Expressive Touch: Using a Wrist-worn Inertial Measurement Unit to add Expressiveness to Touch-based Interactions

Author: Gerard Wilkinson
Supervisor: Patrick Olivier
Student No.: 100972267
Newcastle University
Abstract

Interaction with touch screens has always been a very intuitive and simple interaction to perform. As smartphones and tablets have penetrated the worldwide market users are now very familiar with this kind of interaction. The interaction however lacks expressiveness. Users are limited in how they can affect their touch interaction due to the lack of information from touch screens about hand characteristics and pre- or post-touch information. Recent advances in smart watch technologies, together with increased sales over recent years, have provided developers with more information about the touch interaction through the movement sensors present within smart watches. Expressive Touch aims to provide a system which takes advantage of this new source of information from wrist-worn sensors by adding expressiveness to the touch interaction on smartphones and tablets. This project proposes a set of Expressive Touch interactions which can be detected from a wrist worn device with inertial measurement sensors. These interactions provide developers with a new set of events and interaction modalities which can incorporated into their applications and games.
Declaration

“I declare that this document represents my own work except where otherwise stated.”

Gerard Wilkinson

Acknowledgements

I would like to thank Patrick Olivier, Gavin Wood, Daniel Jackson, Nils Hammerla, Bradley Pursglove, Steve Hodges & Jonathan Hook for their support and guidance throughout the project. Also the members of the Culture Lab who agreed to participate in the evaluation of the project.
# Table of Contents

Abstract 2  
Declaration 3  
Acknowledgements 3  
Table of Contents 4  
Figures & Tables 6  
1. Introduction 7  
  1.1 Motivation & Rationale 7  
  1.2 What is Expressive Touch? 8  
  1.3 Aims 9  
  1.4 Objectives 10  
  1.5 Outline 10  
2. Background 12  
  2.1 Joint Interaction Research 12  
  2.2 Hand Characteristics Research 13  
  2.3 Other Relevant Research 15  
  2.4 Smartwatches & Smartbands 16  
  2.5 Summary 18  
3. Interaction Design & System Design 20  
  3.1 Design Space 20  
  3.2 System Diagram 22  
  3.3 Research Methodology 23  
  3.4 Interactions 24  
  3.5 Hand Model 25  
  3.6 Technologies 26  
  3.7 Development Process 27  
  3.8 Demonstration Ideas 27  
4. Implementation 29  
  4.1 Application Development 29  
  4.2 IMU Analysis & Sensor Fusion 30  
  4.3 Live Data 31
4.4 Plane Model 31
4.5 Maps demo 32
4.6 Controls demo 33
4.7 Expressive Touch Video 34
4.8 Interaction Detector 34
4.9 Testing Strategy 39

5. Evaluations 40
  5.1 Evaluation Techniques 40
  5.2 Technical Evaluation 40
  5.3 Results 46

7. Discussion & Conclusions 53
  7.1 Aims Evaluation 53
  7.2 Evaluation Issues 53
  7.3 Information Overload 54
  7.4 Expressive Touch as a Product 55
  7.5 Future Developments 56
  7.6 Conclusions 57

References 58

Appendices 61
  A - Evaluation Participants 61
  B - Specific Force Category Evaluation 61
  C - Categorised Tap Force Evaluation 61
  D - Roll Evaluation 61
  E - Pitch Evaluation 61
  F - Flick Evaluation 61
  G - Code 61
Figures & Tables

Table 2.5.1 - Smartwatch & Smartband comparison

Figure 3.2.1 - System Diagram illustrating the flow of data and events.

Figure 3.4.1 - Touch/Flick Force (Orange), Roll (Green), Pitch (Blue)

Figure 4.2.1 - WaxData.swift 23-26

Figure 4.3.1 - Live data view allows for analysis of data returned from sensor for feature detection.

Figure 4.4.1 - The plane model provides a simple and accurate model of the current hand orientation.

Figure 4.5.1 - Maps demo using pitch and roll changes to manipulate a 3D map view.

