, and ends with a different orientation, the acceleration will occur in a different axis than the de-acceleration (Figure 10). Practically, the de-acceleration will be reconstructed as the device accelerating in yet another direction.

Gravity in combination with device rotation, poses an even larger problem: it is a constant and large source of acceleration, and as the device rotates, the component of gravity in each accelerometer axis changes. Since gravitational acceleration is much larger than motion acceleration, it can easily overwhelm the signal. Furthermore, the change in axis of gravity typically occurs over an interval of seconds, and is thus recorded as a high-amplitude, low-frequency noise that is not easily filtered without removing the bulk of the signal.

Our selected solution uses the `TYPE_ROTATION` virtual sensor to track device rotation at a hardware level on the LG G Watch. The acceleration vector is transformed from the device axis to the Earth's axis based on the rotation vector.

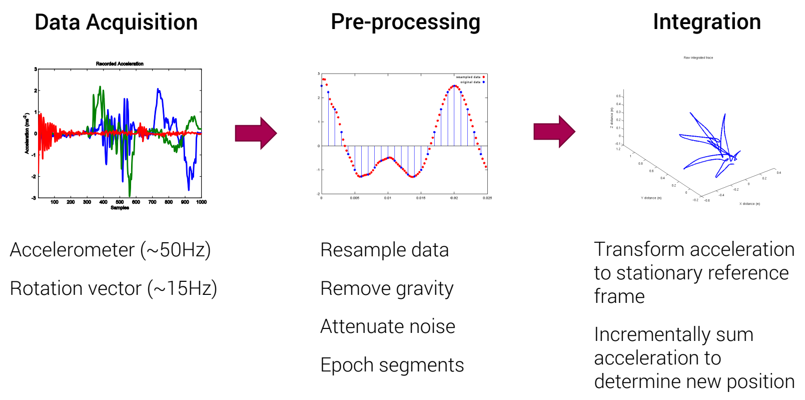


Figure 11. Schematic representation of the signals processing pipeline.

13.3.3 Final Signal Processing Methods

The varied nature of the confounding factors requires a multifaceted approach to analysis of the recorded data to reconstruct the position values. The algorithm for signals processing was developed using MATLAB R2014b (Mathworks Inc., Natick, MA) software, and subsequently rewritten in Java using the Apache Commons math library (Apache Software Foundation, Forest Hill, MD) for Java. The code is provided in Appendix C. Figure 11 gives a schematic representation of the signals processing pipeline.

Correcting for rotation. Rotation of the device is tracked using the Android `TYPE_ROTATION_VECTOR` sensor. It's implementation is device and manufacturer specific. In general, the sensor uses a combination of gyroscope, magnetometer, and long-term average acceleration to track the orientation of the device in space. The gyroscope provides rapid updates with measurements of angular velocity. The magnetometer orients the device with respect to the Earth's magnetic field, and the long-term average acceleration yields the direction of gravity over time to account for integration drift of the gyroscope data.

The rotation of the device is returned in much the same way as the acceleration data. During data acquisition, acceleration values are buffered in memory. As a new rotation vector is made available from the system, the rotation at each buffered acceleration value is linearly interpolated from the current and previous rotation vector. The rotation matrix is calculated using the `SensorManager.getRotationMatrixFromVector` Android API call and multiplied with the acceleration vector to rotate it into the extrinsic frame. The time in nanoseconds at which the acceleration was sampled and their rotated values are stored in a comma-separated values (CSV) format and the acceleration buffer is flushed.

Resampling. After data acquisition, the data is loaded into the signals processing engine and sorted by time. To apply filtering and other conventional signal processing techniques to the acceleration data, it must be resampled to a constant rate to eliminate the jitter. The data is upsampled to approximately 5x the mean sampling frequency to preserve resolution. Since an FFT/iFFT based filter is used to filter the data, and is most computationally efficient with data of length 2^n, the data is resampled to a length of 2^n which most closely approximates the target 5x upsampling.

Filtering. The resampled data is then filtered using an FFT-based filter. The data is transformed into the frequency domain and undesired frequencies are set to zero, then the data is transformed back into the time domain. A variety of other filtering methods were tested but none were capable of selectively removing DC signal components as well as the FFT-based filter without also removing critical low-frequency acceleration information. The data is band-pass filtered, removing the DC and frequencies >25 Hz.

The development of the filtering step occurred relatively early in the overall development of the signals processing pipeline. Due to time constraints, we were not able to further refine the filtering step to account for other enhancements in the pipeline. We discuss this further in 16.4.1.

Subtracting the moving average. Even with the removal of DC components, we noticed the acceleration data still exhibited local DC bias. Due to integration drift, these biases could result in inaccuracies on the order of 10s of meters. The most effective way to remove these biases was to subtract the moving average of the data over a centered window of approximately 3 sec from each time point. Shorter time intervals were influenced too greatly by the fluctuations in acceleration due to device motion, and longer time intervals were not capable of removing the local DC biases.

Integrating. Finally, the processed acceleration data is doubly integrated to reconstruct the position of the device over time as the user performed the protocol, as above.

Accounting for integration drift. In order to reduce integration drift, the position of the smartwatch is reset to the origin at the beginning of each step. Due to integration drift, the reconstructed position is often not at the origin. Since the protocol is designed such that the smartwatch begins and ends at the origin, the reconstructed position can be adjusted under this assumption to theoretically correct for the integration drift in the reconstruction during the protocol.

13.3.4 Functional Workspace Calculation

Using the reconstructed position of the smartwatch during the measurement protocol, the user's functional workspace can be approximated. The furthest position of the device during each step of the protocol is measured, and averaged for each of the three planes to calculate the functional workspace area in each respective plane, and averaged across all the measurement points to calculate the functional workspace volume. The average radius is then used to determine the area of a circle (Equation 4) or volume of a sphere (Equation 5).

$$A_{\text{approx}} = \frac{1}{2}\pi r_{\text{avg}}^2 \tag{4}$$

$$V_{\text{approx}} = \frac{1}{3}\pi r_{\text{avg}}^3 \tag{5}$$

More mathematically complex methods were also tested, including fitting multiple spheres or ellipses to the position over time. However, these methods are comparable to the results obtained through the use of mean max displacement as the radius for a single sphere. As a standardized set of specific volume measures for functional workspace have not yet been reported in the literature, thus rendering comparison or equivalence moot, we decided that the absolute volume of the functional workspace is not critical to the goals of this application. Rather, we would like to observe changes in functional workspace over time. This is adequately summarized by the proposed measures above. In light of the additional complexity, and the lack of utility of more advanced functional workspace metrics, the rough approximation yielded by a single sphere model is deemed the superior option.

13.4 Information Display Library

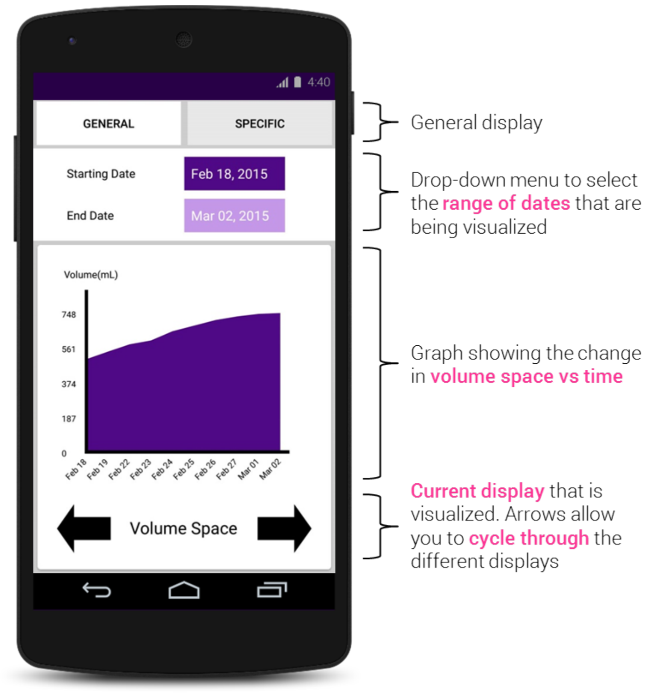
The role of this module is to display summary statistics and trends in the user's functional workspace. The information display library will receive processed data from the signals processing engine, render the visualizations, then return the images to the app to be displayed to the user. The information display interface itself was designed to be purely touch-based. This is because ordinary swiping, which is more commonly used to navigate between pages on touch-enabled devices, is a more complex motion which the user might not be able to perform if they are recovering from an SCI.

The visualization display separates the information displayed into two categories, general and specific information. The aim of the general information category is to provide quick and easy-to-understand information regarding the user progress. The data that will be presented in this category includes the change in functional workspace volume as well as the functional workspace area in the 3 different planes, all with respect to time. For the specific information category, the data presented will include the actual measurements made with the protocol, which will allow the user to better compare and identify strengths and weaknesses among the specific planes and movement directions within the planes.

General information. There are four different sections that make up the general category, namely the volume space, X-Y Space, X-Z Space, and the Y-Z Space. The volume space shows the change in functional workspace volume of the user over time, which allows the user to track their general progress and to see if they have been improving as a whole. The other 3 spaces represent the different planes of motion that were measured using the protocol. These spaces will allow the user to further break down their improvement in a specific plane and allow for the focusing on a specific motion protocol to emphasize improvement for a specific upper arm motion which has shown a lack of progress. An example of the general information layout as displayed on the phone is shown in Figure 12.

Specific information. The first page of the specific section is the summary page which allows the user to identify his/her progress in all the different measured areas at a glance. This summary page is shown in Figure 13. The next 3 pages of the specific section gives the actual measurements made with the protocol, which allow the user to identify his/her actual motions in the 3 different planes that were measured. An example of the visualization of the motions made by the user are shown in Figure 14. Here, the 5 different points that were recorded by the user for one particular plane are shown.

Send to physician. This module will also be able to generate reports for physicians and other medical professionals. At this stage, the "Send to Physician" feature is only expected to function as a basic transfer of information. As above, full functionality and secure information transfer is outside the scope of this project.



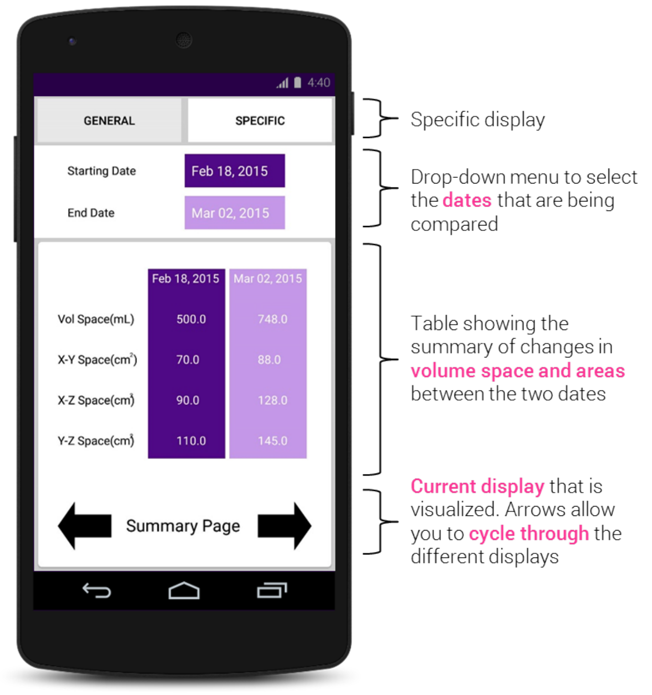
General display

Drop-down menu to select the **range of dates** that are being visualized

Graph showing the change in **volume space vs time**

Current display that is visualized. Arrows allow you to **cycle through** the different displays

Figure 12. Function workspace volume changes within a specified range of time.



	Feb 18, 2015	Mar 02, 2015
Vol Space(mL)	500.0	748.0
X-Y Space(cm^2)	70.0	88.0
X-Z Space(cm^2)	90.0	128.0
Y-Z Space(cm^2)	110.0	145.0

Specific display

Drop-down menu to select the **dates** that are being compared

Table showing the summary of changes in **volume space and areas** between the two dates

Current display that is visualized. Arrows allow you to **cycle through** the different displays

Figure 13. Summary of workspace changes between two chosen dates.

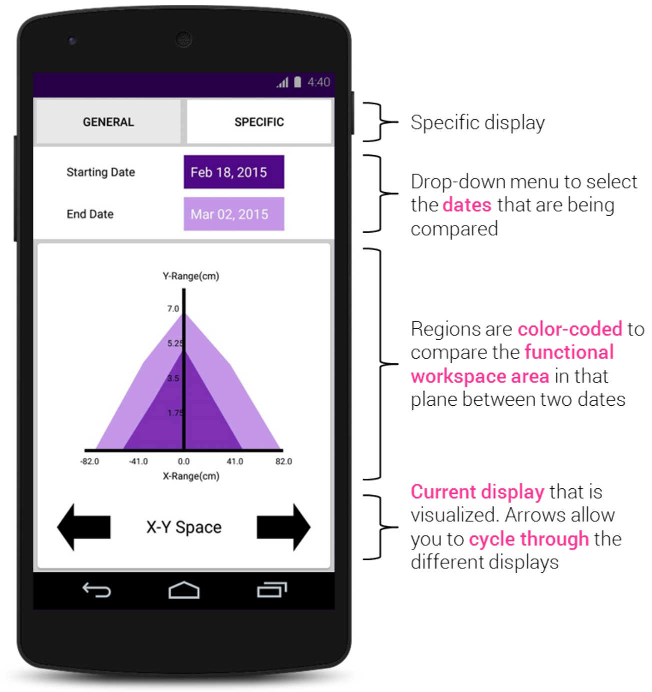

4:40
GENERAL
SPECIFIC
Starting Date
Feb 18, 2015
End Date
Mar 02, 2015
Y-Range(cm)
7.0
5.25
3.5
1.75
-82.0
-41.0
0.0
41.0
82.0
X-Range(cm)
X-Y Space
Specific display
Drop-down menu to select the dates that are being compared
Regions are color-coded to compare the functional workspace area in that plane between two dates
Current display that is visualized. Arrows allow you to cycle through the different displays

Figure 14. Visualization of the motions performed by the user for a particular plane between two chosen dates.

Part V

Testing, Results, and Discussion

14 Simulated Results

The development and testing of the signals processing engine (SPE) component of the app occurred concurrently with development of the protocol. Thus, a set of accelerometer measurements was recorded for the purposes of evaluating iterations of the SPE.

Cardboard testing track. In brief, a cardboard track and device holder was constructed to snugly fit an HTC One M7 Google Play Edition smartphone (HTC Corporation, Taipei City, Taiwan). The cardboard track guided precise and smooth one-dimensional movement of the smartphone over a length of 50 cm. Acceleration data was acquired using MATLAB Mobile for Android (Mathworks Inc.) available on the Google Play App store and imported directly into MATLAB.

The phone began at one end of the track. Data acquisition was started. Approximately 3 seconds after the beginning of the acquisition, the phone was moved to the opposing end of the track. The data acquisition was terminated at least 3 seconds after reaching the end of the track. Recordings were repeated multiple times, and were performed along each of the axes of the device as shown in Figure 8. A representative recording of motion along the y-axis is shown in Figure 15.

Visually, the acceleration data appears noisy in all three axes during motion. Initial attempts yielded position results that visually resemble the traversed path. However the numerical values for position differed from the intended position by over 300% in axis of motion, and 40 cm in axes with no motion. Qualitative tests with smartphone held in the hand showed less noise. These errors could be attributed to many confounding factors, discussed in more detail in 13.3.3. The performance of each iteration of the signal processing engine were evaluated using this test data set.

Handheld tests. In later iterations of the design, when the app was ready to collect data, a new set of sample data was acquired in a similar fashion using the new methods. In brief, a custom version of the app was built, collecting acceleration data using the `TYPE_LINEAR_ACCELERATION` virtual sensor. Since the device traveled solely in one axis in a constant orientation, device rotation was not necessary.

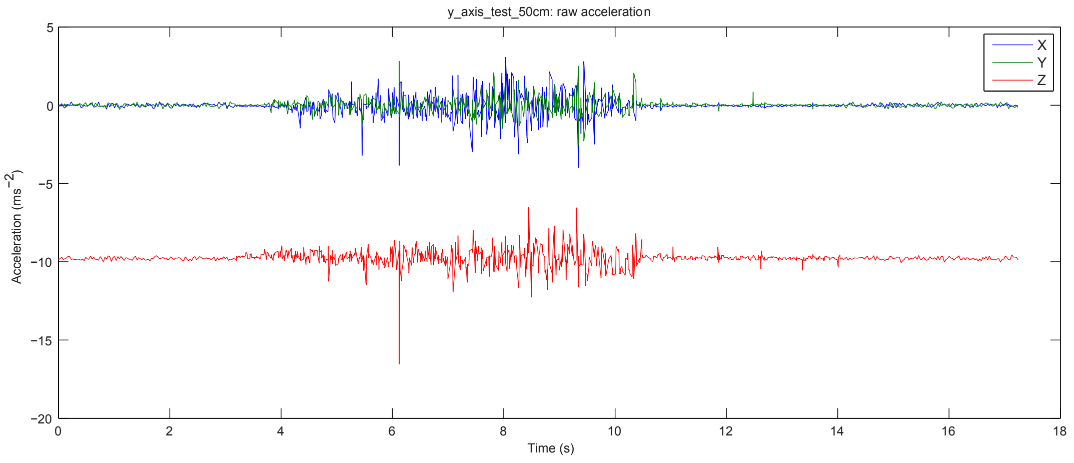


Figure 15. Acceleration values over time captured during a representative test track movement along the y-axis.

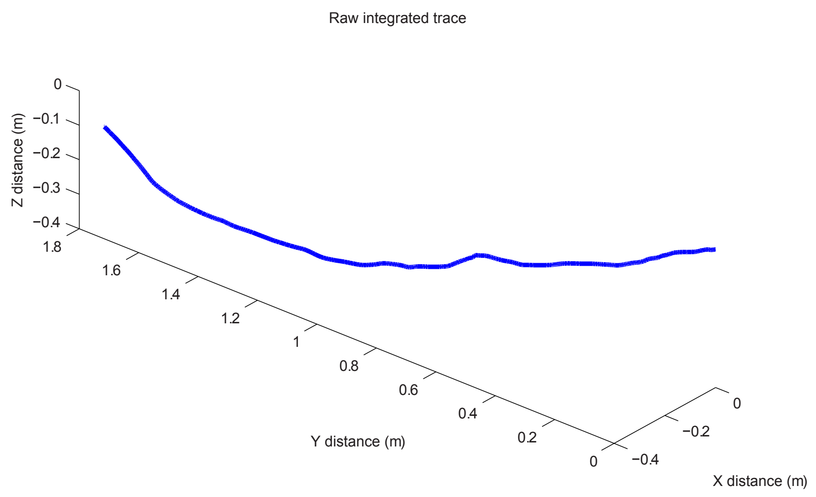


Figure 16. Reconstructed motion from a representative recording of one-directional motion show initial attempt at reconstructing motion along a fixed 50 cm track exhibit errors on the order of 10s of centimeters in all three dimensions.

Motion Direction	+X	+Y	+Z	+X/+Y	+Y/+Z	+Z/+X
Measured Distance	43.6 ± 1.7	45.8 ± 0.9	46.1 ± 3.6	51.2 ± 2.6	45.5 ± 4.6	59.0 ± 8.2

Table 1. The reconstructed change in position using the final signals processing engine from recordings of the smartphone being moved 50 cm along a straight line. The device orientation corresponds to movement of the device in the direction of the given axis as shown in Figure 8. The combination of axes denotes that the device is moved such that it's direction forms a 45 degree angle between the given axes directions.

In these tests, the decision was made to hold the smartphone in the hand due to the noise in the acceleration during movement recorded from the test track (Figure 15).The smartphone was held in the hand above the test track with no contact with external surfaces. Acceleration data was acquired as the smartphone was moved by hand from one end of the test track to the other with 3 sec of data on either end of the motion. The smartphone was held in a constant orientation (ie. not rotated). This was repeated with motion along the other phone axes.

Many variants of the signals processing module were tested over the course of the project using this data set. For the sake of brevity, we only provide the results obtained from the final signals processing engine in Table 1. The signals processing engine accurately reconstructs the motion in a single axes with a precision of up to <1 cm. The variation in precision between the three axes may be due to the errors introduced by the linear acceleration algorithms on the IMU chip. Motion recorded in two axes were less precise, possibly due to device rotation during movement as rotation data was not used in this case, or the lower signal-to-noise due to the acceleration signal being divided between two axes.

15 Validation Results

To test and verify our system, we performed the protocol on 8 volunteers using the app. The 8 users can be split into two separate groups, with one group (Users 1-4) performing 3 sets of 5 measurements each, while a second group (Users 5-8) performed only 1 set of 5 measurements. Additionally, 3 of the volunteers also measured their functional workspace using a robotic rehabilitation device for comparison. The aspects that we studied are described in detail in the following sections.