Figure 4.6.1 - Controls demo showing Expressive Touch control and image rotation.

Figure 4.6.2 - Apple Control centre showing large brightness and volume sliders, could be condensed with Expressive Touch controls.

Figure 4.8.1 - InteractionDetector.swift 11-45

Figure 4.8.2 - InteractionDetector 51-70

Figure 4.8.3 - InteractionDetector.swift 72-112

Figure 4.8.4 - InteractionDetector.swift 114-132

Figure 4.8.5 - InteractionDetector.swift 134-160

Figure 4.8.6 - InteractionDetector.swift 178-209

Figure 5.2.1 - Specific Tap Force Evaluation

Figure 5.2.2 - Categorised Tap Force Evaluation

Figure 5.2.3 - Rotation Evaluation

Figure 5.2.4 - Pitch Evaluation

Figure 5.2.5 - Flick Evaluation

Figure 5.3.1 - Specific tap force

Figure 5.3.2 - Median touch force with 20:80 percentile

Figure 5.3.3 - Time to rotate graph

Figure 5.3.4 - Rotate task completion accuracy graph

Figure 5.3.5 - Time to pitch graph

Figure 5.3.6 - Pitch task completion accuracy graph

Figure 5.3.7 - Median flick force with 20:80 percentile graph
1. Introduction

1.1 Motivation & Rationale

**Touch**

Smartphones and tablet touch screens provide users with an obvious and intuitive interaction with the information displayed on the screen. Touch screens have made advanced mobile interaction a possibility for the vast majority of users, it also provides many advantages in terms of portability and ensuring that the maximum possible screen real estate is achieved based on the device size.

While the touch interaction available to developers is simple and accurate it lacks expressiveness. Performing complex interactions on a touch screen often requires the use of multi-touch gestures which can often feel complex and unintuitive in comparison. Smartphone and tablet games are a good example of where touch screen interactions are often considered lacking the required control. Complex interactions that need to be performed quickly within games often leave users frustrated at the lack of fine control. This in turn raises the skill entry barrier to many of these games reducing sales, numbers of active users and revenue. The reason often attributed to the reason for such games failing to provide users with an adequate interface is that many of these games lack realistic inspiration. In the vast majority of cases the developers are trying to emulate a game which works well with a gamepad where users have a range of physical buttons and analog sticks available, which immediately provide haptic feedback to the user, something which is absent from the traditional touch interaction. This is of particular issue when mapping analog sticks to a touch screen where users have no longer have any feedback relating to the position of the stick and hence often lose track of the centre of the on screen control.

Recent improvements proposed by Apple through the addition of gamepad support in iOS 7 demonstrates a willingness from developers and manufacturers to find a solution to the issue. However these solutions require users to purchase specialist hardware and keep this hardware with them to play. Adding expressiveness to touch based interactions provides developers with new interaction modalities which can be mapped to a wide variety of responses such as manipulating 3D views, rotating of objects or modulation of actions.

**Smartwatch/Smartbands**

Smartwatches are in a rapidly growing market with over 3 million smart watches sold in 2013 and sales expected to increase vastly in response to new releases from Apple with the Apple Watch and Android Wear partners. [13] Inertial Measurement Units present in smartwatches and smart bands are primarily used for tracking of activities, for instance accelerometer tracking provides an estimation of steps as well as an estimation of exercise intensity. The sensors present in these devices can also be used...
to provide information about a users hand and wrist movements which has not previously been investigated.

Investigation into what can be determined from a wrist-worn sensor expands the potential application scenarios of a smartwatch/smartband the purpose of which has been called into question recently. These new devices provide us with a new source of information about the movement and orientation of the wrist, the accelerometers and gyroscopes present in many of the smart watches provide us with this information.

Expressive Mediums

There are many examples of expressive mediums which demonstrate the power of adding expressiveness to touch. Playing the piano, this can be considered an Expressive Touch interaction. Pianists can determine the loudness of a note by striking the key with a specific force, providing a new aspect to the touch interaction. This allows for much greater control of the piano and hence a wider range of music can be played. A similar scenario applies to playing the drums.

A painter has a vast catalog of brush techniques which effect the paint on a canvas, the pressure applied, the rotation of the brush and the manner in which they lift away from a canvas provides one of the most expressive mediums available. Looking at the brush as a touch interaction we can see the power of providing new information about the touch (force, roll, pitch & flick), which is needed to make such an expressive medium translate into its digital equivalent.

Striking of a football is another example where the way in which the ball is struck determines how the ball moves. A footballer has a great amount of control over speed of the ball with the force the ball is struck, the movement of the ball through the air applying bend and swerve to a kick by the orientation of the foot and angle of the strike.

1.2 What is Expressive Touch?

As we can see expressiveness is an important aspect of the touch interaction which is not captured by smartphone and tablet touch screens. Expressive Touch aims to use the inertial data from smart watches to provide expressiveness to the touch interaction on smartphones and tablets. This would allow users to perform interactions above the screen moving onto the display, starting on the screen and moving off, as well as being able to detect extra information about the touch interaction which was not available before.

The addition of expressiveness to the touch interaction on smartphones and tablets will provide developers with a set of new interaction modalities which allow for much greater freedom for users to express themselves through a touch interaction which has traditionally been basic and lacking detail. The potential applications for Expressive Touch are vast, both in terms technical aspects of the new data available to develop new applications and to improve current interaction modalities which can often be
considered awkward and unintuitive. A classic example of the latter is touch-based
games, specifically games which require complex and fast manipulation of a 3D world
or objects. Expressive interactions may provide an intuitive, fast and simple solution to
this problem.

The purpose of this system would be to leverage this extra information available from
smartwatch and smartbands to produce a system to detect Expressive Touch
interactions allowing developers to respond to interactions. As this project will involve
the use of a wrist-mounted sensor to perform these interactions, Expressive Touch will
also ensure that users can take advantage of this system without the need for specialist
equipment. Many smartwatches/smartbands provide the required information to
perform the expressive interactions proposed in this project. This was an important
consideration during the project to ensure that the proposed interactions and
technologies used were already available to the user and would not require any addition
or modification of hardware.

1.3 Aims

To explore the integration of wrist worn movement sensing and tablet devices.

An exploration of the what can be determined about the touch interaction from a wrist-
worn movement sensor will be performed. To analyse the sensor data returned and
trialiasing various detection techniques for a wide range of possible expressive
interactions.

To investigate possibilities for adding expressiveness to touch interactions using
data from the sensor.

Looking at current touch interactions and gestures which are available to developers
and performing an evaluation of what is possible when adding expressiveness to these
interactions will be performed. This will be performed by looking at current single touch
interactions and investigating information that is currently missing from those
interactions. Additionally looking at the wide range of multi-touch interactions which can
be potentially mapped to an Expressive Touch interaction such as the two-finger
rotation gesture.

To develop a system for the application of this research to other platforms or
applications.

Production and development of this system will be performed in such a way as to make
it as available as possible to other platforms and applications. The selection of
technologies to be used in this project will be performed to consider making this
research open to repeatability and application of this system to other platforms and
applications.
1.4 Objectives

To produce an iOS application to demonstrate some Expressive Touch interactions with an Axivity WAX9 sensor and an iPad.

This objective relates to the first aim of exploring what possible applications there are for these Expressive Touch interactions. The produced application will contain the required interaction detection system along with a range of demos to show possible application scenarios for Expressive Touch. This will involve mapping each of the events and metrics to be developed onto manipulation of elements on screen.

To develop a set of Expressive Touch interactions that can be accurately detected using data from the sensor.

This objective relates to the second aim to identify what Expressive Touch interactions can be detected. Each interaction developed will be carefully considered to ensure that it can be accurately detected from the wrist-worn sensor. This is a one of the most important stipulations of this research. Users and hence developers will not take advantage of these new interaction techniques if they are: difficult to perform, can be performed accidentally or do not appear to match a user’s apparent motions.