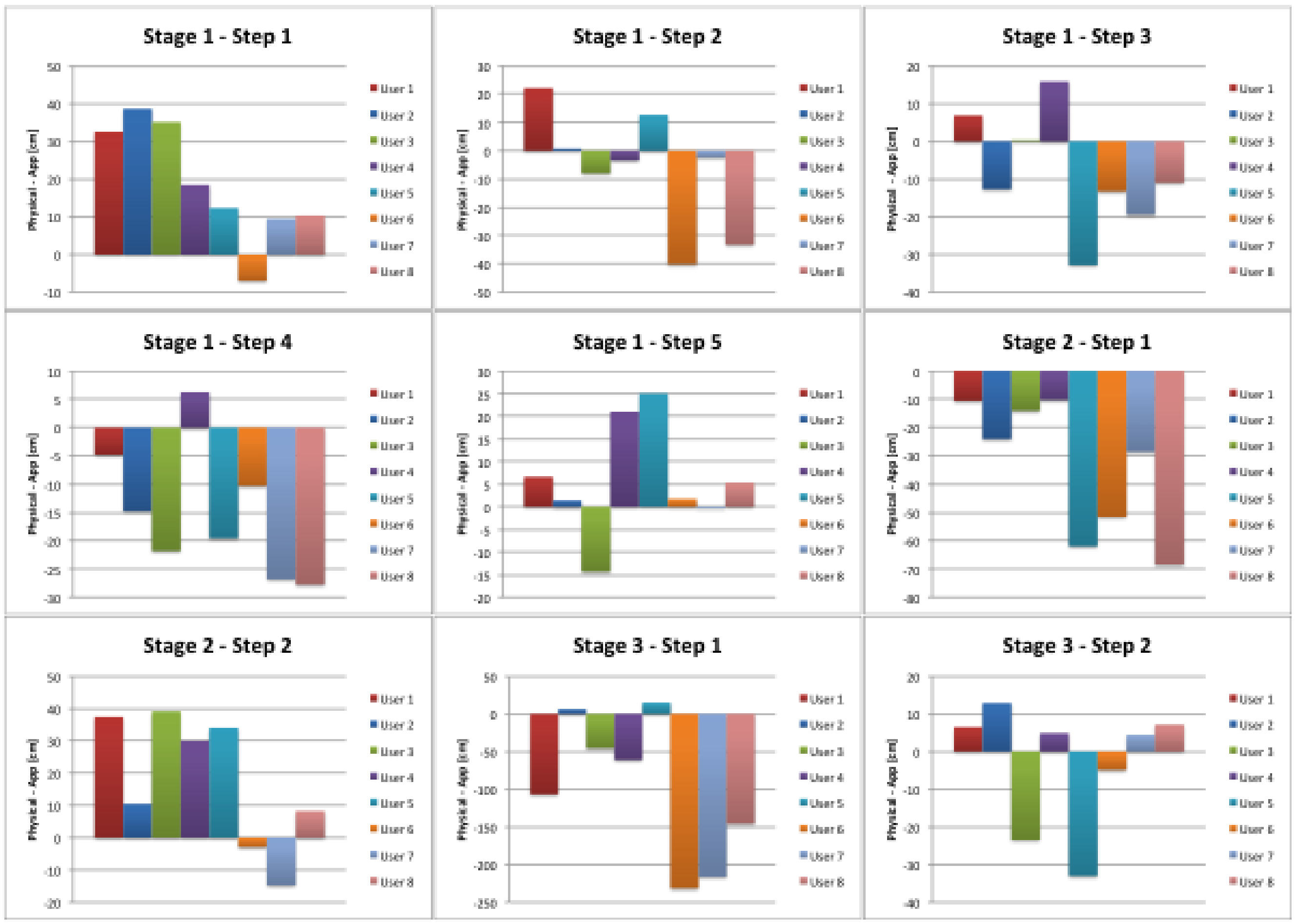


Figure 17. Differences between the reach distance measured physically and by the app for each volunteer user, separated by protocol step.

Reach distance: app vs. physical measurements. To start off, we compared the reach distances for the 8 volunteers, measured by the app and measured physically using measuring tape. During one of the app measurements for each volunteer, the user was asked to pause at their maximum extension in each step for a physical measurement of the distance between the smartphone and the smartwatch. The differences between the distance measured physically and by the app for each user are shown for each step in Figure 17. The average difference between these two measurement methods for each step among the 8 users is shown in Figure 18.

In general, the app overestimates or underestimates the reach distance at every step. From the single measurement, the overall average error per step is approximately -13cm. We expect that effect of these differences will be reduced by averaging of the different steps to generate a representative distance value and performing multiple measurements. The results also showed a large difference between the app calculated values and the physically measured values for Stage 3, Step 1. This suggests that there is a systematic error in measuring that particular step in the protocol, although it is currently unclear as to why this is so. A postulation was that this particular step in the protocol involved a hard-to-reach place (with a right hand, this involves moving to the upper left corner across your body), which may have resulted in the deviations as the user attempts to move their arm in that direction. However, this is most likely only a contributing reason, as the average error is just under a meter, which is much greater than expected from any deviation that could occur during the protocol.

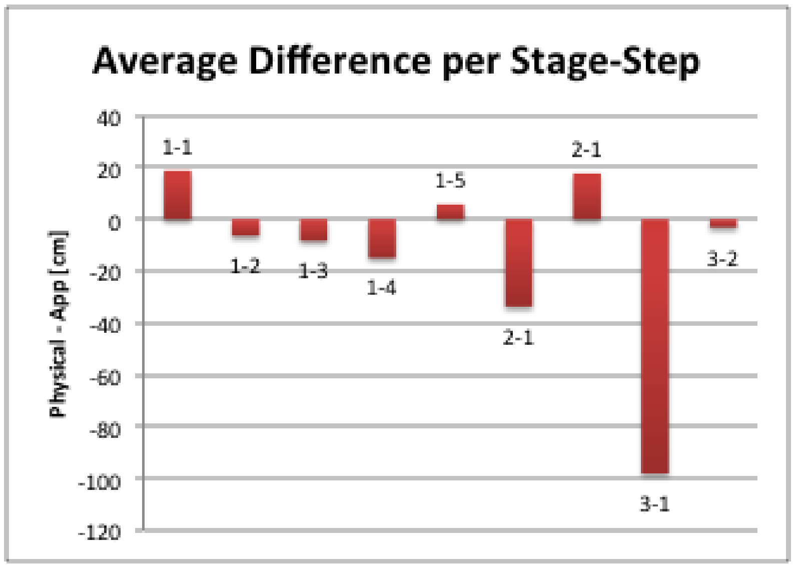


Figure 18. Average differences between reach distance measured physically and by the app for 8 volunteer users for each step of the protocol.

Functional workspace volume: app vs. physical measurements. After identifying the accuracy of the individual positions that were mapped out by the app, we proceeded to calculate the functional workspace volume of the user. Since the value measured by the app for Stage 3, Step 1 had significant errors and appeared to be a significant outlier for most of the 8 users as shown in Figure 18, this value was excluded from the calculations. The volume calculated by the app was compared to the volume generated by using the average of the 9 points measured physically as the radius for a quarter-sphere. Both volumes were calculated using Equation 6. Note that Equation 6 is equivalent to Equation 5 as the same model is used

$$V_{\text{phys. meas.}} = \frac{1}{3}\pi l^3 \tag{6}$$

where l is the average radius measured/calculated. The results show that our app was capable of accurately matching the physical volume measurements for 2 out of 8 of the users, both coming from the group that performed 3 sets of measurements. In general, the performance of the app was better for users who performed a higher number of measurements, which suggests that a more accurate result could be obtained if more measurements were made. The results are summarized in the graph shown in Figure 19.

Functional workspace volume vs. limb length. To further validate the accuracy of the app, a graph plotting user functional workspace volume against user upper arm length was generated. It is expected that longer limb lengths would be correlated with larger functional workspace volumes, as per our model (see Equation 6). However, the results did not show a good correlation between these two values. A possible explanation for this is that some of the users may not have been reaching as far as they could during the protocol. Another potential source of error is that the user may have moved their torso during the protocol, which will alter the distance recorded by the app. The results are shown in Figure 20.

Functional workspace volume: app vs. Armeo® Spring measurements. Lastly, we compared the functional workspace volume calculated by our app with the volume measured by the Armeo® Spring robotic rehabilitation device. Since the Armeo® Spring uses a rectangular prism to estimate the functional workspace volume, the Equation 6 is not valid. As such, it can be modified to Equation 7.

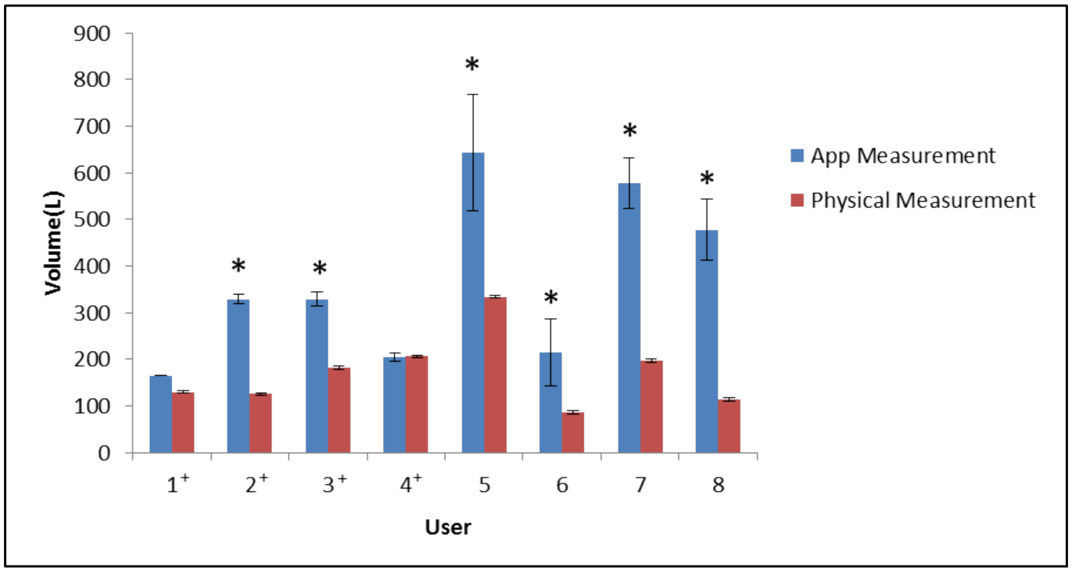


Figure 19. Functional volume workspace calculated by the app vs the volume calculated by physical measurements. * $p < 0.05$ compared to physical measurement. $^+$ users that performed 3 sets of 5 measurements each as opposed to just 1 set of 5 measurements.

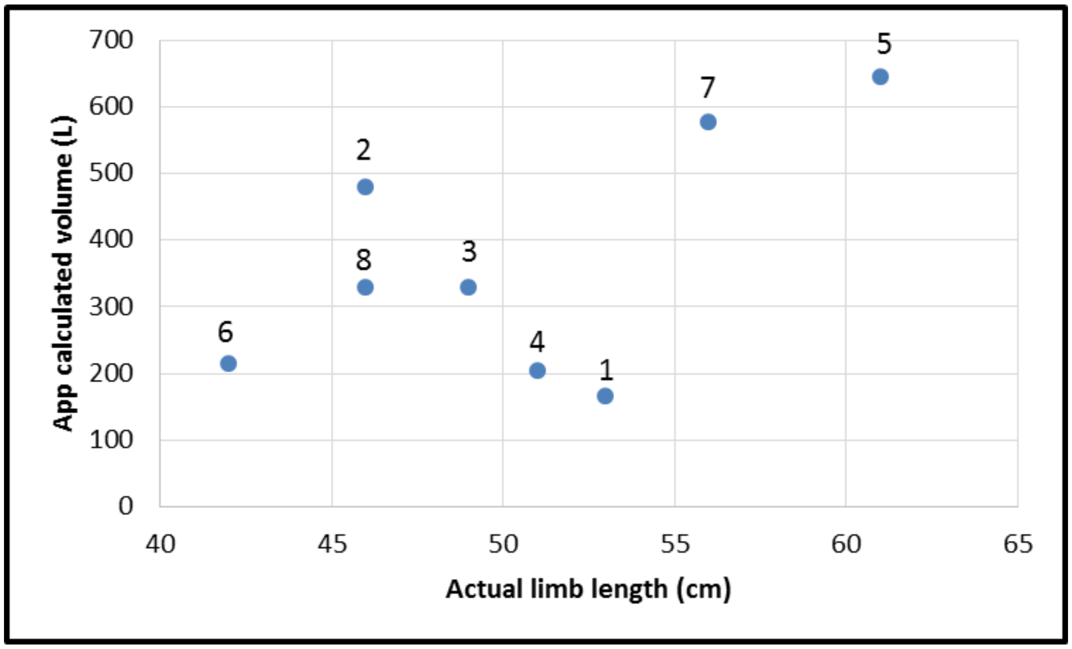


Figure 20. Functional volume workspace calculated by the app vs the upper arm length of the user. The upper arm length was measured from the shoulder joint to the wrist. Note that the expected trend is a cubic based on Equation 5, which was not seen in these results.

Subject Number	App calculation	Physical measurements	Armeo Spring
1	174.2	130.0	175.8
3	359.2	181.7	166.9
4	138.9	206.7	174.7

Table 2. Comparison between different methods of measuring functional workspace volume. Volume reported in litres (L). Physical measurements are calculated using Equation 6, and Armeo® Spring by Equation 7.

$$V_{\text{robot}} = |x_{\max} - x_{\min}| \times |y_{\max} - y_{\min}| \times |z_{\max} - z_{\min}| \tag{7}$$

Note that this estimation will result in an over-approximation of the functional workspace. This however, may be balanced by the fact that the Armeo® Spring restricts the user's range of motion during the measurement. By contrast, the app allows for free range of motion. The results show that the functional workspace volume measured by the app was similar to the that measured by the Armeo® Spring for Users 1 and 4. However, the volume calculated by the app for User 3 was more than two times greater than that measured by the Armeo® Spring. This highlights the erratic nature of the app, where it does not consistently generate accurate functional workspace volumes. As expected, the volumes calculated by the physical measurements and the volumes measured by the Armeo® Spring are generally in good agreement with each other. The results obtained are summarized in Table 2. Note that only Users 1,3 and 4 performed the measurements on the Armeo® Spring.

16 Discussion

16.1 Accuracy of App Measurements

While the app was successful in measuring the functional workspace volume of 2/8 users, it is far from consistent in accurately measuring the protocol performed by the user. As has been shown from the results section above, the app has quite erratic measurements especially for single point measurements without averaging. We suspect that this has to do with the use of the linear accelerometer in removing gravity. Analysis of the raw accelerometer and gravity vector data indicates that some acceleration was shifted to the gravity vector, resulting in lower than expected linear accelerometer values. Previous attempts at using the accelerometer itself, however, resulted in poor measurement results unless used in carefully controlled scenarios. The benefit of using the linear accelerometer was determined to be far greater than the loss in acceleration.

A possible method to improve the accuracy of the app measurements is to increase the number of points that were measured by the protocol. This will allow for the potential to "drop" points that have high error bars which distorts the accuracy of the app measurements. In addition, from Figure 20, the results suggest that the more measurements that are performed, the better the accuracy of the app. Similarly, increasing the number of points measured by the protocol should in theory increase the accuracy of the app as well.

The app measures the functional workspace by assuming that the radius is the distance from the initial starting position to the endpoint at the wrist. For the sake of simplicity, a spherical model using boundaries at the table and at the torso were used. However, physically, the point of rotation is along the shoulder not the torso, and the full functional workspace may be more accurately measured by using the length up to the palm. As such, an ellipsoid centered at the shoulder may be a more accurate model.

16.2 ROM Protocol: Point Reaching vs. Curve Tracing Exercise

The ROM protocol as described in Section 13.1 involves a series of upper-arm reaching exercises, where the user reaches out to 9 points in space that form one-quarter of an elliptical sphere. An earlier design involves tracing out

3 curves in space corresponding to the XY-, YZ- and XZ-planes. While such design arguably would enable a faster and more intuitive exercise to be performed when measuring the workspace, we opted for the current protocol in order to improve the accuracy of the sensor readings.

Although a sphere is fitted to the functional workspace, the protocol does not assume reach is consistent in all directions. This may especially hold true for those with a mobility impairment, such as SCI patients. In terms of the calculated functional workspace, this app takes the average distance reached along all points in all directions. If the user has difficulty reaching in certain directions, it would skew the volume in all axes instead of just the side which has limited mobility. One improvement for the protocol would be to modify the protocol so that the user reaches the same point several times and the adjust the model to calculate a more irregular shape. Further modification would be to have the app record the user's motions through all of space. The maximum distances in all directions would be used to generate a free-form surface. The volume encapsulated by this surface would then be the user's functional workspace.

Sensor accuracy. Tracing curves created an error of approximately 12.8 cm per meter under controlled circumstances (only translation along x, y, z; no rotation), which reduced to 2 cm per meter after 10 trials. When transitioning to following that corresponding protocol, which allowed the smartwatch to rotate along all 9 axes, we were unable to get a trace. Free-form tracing involves frequent changes to the intrinsic frame of reference, relative to the extrinsic frame, resulting in the inability to correct the acceleration vector. In addition, these traces would involve greater distances which compounds the error due to dead reckoning. The current protocol chosen helps increase sensor accuracy by using short distances where the distance can be reset with each step, and introduces less changes in the intrinsic frame of reference.

16.3 Comparison to Other Clinical Assessment Methods

We have developed a novel system for assessing upper limb recovery. To our knowledge, we are the first to develop a measurement for functional workspace using metric in this form.

Data-driven approach. Our system provides a more objective metric for measuring upper-limb capabilities in SCI patients. Previously, the assessments used are mostly based on scoring done by clinicians. Gold standard examples of these are the INSCSCI neurological assessment and SCIM III functional assessment. Our system enables a more data-driven approach in measuring upper-limb rehabilitation progress.

Frequency. Our system also allows a more frequent assessment for patients, enabling measurements to be performed every day. SCIM III functional assessment has a self-administered questionnaire version that has been developed to allow patients to perform the assessment on their own at home, however, the metrics used are still fairly subjective when compared to our metrics.

Cost. We also achieved a more cost-effective alternative to spatial position-based ROM assessments. Our system, inclusive of smartwatch and smartphone, costs less than 2% of the price of one Armeo Spring unit (estimated based on [25]. Refer to Table 3 for cost breakdown.)

16.4 Project Challenges

This section discusses the challenges faced during the progression of the project.

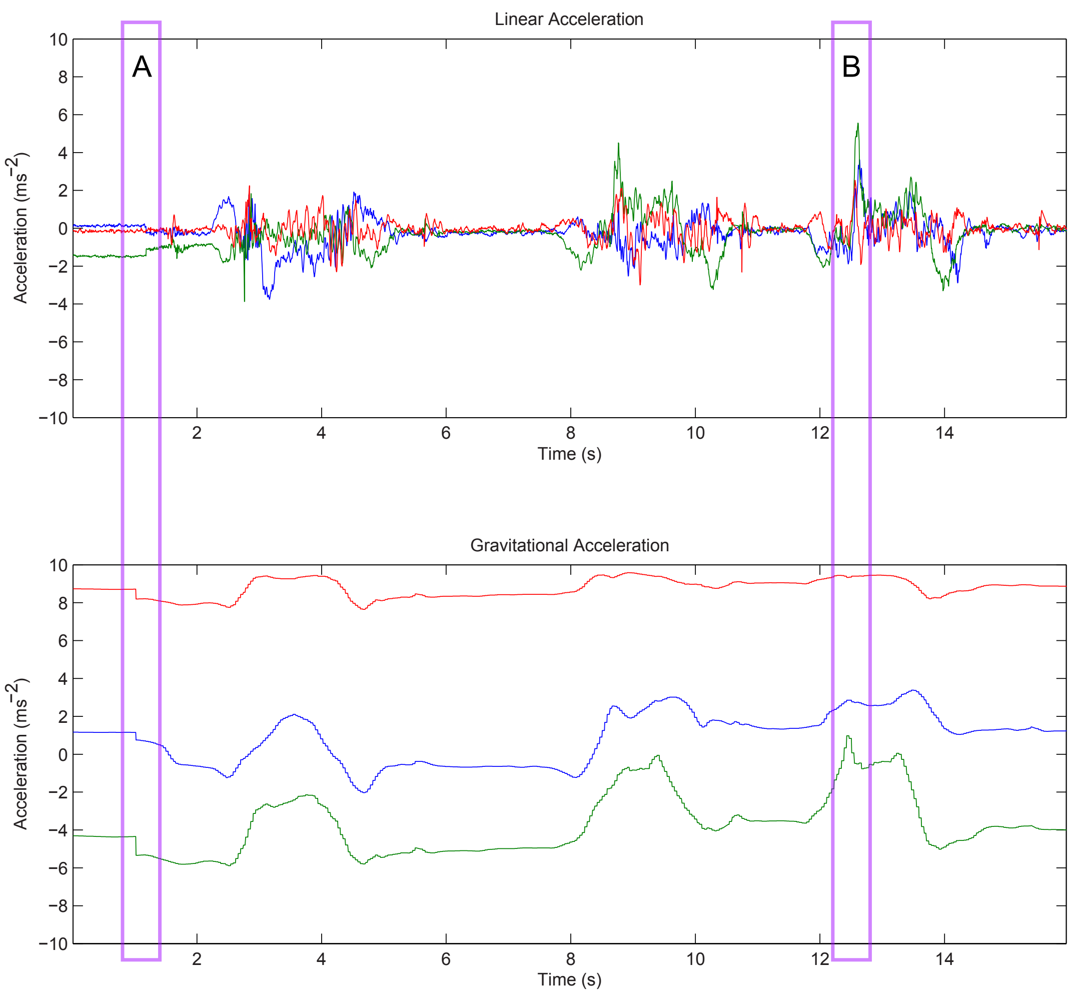


Figure 21. Acceleration values from this recording exemplify common issues encountered with use of the linear accelerometer on the LG G Watch. At rest, the acceleration is commonly not zero resulting in large accumulated errors (A), and peaks in acceleration may be considered the effects of gravity (B).