To produce an API/Framework for Expressive Touch interactions.

This objective relates to the third aim allowing for a simple application of this research to other applications and platforms. Development of the interaction detection system will involve careful design to ensure abstraction of the underlying detection algorithms. Providing a simple and intuitive framework for wrapping around the Expressive Touch functionality. Development techniques such as the modular design guide will be employed to ensure that there is a proper level of abstraction of information away, allowing developers to employ Expressive Touch as a ‘black box’. Touch events and sensor data are passed and events are returned. An event subscription system will provide developers with a simple method for determining when events occurred.

1.5 Outline

In the second chapter of this document, relevant work and the current market will be explored. Investigating what work has currently been performed and taking inspiration from previously researched and proposed interaction techniques. Also as part of this section, an investigation into current smartphones, tablets, smartwatches and smartbands. Looking at the technology offered by these devices and investigating the restrictions placed upon the Inertial Measurement Unit data available from them.

The third and fourth chapters cover the design and implementation process used to develop Expressive Touch. The design chapter outlines how the proposed interactions were designed through a description of the research methodology of the project. The implementation chapter covers the underlying implementation techniques and technologies used to achieve the proposed interactions.
The fifth chapter of this document covers the evaluations performed to investigate users' ability to perform interactions. It covers the evaluation techniques detailing the choice of evaluations followed by a detailed description of each of the five evaluations performed. Finally, the results from these evaluations are shown and discussed in depth, looking at which evaluations were considered to show the interaction being successful and which were considered difficult by users. The evaluations also provided a large amount of data which was used to determine interaction detection thresholds for, for instance: soft, medium and hard presses.

Finally, the sixth chapter provides discussion and conclusions. An evaluation of the aims of this project is performed. Along with a detailed discussion of the issues users faced in the evaluations and limitations of Expressive Touch. Also discussed are the practical and technical limitations involved with Expressive Touch in a real-world scenario and future work that could be performed to expand Expressive Touch. This is followed by some final conclusions.
2. Background

This section covers related work and makes comparisons with the work performed as part of this project. Much of the work looked at has been used to provide inspiration for the proposed Expressive Touch interactions.

2.1 Joint Interaction Research

Research into what is possible when combining touch interactions with information about the behaviour before the touch interaction has covered a wide range of possible application scenarios. Duet proposed an interaction system using smartwatch to augment the touch interaction.

In this paper Chen et al investigate the possible interactions between a smart watch and a smartphone. The authors use a Sony smartwatch and a Sony smartphone for their testing purposes. This watch only contains an accelerometer which limits the possible gestures and interaction techniques to smaller range.

The proposed interaction techniques were divided into distinct categories based upon the state of the two devices based upon Falk’s research on foreground-background frameworks. Interactions which involved the smart watch and the smartphone being in the ‘foreground’ were further sub-divided into two distinct areas: smartwatch - smartphone dual screen interactions and using the smart watch to augment the touch interaction on the smart phone. Expressive Touch will be considering the latter forms of interaction so I will focus on their research and conclusions.

Chen et al demonstrated a simple example of their augmentation of the touch interaction using a simple text editor. They were able to use the current orientation of the wrist to infer the orientation of the hand with a high degree of accuracy provided the user interacted with the screen with their hand outstretched and using their fingers rather than their thumb. The demonstration showed an application which allowed users to: draw on a document with the pad of their finger, select text with the side of their finger and highlight text with their knuckle. This could be detected as each of these interactions required rotating of the wrist. With the hand movement therefore the application could determine which action to perform by looking at which axis of the accelerometer gravity is currently acting upon. [2]

An important consideration for Expressive Touch are the possibilities for multi-user interactions. However the immediately apparent issue with introducing multi-user and hence multi-sensor support is matching a sensor to a touch point. This type of identification also potentially has implications for identification of which wrist the sensor is on when interacting with the screen. If it is possible to identify if the user is interacting with the screen using the wrist with the sensor on or not, then this could extend the interaction allowing users to determine if they would like to use Expressive Touch or not simply by choosing their interaction hand. The work by the Digital Interaction Group at Newcastle University using wrist-worn inertial measurement to assign touch points to
specific hands (based upon the sensor readings from their wrist) could enable Expressive Touch with multiple users. This also identifies if users are using the hand with their wrist on or not, the system was called TouchI.

Identification of multiple users interacting on a tabletop surface has previously had substantial practical implications. Their proposed system used a wrist-mounted IMU to detect which user is performing which gesture on a table top surface. An issue highlighted was difficulties in identifying small or subtle gestures as the signal produced by these gestures was not strong enough to identify them. This is an interesting observation that must be taken into account when designing expressive interactions. Considering the possibilities for using a system such as this to identify users or the hand being used for Expressive Touch the authors discovered several key practical issues which might also cause issues for Expressive Touch. The most prominent of these was that identification was not immediate. This is due to the fact that at the initial touch there is not enough information available from the touch screen to map the IMU data to. [11]

2.2 Hand Characteristics Research

Investigations into determining characteristics of the hand and researching possible applications of said characteristics both to touch and to other object manipulation has been performed extensively through the use of hand tracking systems using a wide range of technologies. Marquardt et al investigated a wide range of possible application scenarios involving using hand characteristics to augment the touch interaction called the continuous interaction space. The concept of the continuous interaction space developed by the authors proposes that current interaction techniques in the form of table top surfaces and hand tracking techniques allow us to perform touch based interactions (touch, swipes, dragging, etc.) and hand gesture interactions (pointing, grasping, etc.).

They propose a system for interactions that can be performed: on the display, above the display and between the two. A table top surface tracks touch interactions for gestures that can be performed on the display. The data obtained provides touch position and movement of the touch point across the screen. The tracking of the hand is performed by a Vicon hand tracking system is used to track gestures which are performed above the display. The system can track hand movement and finger yaw, pitch and roll angle. Processing of the data from these two systems is used to detect a set of proposed gestures.

Marquardt et al suppose that their system breaks down the arbitrary restrictions placed between hand gestures above a display and gestures on a touch screen. The work performed is comparable to Expressive Touch. Many of the proposed gestures provide a great source of inspiration for some Expressive Touch interactions.

Issues that were highlighted by the authors of the paper were a lack of possible application scenarios. The equipment required for the proposed system is both
expensive and large. As part of this project an iOS device will be used along with a wrist mounted sensor emulating a smart watch. The project will therefore be using devices which are both widely available and inexpensive. [1]

An interesting area possible for investigation as part of this project was looking at the detail which you can detect about the hand from a wrist mounted IMU. Information such as hand posture and finger position as well as details such as what object has been grasped could provide Expressive Touch with another level of information to be exploited for interaction detection.

The work by Morganti et al covered each of these aspects in detail using a wrist mounted smart watch with embedded sensors. Tendon sensors placed in the wrist band over the relevant tendons were used to detect hand posture and finger position as well as an RFID scanner on the band to detect which object was grasped by reading the RFID chip on the object.

The authors discovered that the system worked and could detect hand posture and finger position with a high degree of accuracy. However this required careful placement of each of the sensors on the wrist over each tendon and they would often require repositioning during demonstrations. This would therefore cause issues in real world applications as the user would require knowledge of how to align the band each time.

The detection of objects also faced significant issues during testing. Morganti et al discovered that the detection range of the RFID scanner was a major problem. As the sensor was mounted on the wrist objects held in the hand were too far from the sensor to be read. A similar project called iBracelet achieved readings of 10 or 12 cm (other research proposed using solenoid antenna integrated in the antenna of the bracelet to improve reading ranges, however they only achieved reading ranges of 9 cm). The authors proposed using a camera to detect objects which would not suffer from the same range issues. They would however require line of sight to the object.