16.4.1 Technical Challenges

Primarily, the technical challenges we experienced revolved around limitations in the IMU sensors available on our development devices, as well as limitations in how the application is able to interface with the hardware. To maximize efficiency, development occurred in parallel. We did not have the chance to review all design decisions as a group. Therefore, some design decisions may have benefited from in-depth knowledge of the functioning of another subsystem suffered. Because the various subsystems of the application are extremely interdependent, later enhancements may have rendered a previously-discarded alternative design in another subsystem the new optimal option.

Some of these technical challenges required extensive modifications of other subsystems in the middle of the project (see ROM protocol issues in 16.2). Some of these challenges were discussed in an earlier section: noisy signal due thermal noise, inconsistent sampling rate between the accelerometer and the gyroscope, and accounting for gravitational acceleration and device rotation (see the section on signal processing methods 13.3.2, and our solutions in 13.3.3).

In the following, we discuss many of the technical limitations and some proposed solutions for possible mitigation that we were not able to develop due to one or more limitations including device availability, budget, and time.

Linear acceleration. During initial testing, the reconstructed position values were lower than expected. An analysis of the signals processing pipeline at that stage identified the linear acceleration virtual sensor as the cause. The InvenSense proprietary algorithms for removing the gravitational components of the acceleration vector also appeared to reduce the peaks of sudden increases in acceleration. We confirmed this by examining the gravitation components of acceleration as provided by the (`TYPE_GRAVITY`) sensor [31]. Representative examples of common issues are shown in Figure 21. Furthermore, the components of acceleration assigned to the linear acceleration and gravity virtual sensors change in steps (Figure 21, A). Even at rest, the linear accelerometer sensor may also read non-zero values for acceleration, in this case approximately -1.5 ms^{-2}, resulting in large errors in the reconstructed position.

Methods for data filtering. Early in the development of the signals processing pipeline, the raw acceleration and gyroscope data was recorded from the device IMU. Sample data used for development were recorded with the device in a constant orientation. Therefore, the gravitational acceleration was rendered as a constant DC signal.

We tested the utility of various FIR filters, Butterworth IIR, and an FFT-based filter for removing the gravitational acceleration, based on the accuracy of the resulting reconstructed position values. Completely removing any component of gravity within the data was crucial due to the effects of integration. Over the course of a 10 sec recording, a constant 0.002 ms^{-2} acceleration, equivalent to a 73.8 dB attenuation of the 9.81 ms^{-2} acceleration of gravity, would yield an inaccuracy of 10 cm. Therefore, the accuracy of the acceleration was critical. Although the FFT-based filter (methods detailed in 13.3.3 may introduce artifacts in the filtered signal, the FFT-based filter was consistently found to return the best result. The bulk of the device motion acceleration signal presented as low-frequency components <0.5 Hz and the filter orders required for FIR and IIR filters to remove gravity while retaining the low-frequency signal were unacceptably large.

Later, we made the decision to use the linear acceleration virtual sensor rather than the raw acceleration sensor. The linear acceleration sensor theoretically removes the effects of gravity, thus the filter was no longer necessary. We were unable to review the specific algorithms used for determining the linear acceleration since they are proprietary to InvenSense Inc. We did notice that the linear acceleration still exhibited slight DC bias on the order of 10^{-2} ms^{-2}. As above, the slight DC bias resulted in accumulated errors of tens of centimeters. Thus, we compensated for this DC bias by subtracting the moving average of the data over a relatively long interval of 3 sec, and this method was fairly successful. The combination of the linear acceleration and moving average debiasing of the acceleration data rendered the use of the filter redundant. Furthermore, the artifacts that the FFT-filter could introduce into the acceleration data may be materially detrimental to the reconstructed position. Time constraints did not permit us to revisit the selection of the filtering method, and therefore it remained in our signals processing algorithms.

16.4.2 Time Limitations

The time spent towards overcoming the technical challenges and debugging the system have caused some of the project goals to be unable to be completed within the given time frame of the project. Such goals include the 'Send to Physician' feature that enables data communication from patient to physician, and the system testing involving simulated limited upper-limb movement. These portions of the project has been included as Future Works in Section 18.

17 Source Code

Please contact the authors for access to the full source code repository. To compile the app the Android and Android Wear SDK v21 (Google Inc.) and the Java Development Kit 1.8_u25 (Oracle Inc.) are required. The signals processing algorithm is also available for MATLAB (Mathworks Inc.), requiring the Signals Processing Toolbox and tested with version R2014b. The authors also recommend the installation of Android Studio v1.1.0.

Part VI
Future Work and Conclusion

18 Future Work

18.1 Signals Processing Algorithms

In 16.4.1, we discussed many design decisions which may have been affected by later enhancements to the project that we were unable to revisit due to time constraints. Naturally, future development efforts may benefit from revisiting many of those decisions and reevaluating the most optimal choice given the additional information and enhancements from later in the project.

18.2 Further Validation

As mentioned previously, further testing is required to fine-tune the volume model, as well as validate the measurements for those with limited upper-limb movements.

18.3 Send to Physician Feature

The send to physician feature, while briefly discussed, has not been implemented for this project. The current idea is to implement a simple server-backed system. The app will post the data from the signals processing engine to the server which will store the information into a database. A front-end would be designed to allow the physician to access both raw data and computed results. However, this is only a simple solution. More complex solutions would involve solutions such as: end-to-end encryption, web-based tracking, and physician alerts.

18.4 Expansion of Protocols

The current implementation only contains a single protocol and associated signals processing algorithm. Further work on this project would include adapting the current system to choose a protocol and associated signals processing algorithm such that the app can be used with other motions or limbs. Since the core architecture is modular, this would be a straightforward change in the code base. In addition, the current method of calculating the functional workspace uses a spherical model. Further work can also include changing this model for a more accurate ellipsoid model or even a free-form volume.

18.5 Better Protocol Transitioning

By using the reaching out protocol, the smartwatch is able to switch protocols between the points in space automatically, by assuming the deceleration and acceleration events to indicate protocol transition. This enables the patient to have less interaction with the smartwatch, requiring them to press the screen only during the transition between plane measurements.

18.6 External Sensors

The current implementation relies on a connected smartwatch for the sensors. However, the architecture allows for the expansion of this to include other sensors, either from additional Android Wear devices, or custom sensors (more sensitive accelerometers, grip strength devices, etc.). Custom sensors would have to be designed to return a data-stream via bluetooth which can be accepted by Android. The associated sensor module code would have to be designed to interpret that data-stream in order to format it to the common format used by the app.

18.7 Comparative Statistics

There is currently no comparative standard which physicians used to quantify upper-limb range of motion and functional workspace. As such, the data currently generated can only be used to look at a patient's performance relative to themselves. Future uses of this app can be to track functional workspace measurements of thousands of users in a big-data format to map out a range of volumes which correspond to a "normal" functional workspace volume.

19 Conclusion

19.1 Goal Completion

Three out of four of the project requirements have been met: 1) Smartwatch IMU data can be used to collect patient movement data and is communicated wirelessly to the smartphone, and smartphone is capable to calculate the user's functional workspace; 2) changes in the user's functional workspace can be tracked over time based on daily measurements; and 3) graphical displays can be generated to visualize these changes. These three goals have been achieved, with validation tests performed at the minimum, for the patients' protocol as described in Appendix A. The fourth project requirement, communicating logs and summary-statistics to clinicians, have not been implemented due to time constraints of the project. The works required for this feature is discussed in Section 18.3.

19.2 Summary

SCI is a debilitating and costly disability. Current treatment monitoring methods are lacking in frequency, convenience and data-drivenness. We have successfully developed a novel Android-based system for frequent upper-limb functional workspace measurements to track neurological recovery of patients with SCI below the C4 level in a non-clinical setting. Our system emphasizes on simple and frictionless user experience. This system utilizes the device IMU to track spatial position, mobile computing to calculate functional workspace, and displays the information and trends in easy-to-understand format.

Part VII

References

[1] S. Kalsi-Ryan, J. Wilson, J. M. Yang, and M. G. Fehlings, “Neurological grading in traumatic spinal cord injury,” *World Neurosurgery*, vol. 82, no. 3/4, pp. 509–518, 2014. DOI: 10.1016/j.wneu.2013.01.007.

[2] C. G. Fisher, V. K. Noonan, D. E. Smith, P. C. Wing, M. F. Dvorak, and B. Kwon, “Motor recovery, functional status, and health-related quality of life in patients with complete spinal cord injuries,” *Spine*, vol. 30, no. 19, pp. 2200–2207, 2005. DOI: 10.1097/01.brs.0000181058.06412.a9.

[3] K. D. Anderson, “Targeting recovery: priorities of the spinal cord-injured population,” *Journal of Neurotrauma*, vol. 21, no. 10, pp. 1371–1383, 2004.

[4] M. N. Hadley, B. C. Walters, B. Aarabi, S. S. Dhall, D. E. Gelb, R. J. Hurlbert, C. J. Rozzelle, T. C. Ryken, and N. Theodore, “Clinical assessment following acute cervical spinal cord injury,” *Neurosurgery*, vol. 72, pp. 40–53, 2013. DOI: 10.1227/neu.0b013e318276edda.

[5] S. C. Kirshblum, S. P. Burns, F. Biering-Sorensen, W. Donovan, D. E. Graves, A. Jha, M. Johansen, L. Jones, A. Krassioukov, M. Mulcahey, M. Schmidt-Read, and W. Waring, “International standards for neurological classification of spinal cord injury,” *J Spinal Cord Med*, vol. 34, no. 6, pp. 535–546, 2011. DOI: 10.1179/204577211X13207446293695.

[6] A. Jha, “ASIA impairment scale,” in *Encyclopedia of Clinical Neuropsychology*, Springer New York, 2011, pp. 255–257. DOI: 10.1007/978-0-387-79948-3_1793.

[7] S. Kalsi-Ryan, A. Curt, M. C. Verrier, and M. G. Fehlings, “Development of the graded redefined assessment of strength, sensibility and prehension (grassp): reviewing measurement specific to the upper limb in tetraplegia,” *JNS: Spine Special Supplements*, vol. 17, no. 1, pp. 65–76, 2012. DOI: 10.3171/2012.6.AOSPINE1258.

[8] M. Itzkovich, I. Gelernter, F. Biering-Sorensen, C. Weeks, M. T. Laramee, B. C. Craven, M. Tonack, S. L. Hitzig, E. Glaser, G. Zeilig, S. Aito, G. Scivoletto, M. Mecci, R. J. Chadwick, W. S. E. Masry, A. Osman, C. A. Glass, P. Silva, B. M. Soni, B. P. Gardner, G. Savic, E. M. Bergström, V. Bluvshtein, J. Ronen, and A. Catz, “The spinal cord independence measure (SCIM) version III: reliability and validity in a multi-center international study,” *Disabil Rehabil*, vol. 29, no. 24, pp. 1926–1933, Jan. 2007. DOI: 10.1080/09638280601046302.

[9] C. Fekete, I. Eriks-Hoogland, M. Baumberger, A. Catz, M. Itzkovich, H. Lüthi, M. W. M. Post, E. von Elm, A. Wyss, and M. W. G. Brinkhof, “Development and validation of a self-report version of the spinal cord independence measure (SCIM III),” *Spinal Cord*, vol. 51, no. 1, pp. 40–47, Aug. 2012. DOI: 10.1038/sc.2012.87.

[10] E. Widerström-Noga, F. Biering-Sørensen, T. N. Bryce, D. D. Cardenas, N. B. Finnerup, M. P. Jensen, J. S. Richards, and P. J. Siddall, “The international spinal cord injury pain basic data set (version 2.0),” *Spinal Cord*, vol. 52, no. 4, pp. 282–286, Jan. 2014. DOI: 10.1038/sc.2014.4.

[11] H. I. Krebs, B. Volpe, M. Ferraro, S. Fasoli, J. Palazzolo, B. Rohrer, L. Edelstein, and N. Hogan, “Robot-aided neurorehabilitation: from evidence-based to science-based rehabilitation,” *Top Stroke Rehabil*, vol. 8, no. 4, pp. 54–70, 2002.

[12] H. Kern, U. Carraro, N. Adami, D. Biral, C. Hofer, C. Forstner, M. Mödlin, M. Vogelauer, A. Pond, S. Boncompagni, C. Paolini, W. Mayr, F. Protasi, and S. Zampieri, “Home-based functional electrical stimulation rescues permanently denervated muscles in paraplegic patients with complete lower motor neuron lesion,” *Neurorehabilitation and Neural Repair*, vol. 24, no. 8, pp. 709–721, 2010. DOI: 10.1177/1545968310366129.

[13] J. Zariffa, N. Kapadia, J. L. K. Kramer, P. Taylor, M. Alizadeh-Meghrazi, V. Zivanovic, U. Albisser, R. Willms, A. Townson, A. Curt, M. R. Popovic, and J. D. Steeves, "Relationship between clinical assessments of function and measurements from an upper-limb robotic rehabilitation device in cervical spinal cord injury," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 20, no. 3, pp. 341–350, May 2012. DOI: `10.1109/tnsre.2011.2181537`.

[14] B. C. Werner, R. E. Holzgrefe, J. W. Griffin, M. L. Lyons, C. T. Cosgrove, J. M. Hart, and S. F. Brockmeier, "Validation of an innovative method of shoulder range-of-motion measurement using a smartphone clinometer application," *Journal of Shoulder and Elbow Surgery*, vol. 23, no. 11, pp. 275–282, 2014. DOI: `10.1016/j.jse.2014.02.030`.

[15] S. H. Shin, D. H. Ro, O.-S. Lee, J. H. Oh, and S. H. Kim, "Within-day reliability of shoulder range of motion measurement with a smartphone," *Manual Therapy*, no. 17, pp. 298–304, 2012. DOI: `10.1016/j.math.2012.02.010`.

[16] G. Ferriero, S. Vercelli, F. Sartorio, S. M. Lasa, E. Ilieva, E. Brigatti, C. Ruella, and C. Foti, "Reliability of a smartphone-based goniometer for knee joint goniometry," *International Journal of Rehabilitation Research*, no. 36, pp. 146–151, 2013. DOI: `10.1097/MRR.0b013e32835b8269`.

[17] J.-Y. Jenny, "Measurement of the knee flexion angle with a smartphone-application is precise and accurate," *The Journal of Arthroplasty*, no. 28, pp. 784–787, 2013. DOI: `10.1016/j.arth.2012.11.013`.

[18] A. Jones, R. Sealey, M. Crowe, and S. Gordon, "Concurrent validity and reliability of the simple goniometer iphone app compared with the universal goniometer," *Physiother Theory Pract*, vol. 30, no. 7, pp. 512–516, 2014. DOI: `10.3109/09593985.2014.900835`.

[19] S. Mellone, C. Tacconi, and L. Chiari, "Validity of a smartphone-based instrumented timed up and go," *Gait & Posture*, no. 36, pp. 163–165, 2012. DOI: `10.1016/j.gaitpost.2012.02.006`.

[20] G. Spina, G. Huang, A. Vaes, M. Spruit, and O. Amft, "Copdtrainer: a smartphone-based motion rehabilitation training system with real-time acoustic feedback," *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pp. 597–606, 2013. DOI: `10.1145/2493432.2493454`.

[21] D. Evans. (Jan. 9, 2014). The wearable tech that got pulses racing at ces 2014, Techradar, [Online]. Available: `https://web.archive.org/web/20150126013711/http://www.techradar.com/news/portable-devices/the-wearable-tech-that-got-pulses-racing-at-ces-2014-1213806` (visited on 01/25/2015).

[22] T. Simonite. (Jan. 8, 2014). Ces 2014: less is more for smart watches and other wearable gadgets, MIT Technology Review, [Online]. Available: `http://www.technologyreview.com/news/523376/ces-2014-less-is-more-for-smart-watches-and-other-wearable-gadgets/` (visited on 01/25/2015).

[23] S. Webster. (Oct. 28, 2014). Google fit walks onto android devices, CNET, [Online]. Available: `https://web.archive.org/web/20150126015810/http://www.cnet.com/news/google-fit-android-app/` (visited on 01/25/2015).

[24] Z. Tian, Y. Zhang, M. Zhou, and Y. Liu, "Pedestrian dead reckoning for MARG navigation using a smartphone," *EURASIP Journal on Advances in Signal Processing*, no. 1, p. 65, 2014. DOI: `10.1186/1687-6180-2014-65`.

[25] G. Turchetti, N. Vitiello, L. Trieste, S. Romiti, E. Geisler, and S. Micera, "Why effectiveness of robot-mediated neurorehabilitation does not necessarily influence its adoption," *IEEE Reviews in Biomedical Engineering*, vol. 7, pp. 143–153, 2014. DOI: `10.1109/rbme.2014.2300234`.

[26] (2014). Smartphone os market share, q3 2014, International Data Corporation, [Online]. Available: `https://web.archive.org/web/20150126020238/http://www.idc.com/prodserv/smartphone-os-market-share.jsp` (visited on 01/25/2015).

[27] N. Kothari, B. Kannan, E. D. Glasgwow, and M. B. Dias, "Robust indoor localization on a commercial smart phone," *Procedia Computer Science*, vol. 10, pp. 1114–1120, 2012. DOI: `10.1016/j.procs.2012.06.158`.

[28] iFixit. (2014). Lg g watch teardown, [Online]. Available: `https://www.ifixit.com/Teardown/LG+G+Watch+Teardown/27037` (visited on 04/27/2015).

[29] G. Inc. (2015). Sensorevent, [Online]. Available: `https://developer.android.com/reference/android/hardware/SensorEvent.html` (visited on 03/10/2015).

[30] Y. S. Kim, S.-w. Jang, and Y.-S. Yoo, "Mobile assessment system for shoulder joint rehabilitation: system development and preliminary study," *IJBSBT*, vol. 6, no. 2, pp. 51–60, Apr. 2014. DOI: `10.14257/ijbsbt.2014.6.2.05`.

[31] G. Inc. (2015). Sensors overview, [Online]. Available: `https://developer.android.com/guide/topics/sensors/sensors_overview.html` (visited on 03/10/2015).

Part VIII
Appendices

A Volunteer Participation Consent Form

DESCRIPTION: You are invited to participate in a study on wearable and gesture technology for rehabilitation diagnostics involving a mobile app, being undertaken in the context of a design project for BME489. You will be asked to wear a smartwatch and make arm gestures according to audio instructions. Accelerometry data from the smartwatch will be recorded. You will NOT be asked to download any app onto your smartphone.

TIME INVOLVEMENT: Your participation will take approximately 10 to 15 minutes.

RISKS AND BENEFITS: There are no identified risks for participation in this study. The risk level is minimal, similar to risks during a stretching exercise. The benefit which may reasonably be expected is that you will be among the contributors towards the development of an early stage of a decentralized rehabilitation diagnostic technology. You may also benefit from the satisfaction of having assisted some engineering students complete one of the requirements of their course project.

CONDITIONS FOR PARTICIPATING: Your participation is voluntary, and you may withdraw at any time, and/or decline to participate in any parts of the procedures/tasks – all without negative consequences. All data will be anonymized, and if you choose to withdraw from the study before completion, data collected up to that point will be retained and will be kept anonymous.

ACCESS TO INFORMATION, CONFIDENTIALITY AND RESULTS PUBLICATION: Raw data will be immediately anonymized and stored on encrypted computers available only to our team and supervisor. Findings from the data will be reported in the final deliverables for BME489. These will be reported both internally and to collaborators at the Toronto Rehabilitation Institute – University Health Network, as well as presented to the public though a showcase presented at the University of Toronto. Examples of data analyses that will be presented include spatial traces of hand gestures, and a metric of app usability. None of the analyses will contain identifying information, images of participants, audio or video.

CONTACT INFORMATION: If you have any questions, concerns or complaints about this research, its procedures, risks and benefits, contact Afiq Asri at mohdafiq.mohdasri@mail.utoronto.ca. You can contact the Office of Research Ethics at ethics.review@utoronto.ca or 416-946-3273, if you have questions about your rights as a participant.

Indicate Yes or No:

I have read and understood the description of the experiment, and I give consent to participate in this study as described. ___Yes ___No

The extra copy of this signed and dated consent form is for you to keep.

SIGNATURE .... DATE ....

B User Protocol

This user protocol describes the necessary steps to measure your functional workspace using the **ther**app**y** app for Android.

Preparation for Measurements

1. Sit upright at a table or desk with your torso touching the edge of the table surface.
2. Adjust your chair so that it is in contact with your back during the exercises.
3. Place your smartwatch on the wrist of the arm you want to measure.
4. Open the **ther**app**y** app on your smartphone tap the "start measurement session" button.
5. Place your smartphone on the table under your smartwatch hand (See Figure 22).
6. When you are ready to begin your measurement, tap the screen to begin. A 5 second countdown will commence.

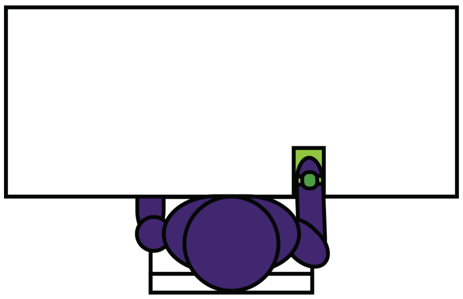


Figure 22. Protocol starting position.

Stage 1: X-Y Plane Measurement

1. Move your smartwatch hand as indicated in the animation shown on your smartphone screen, reaching as far out as possible.
2. Begin by reaching out to your far left.
3. As you return to the starting position, tap the smartphone screen to advance to the next step.
4. Repeat, reaching front left, forward, front right, and then far right.
5. This stage will be complete after you have completed the 5 steps needed to reach for the points shown in Figure 23. The app will prompt you to the next stage.