Expressive Touch would benefit from the ability to detect hand posture and finger position however as shown from this research current technology to detect this information from the wrist is currently too error prone and infeasible to incorporate into a production unit. [3]

Tracking characteristics about the hand has been thoroughly researched with a wide range of tracking technologies and applications scenarios proposed. Glove based tracking of the hand is an area of this which has been extensively explored for some time as a solution to tracking all characteristics of the hand (posture, finger position, palm orientation etc.) with an object which is familiar and simple to end users. Research in 1994 carried out by Sturman et al investigated a wide range of glove based hand tracking technologies and highlighted some key considerations both when evaluating hand based tracking systems and provided some key areas for comparison when concluding the potential applications and practicalities of Expressive Touch.
Sturman et al identified several categories of hand tracking technologies that are broadly applicable today, including optical tracking technologies and marker systems. A modern example of optical tracking would be Kinect and marker systems are widely used in film making. The characteristics and applications of hand tracking for example interpreting sign language for text input and for musical performances could form part of the demo applications produced this project. [5]

2.3 Other Relevant Research

Expressive Touch provides supplementary information about the touch interaction. Investigating the possibilities for application of these interactions, recent work by the Digital Interaction Group at Newcastle University into detecting administrators through IMU data from the wrist and matching said information to the touch interaction, SwipeID.

This paper proposes a system for authenticating an administrator using a wrist mounted Axivity WAX9 sensor. The administrator performs on-screen swipe gestures to authenticate on the device with the sensor data from the wrist corroborating the fact that the admin is the one performing those gestures. The chosen touch gestures provided a distinct detectable pattern in the IMU data which could then be used to authenticate the admin.

Many of the advantages discussed by the authors relating to the system are applicable to Expressive Touch. The wearable sensor is an inexpensive solution that does not require modification to the current device such as adding NFC or a fingerprint scanner such as TouchID.

Much of the data provided in the testing of the solution was interesting, particularly the data about the complexity of gestures. An important consideration for Expressive Touch is ensuring that gestures are not too complex. If the time to complete them well exceeded an equivalent touch interaction achieving the same result, users in this study were left frustrated by authentication gestures that took too long.

The learning curve involved with learning how to perform an interaction was considered in this project with the study showing that: discrete gestures showed higher average correlation, shorter average time to completion and also a lower number of failed attempts. The data provided showed that each of the Expressive Touch interactions developed must also consider this. While the proposed system is not directly related to Expressive Touch, it provides insightful information about gesture detection from a wrist-worn IMU. Along with details of evaluation participants ability to perform interactions categorised by the complexity of the gesture requested and the error rates. [10]

One of the aims of this project was to provide a framework for developers to take advantage of the system proposed through a simple and intuitive interface. Research by Kim et al proposed a gesture authoring framework for using multiple wearable
sensors for providing multi-device interactions. The research methodology and proposed methods for encapsulating the sensor data providing end-points for developers provided some excellent inspiration during the design and development process of Expressive Touch’s framework.

M-gesture proposed by Kim et al is a very interesting concept involving sensor fusion across a range of devices providing developers and end users with a framework for authoring of gestures, without detailed knowledge of the underlying sensor data. The authors made several attempts to create joint-gesture interfaces using multiple wearable devices. They did however require specific device combinations, application and contexts. M-Gesture provides a gesture authoring framework that is not reliant on textual based programming allowing users unfamiliar with this to create their own gestures.

The concept of a framework encompassing the data from wearable devices is one which has been incorporated into Expressive Touch. Developers can subscribe to events with callbacks when the subscribed interaction has been performed. Therefore abstracting away the underlying data analysis operations being performed by Expressive Touch. Expressive Touch currently contains a gesture recorder to aid development purposes. This allows for recording of the data from the IMU and the touch interactions on the smartphone or tablet. Further work could explore allowing developers or users to define their own gestures. [9]

2.4 Smartwatches & Smartbands
Sensors present in smartwatches and smartbands which have recently seen a surge in sales and market adoption provide a wide range of information about the wearer from heart rate to step count. Inertial Measurement Units present in these devices contain accelerometers and gyroscopes providing movement information about the wearers wrist which are used for various functionalities provided by the devices. For instance gyroscope readings are used to turn on the screen on the Apple Watch when users rotate their wrist into their view.