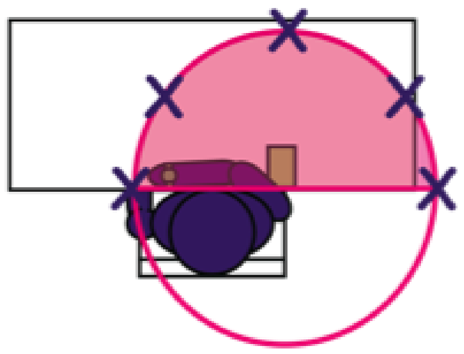


Figure 23. Steps for Stage 1 (X-Y plane measurements), indicated by “X”’s.

Stage 2: Y-Z Plane Measurement

1. Move your smartwatch hand as indicated in the animation shown on your smartphone screen, reaching as far out as possible.
2. Begin by reaching out in front of you, at eye-level.
3. As you return to the starting position, tap the smartphone screen to advance to the next step.
4. Then reach directly above your head.
5. Return to the starting position and tap the smartphone screen to continue advancing.
6. This stage will be complete after you have completed the 2 steps needed to reach for the points shown in Figure 24. The app will prompt you to the next stage.

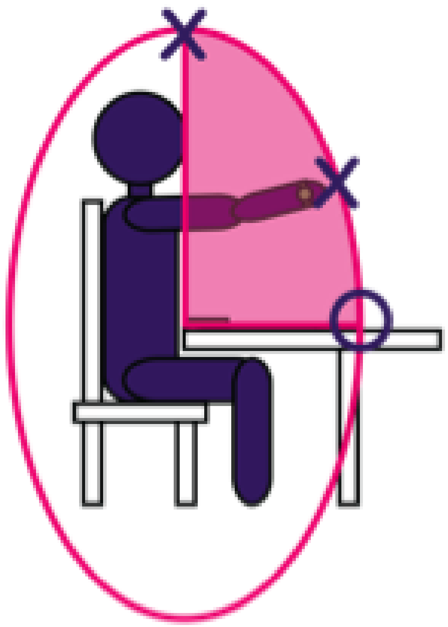


Figure 24. Steps for Stage 2 (Y-Z plane measurements), indicated by “X”’s.

Stage 3: X-Z Plane Measurement

1. Move your smartwatch hand as indicated in the animation shown on your smartphone screen, reaching as far out as possible.
2. Begin by reaching out to your left, as high and as far as possible.
3. As you return to the starting position, tap the smartphone to advance to the next step.
4. Repeat this, but reaching to your right.
5. This stage will be complete after you have completed the 2 steps needed to reach for the points shown in Figure 25.
6. At the end of this stage, the protocol will be complete. The app will automatically advance to the signals processing stage and return to the main menu when complete.

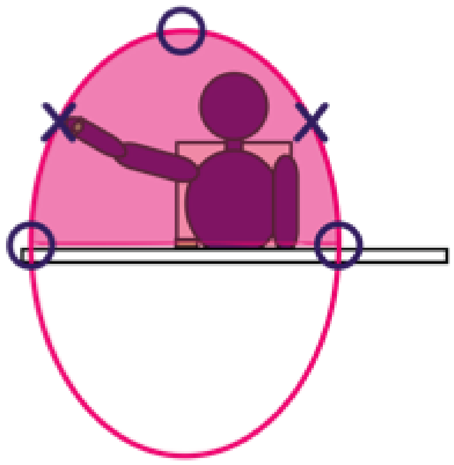


Figure 25. Steps for Stage 3 (X-Z plane measurements), indicated by "X"'s.

Appendix

C Code: Signal Processing Engine

```java
package ca.utoronto.therappysignalstest;

import java.util.ArrayList;
import java.util.Collections;

import org.apache.commons.math3.analysis.interpolation.LinearInterpolator;
import org.apache.commons.math3.analysis.polynomials.PolynomialSplineFunction;
import org.apache.commons.math3.complex.Complex;
import org.apache.commons.math3.transform.DftNormalization;
import org.apache.commons.math3.transform.FastFourierTransformer;
import org.apache.commons.math3.transform.TransformType;
import org.apache.commons.math3.stat.StatUtils;
import org.apache.commons.math3.geometry.euclidean.threed.Rotation;


/**
 * Created by simeon on 2015-03-14.
 */

public class SPM_FunctionalWorkspace {

    // Constants
    private final static double LONGTERM_WND_LENGTH = 3.0;
    private final static double time_div = 1E-9;       // timestamp is in nanoseconds

    // INPUT
    private ArrayList<sensorPoint> data_input;

    // Interim
    private double [][] resampled_data;
    private int [][] resampled_idx;
    private double meandiff;

    private double [][] data_accl, data_rota;
    private double[] time_accl, time_rota;
    private double[][] time_split;

    // OUTPUTS
    private double [] fitmeasures; // fwvol, xyarea, yzarea, xzarea;
    private double [] maxpositions;
    private ArrayList<double[]> position;

    // when creating the signal processing module, must provide acceleration data
    public SPM_FunctionalWorkspace(ArrayList<sensorPoint> data_input) {
        // pass the loaded acceleration data here
        // the vector is pre-rotated
        this.data_input = data_input;
    }

    // do the whole signals processing thing here.
    public void doChurnData () {
        // do signals processing stuff

        Collections.sort(this.data_input);

        // STEP: remove duplicates and separate data
        doSeparateData();

        // STEP: preprocess everything
        this.doPreprocessing();

        // STEP: run signals processing code on it.
        ArrayList<double[]> position = new ArrayList<>();
        this.maxpositions = new double[this.resampled_idx.length];      // this stores the max position within
each interval

        for(int kk = 0; kk < this.resampled_idx.length; kk++) {
            int[] curr = this.resampled_idx[kk];

            double[][] tempposition;
            tempposition = this.doIntegration(curr[0], curr[1]-curr[0]);

            // transfer new position vectors into array
            position.ensureCapacity(position.size() + tempposition[0].length);

            for(int jj = 0; jj < tempposition[0].length; jj++) {
                position.add(new double[]{tempposition[0][jj], tempposition[1][jj], tempposition[2][jj]});

                // within each interval, find max position
                double vecnorm = Math.sqrt(Math.pow(tempposition[0][jj],2) + Math.pow(tempposition[1][jj],2) + Math.pow(tempposition[2][jj],2));
                this.maxpositions[kk] = Math.max(vecnorm, this.maxpositions[kk]);
            }

        }
        this.position = position;

        // STEP: fit areas to get metrics
        doFitTargets();
    }

    // separate the input sensorPoint data into individual components for processing
    protected void doSeparateData() {

        // ensure sorted
        Collections.sort(this.data_input);

        ArrayList<Integer> idx_accl = new ArrayList<>(), idx_gyro = new ArrayList<>(), idx_split = new ArrayList<>();

        // add first segment
        idx_split.add(-1);

        // count the number of each item
        for(int kk = 0; kk < this.data_input.size(); kk++) {
            int type = this.data_input.get(kk).datatype;

            if(type == sensorPoint.DATA_ACCELERATION) {
                idx_accl.add(kk);
            }
            else if(type == sensorPoint.DATA_ROTATIONVEC) {
                idx_gyro.add(kk);
            }
            else if(type == sensorPoint.TRACE_BREAK) {
                idx_split.add(kk);
            }
        }

        // remove duplicates
        idx_accl = doRemoveDuplicates(idx_accl);
        idx_gyro = doRemoveDuplicates(idx_gyro);


        // >>> take data out of sensor points
        // - acceleration
        this.data_accl = new double[3][idx_accl.size()];
        this.time_accl = new double[idx_accl.size()];

        for(int kk = 0; kk < idx_accl.size(); kk++) {
            sensorPoint sp = this.data_input.get(idx_accl.get(kk));

            this.data_accl[0][kk] = sp.value[0];
            this.data_accl[1][kk] = sp.value[1];
            this.data_accl[2][kk] = sp.value[2];

            this.time_accl[kk] = sp.time * time_div;
        }

        // - rotation
        this.data_rota = new double[3][idx_gyro.size()];
        this.time_rota = new double[idx_gyro.size()];

        for(int kk = 0; kk < idx_gyro.size(); kk++) {
            sensorPoint sp = this.data_input.get(idx_gyro.get(kk));

            this.data_rota[0][kk] = sp.value[0];
            this.data_rota[1][kk] = sp.value[1];
            this.data_rota[2][kk] = sp.value[2];

            this.time_rota[kk] = sp.time * time_div;
        }

        // - split
        this.time_split = new double[idx_split.size()][2];

        for(int kk = 0; kk < idx_split.size(); kk++) {
            this.time_split[kk][0] = this.data_input.get(idx_split.get(kk)+1).time * time_div;

            if(kk+2 >= idx_split.size()) {
                this.time_split[kk][1] = this.data_input.get(this.data_input.size()-1).time * time_div;
            } else {
                this.time_split[kk][1] = this.data_input.get(idx_split.get(kk+1)-1).time * time_div;
            }
        }

      //  this.data_input.clear();
        //this.data_input = null;
    }

    protected void doFitTargets() {
        // points for xy, yz, xz
        int[][] indices = new int[][]{new int[] {0, 1, 2, 3, 4, 5, 6, 7, 8},
                new int[]{0, 1, 2, 3, 4}, new int[]{2, 5, 6}, new int[]{0, 7, 6, 8, 4}};
        this.fitmeasures = new double[indices.length];

        for (int aa = 0; aa < indices.length; aa++) {

            this.fitmeasures[aa] = 0;
            for (int bb = 0; bb < indices[aa].length; bb++) {
                this.fitmeasures[aa] += this.maxpositions[indices[aa][bb]];
            }
            this.fitmeasures[aa] /= (indices[aa].length+1);
        }

        // calculate sphere volume
        this.fitmeasures[0] = Math.PI * Math.pow(this.fitmeasures[0], 3) / 3.0;

        // calculate workspace areas
        for (int aa = 1; aa < indices.length; aa++) {
            this.fitmeasures[aa] = Math.PI * Math.pow(this.fitmeasures[aa], 2) / 2.0;
        }
    }


    /* perform acceleration preprocessing: all the steps up until integration.
       split into segments, resample, filter, subtract moving average   */
    protected void doPreprocessing() {


        // STEP: do linear interpolation of the data
        double startTime = Math.max(this.time_accl[0], this.time_rota[0]),
               endTime = Math.min(this.time_accl[this.time_accl.length - 1], this.time_rota[this.time_rota.
length - 1]);

        //  - find sampling frequency
        double meandiff = StatUtils.mean(calculateDiff(this.time_accl));
        meandiff = meandiff / 5;                            // aim for oversample by 5x
        int resampled_length = (int) Math.floor((endTime - startTime) / meandiff);

        //  - turn resampled_length into closest higher power of 2
        resampled_length = (int) Math.pow(2, Math.ceil(  (Math.log(resampled_length)/Math.log(2)) - 0.1 ));
        meandiff = (endTime - startTime) / (resampled_length-1);

        //  - generate new time vector
        double[] resampled_time = new double[resampled_length];

        for(int kk = 0; kk < resampled_length; kk++) {
            resampled_time[kk] = (meandiff * kk) + startTime;
        }

        //  - resample each acceleration dimension
        double[][] resampled_data = new double[3][];
        for(int kk = 0; kk < 3; kk++) {
            resampled_data[kk] = calculateInterp1(resampled_time, this.time_accl, this.data_accl[kk]);
        }

        //  - free for garbage collect
        this.data_accl = null;
        this.time_accl = null;

        // STEP: rotate the acceleration vectors
        double[][] resampled_rotation = new double[3][];
        for(int kk = 0; kk < 3; kk++) {
            resampled_rotation[kk] = calculateInterp1(resampled_time, this.time_rota, this.data_rota[kk]);
        }

        for(int tt = 0; tt < resampled_length; tt++) {
            double q0 = Math.sqrt(1 - Math.pow(resampled_rotation[0][tt], 2) - Math.pow(resampled_rotation[1][tt], 2) - Math.pow(resampled_rotation[2][tt], 2));
            Rotation rotator = new Rotation(q0, resampled_rotation[0][tt], resampled_rotation[1][tt], resampled_rotation[2][tt], false);

            double[] output = new double[3];
            rotator.applyTo(new double[]{resampled_data[0][tt], resampled_data[1][tt], resampled_data[2][tt]}, output);

            resampled_data[0][tt] = output[0];
            resampled_data[1][tt] = output[1];
            resampled_data[2][tt] = output[2];
        }


        // *****************************************

        // STEP: filter data
        for(int kk = 0; kk < 3; kk++) {
            double normalized_hicutoff = 30 * meandiff / 2;
            resampled_data[kk] = calculateFilterNoDC_FFT(resampled_data[kk], normalized_hicutoff);
        }
        // *****************************************


        int half_longterm_wnd_length = (int) Math.round((LONGTERM_WND_LENGTH / 2) / meandiff);

        // STEP: subtract longer term moving average
        double[][] resampled_debiased_data = new double[3][resampled_length];

        for(int kk = 0; kk < 3; kk++) {
            for(int tt = 0; tt < resampled_length; tt++) {
                int idxBegin = Math.max(0, tt-half_longterm_wnd_length),
                    idxLength = Math.min(resampled_length-1, tt+half_longterm_wnd_length) - idxBegin;
                resampled_debiased_data[kk][tt] = resampled_data[kk][tt] - StatUtils.mean(resampled_data[kk], idxBegin, idxLength);
            }
        }
        // *****************************************


        // STEP: convert separator times into indices
        this.resampled_idx = new int[this.time_split.length][2];

        for(int kk = 0; kk < this.resampled_idx.length; kk++) {
            this.resampled_idx[kk][0] = (int) Math.ceil((this.time_split[kk][0] - resampled_time[0]) / meandiff);
            this.resampled_idx[kk][1] = (int) Math.floor((this.time_split[kk][1] - resampled_time[0]) / meandiff);
        }
        // *****************************************

        this.resampled_data = resampled_debiased_data;
        this.meandiff = meandiff;
    }


    // perform integration over a certain interval to get position
    protected double[][] doIntegration(int segmentBegin, int segmentLength) {

        // STEP: integrate acceleration twice to get position
        //  - integrate accl to get velocity
        double[][] velocity = new double[3][segmentLength];
        for(int kk = 0; kk < 3; kk++) {
            // initial velocity is zero + change in first time step
            velocity[kk][0] = (meandiff * this.resampled_data[kk][segmentBegin]);

            // loop through time steps and add changes
            for(int tt = 1; tt < segmentLength; tt++) {
```

```java
                velocity[kk][tt] = velocity[kk][tt-1] + (meandiff * this.resampled_data[kk][tt + segmentBegin]);
            }
        }

        //  - integrate velocity to get position
        double[][] position = new double[3][segmentLength];
        for(int kk = 0; kk < 3; kk++) {

            // initial position is zero + change in first time step
            position[kk][0] = (meandiff * velocity[kk][0]);

            for(int tt = 1; tt < segmentLength; tt++) {
                position[kk][tt] = position[kk][tt-1] + (meandiff * velocity[kk][tt]);
            }
        }
        // ****************************************

        return position;

        // TODO: correct position based on return-to-zero
    }


    // calculate the moving average of a 3 x N matrix of values in the N dimension.
    protected double[][] calculateMovingAverage(double[][] input, int numsamples) {
        double[][] output = new double[3][];
        for(int kk = 0; kk < 3; kk++) {
            output[kk] = new double[input[0].length - numsamples + 1];
        }

        for(int kk = 0; kk < output.length; kk++) {
            for(int jj = 0; jj < numsamples; jj++) {
                output[0][kk] += input[0][kk+jj];
                output[1][kk] += input[1][kk+jj];
                output[2][kk] += input[2][kk+jj];
            }

            output[0][kk] = output[0][kk] / numsamples;
        }

        return output;
    }

    // remove values with duplicated time stamps
    // TODO: perhaps consider averaging duplicates instead
    protected ArrayList<Integer> doRemoveDuplicates(ArrayList<Integer> idx) {

        ArrayList<Integer> duplicatedTimes = new ArrayList<>();

        // loop through sensor points and mark duplicated time stamps for removal
        for(int kk = 1; kk < idx.size(); kk++) {
            if(this.data_input.get(idx.get(kk)).time == this.data_input.get(idx.get(kk-1)).time) {
                duplicatedTimes.add(kk);
            }
        }

        // remove all items marked for removal
        // we can't remove while searching, because then the size of the arraylist would change, that that makes
things complicated.
        // - search backwards and delete, so that it doesn't upset the indexing
        for(int kk = duplicatedTimes.size()-1; kk >= 0; kk--) {
            idx.remove((int) duplicatedTimes.get(kk));
        }

        return idx;
    }

    // do 1D linear interpolation of data, similar to matlab interp1 command
    protected double[] calculateInterp1(double[] newTime, double[] oldTime, double[] oldX) {
        // fit linear interpolator model to provided data
        PolynomialSplineFunction psfmodel = (new LinearInterpolator()).interpolate(oldTime, oldX);

        // allocate new data vector of same length
        int num_samples = newTime.length;
        double[] newX = new double[num_samples];

        // get new data values using model
        for(int tt = 0; tt < num_samples; tt++) {
            newX[tt] = psfmodel.value(newTime[tt]);
        }

        // return
        return newX;
    }

    // low pass filter data using an FFT, and removing DC components
    protected double[] calculateFilterNoDC_FFT(double[] datain, double hicutoff) {
        // filter the signal using an FFT / iFFT algorithm, removing the DC component, and any
        // components above the specified hicutoff
        //      hicutoff should be provided as normalized frequency
        //
        // Essentially, this function will:
        //   - transform the signal into frequency space (using FFT)
        //   - create a vector of multiplication ratios
        //   - zero out the multiplication ratio vectors that we want to filter out
        //   - element-wise multiply the ratio vector with the frequency space of signal
        //   - do inverse FFT to recover filtered signal

        int num_samples = datain.length;

        // find corresponding index
        int hicutoffidx = (int) Math.ceil(num_samples * hicutoff);

        // FFT MAGICKS HAPPENS HERE
        // ***************
        FastFourierTransformer fftengine = new FastFourierTransformer(DftNormalization.STANDARD);
        Complex [] fftoutput = fftengine.transform(datain, TransformType.FORWARD);
        // ***************

        // create a vector of things to zero out, set everything to 1
        double[] zeroidx = new double[num_samples];
        for (int kk = 0; kk < num_samples; kk++) {
            zeroidx[kk] = 1;
        }

        // zero out DC component
        zeroidx[0] = 0;

        // zero out high frequency components above the hicutoff
        int vecmiddle = (int) Math.ceil(num_samples/2);
        for(int kk = hicutoffidx; kk < vecmiddle; kk++) {
            zeroidx[kk] = 0;
        }

        // mirror the zero-out vector (since FFT is mirrored)
        for(int kk = 0; kk < vecmiddle; kk++) {
            zeroidx[num_samples - kk - 1] = zeroidx[kk];
        }

        // actually zero out the components that need to be zeroed out
        for(int kk = 0; kk < num_samples; kk++) {
            fftoutput[kk] = fftoutput[kk].multiply(zeroidx[kk]);
        }

        // INVERSE FFT MAGICKS HAPPENS HERE
        // ***************
        Complex[] cpx_output = fftengine.transform(fftoutput, TransformType.INVERSE);
        // ***************

        // get real part of the FFT output.
        double[] output = new double[num_samples];
        for(int kk = 0; kk < num_samples; kk++) {
            output[kk] = cpx_output[kk].getReal();
        }

        return output;
    }

    // imitates MATLAB diff command. takes the difference between elements of a vector
    protected double[] calculateDiff(double[] input) {
        double[] output = new double[input.length - 1];

        for(int kk = 0; kk < input.length - 1; kk++) {
            output[kk] = input[kk+1] - input[kk];
        }

        return output;
    }

    // return the computed workspace volume
    public double getWorkspaceVolume () {

        return this.fitmeasures[0];
    }

    // return the computed XY plane area
    public double getXYplane() {

        return this.fitmeasures[1];
    }

    public double getYZplane() {

        return this.fitmeasures[2];
    }

    public double getXZplane() {

        return this.fitmeasures[3];
    }

    public ArrayList<double[]> getPosition() {
        return this.position;
    }

    public double[] getMaxpositions() {
        return this.maxpositions;
    }


    // calculate the z-score of data
    public static double[] calculateZScore(double[] data) {
        double [] output = new double[data.length];
        double popmean = StatUtils.mean(data);
        double popstd = Math.sqrt(StatUtils.variance(data, popmean));

        for(int kk = 0; kk < data.length; kk++) {
            output[kk] = (data[kk] - popmean) / popstd;
        }

        return output;
    }

    // return all the values between the middle prctile-th percentiles of the data
    public static double[] calculateThresholdPrctile(double[] data, double prctile) {
        boolean[] output_temp = new boolean[data.length];
        int newlength = 0;

        double edgeprctile = (1-prctile)/2;
        double thresholdlow = StatUtils.percentile(data, edgeprctile),
               thresholdhigh = StatUtils.percentile(data, 1-edgeprctile);


        // find data within bounds
        for(int kk = 0; kk < data.length; kk++) {
            if(data[kk] >= thresholdlow && data[kk] <= thresholdhigh) {
                output_temp[kk] = true;
                newlength++;
            } else {
                output_temp[kk] = false;
            }

        }

        // copy data between thresholds to new array
        double[] output = new double[newlength];
        int counter = 0;
        for(int kk = 0; kk < data.length; kk++) {
            if(output_temp[kk]) {
                output[counter] = data[kk];
                counter++;
            }
        }


        return output;
    }
}

```