Apple Watch
The Apple Watch is the latest addition to the smartwatch market. It is one of the most advanced products discussed here. The device contains both an accelerometer and gyroscope however Apple currently do not provide access to the data from these sensors through their API. Therefore any such application which takes advantage of this data currently would have to go through un-official routes (jailbreaking) to gain access. The reason for this would appear to be due to battery issues and there is a possibility Apple could open access with a software update.

The hardware present in the Apple Watch makes it technically capable of supporting Expressive Touch but due to API restrictions this is not currently possible. [16]
Moto 360
The Moto 360 was one of the first popular smartwatches to come onto the market. The device contains an accelerometer and gyroscope however they are not mentioned on Motorola website so access to these values is unclear at this point without testing.

The hardware present in the Moto 360 allows for full Expressive Touch provided the data is available from the Android Wear SDK. [21]

Pebble Time
The Pebble Time is the latest offering from Pebble which has been one of the most successful smartwatches to date. The device contains an accelerometer but no gyroscope. Pebble do however have a good API for accessing all the sensor data.

The hardware present in Pebble Time allows for force Expressive Touch but not full Expressive Touch as there is no gyroscope for roll and pitch change detection. [22]

FitBit Surge
The FitBit Surge is a smartwatch/fitness tracker tailored for the fitness market. The device contains both an accelerometer and gyroscope. However FitBit do not give direct access to sensor data on their devices.

The hardware present in the Surge allows for full Expressive Touch but due to API access restrictions this is not currently possible but could be made available through a software update. [17]

Jawbone UP3
The Jawbone UP3 is a new fitness tracker which is purely designed for health tracking. The device contains an accelerometer but no gyroscope. Jawbone does not provide access to the sensor data through their API.

The hardware present in the UP3 allows for force Expressive Touch but due to API access restrictions this is not currently possible but could be made available through a software update. [18]
Device Table

<table>
<thead>
<tr>
<th>Device</th>
<th>Accelerometer</th>
<th>Gyroscope</th>
<th>API Access</th>
<th>Potential for ET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple Watch</td>
<td>✔️</td>
<td>✔️</td>
<td>No access to sensor data through API.</td>
<td>Not possible.</td>
</tr>
<tr>
<td>Moto 360</td>
<td>✔️</td>
<td>✔️</td>
<td>Android Wear SDK provides access.</td>
<td>Full Expressive Touch.</td>
</tr>
<tr>
<td>Samsung Gear 2</td>
<td>✔️</td>
<td>✔️</td>
<td>Android Wear SDK provides access.</td>
<td>Full Expressive Touch.</td>
</tr>
<tr>
<td>Microsoft Band</td>
<td>✔️</td>
<td>✔️</td>
<td>Microsoft Band SDK provides access.</td>
<td>Full Expressive Touch.</td>
</tr>
<tr>
<td>Sony SmartWatch 3</td>
<td>✔️</td>
<td>✔️</td>
<td>Android Wear SDK provides access.</td>
<td>Full Expressive Touch.</td>
</tr>
<tr>
<td>LG G Watch R</td>
<td>✔️</td>
<td>✔️</td>
<td>Android Wear SDK provides access.</td>
<td>Full Expressive Touch.</td>
</tr>
<tr>
<td>Pebble</td>
<td>✔️</td>
<td>✘</td>
<td>Pebble SDK provides access.</td>
<td>Force Expressive Touch.</td>
</tr>
<tr>
<td>Pebble Time</td>
<td>✔️</td>
<td>✘</td>
<td>Pebble SDK provides access.</td>
<td>Force Expressive Touch.</td>
</tr>
<tr>
<td>Asus Zenwatch</td>
<td>✔️</td>
<td>✔️</td>
<td>Android Wear SDK provides access.</td>
<td>Full Expressive Touch.</td>
</tr>
<tr>
<td>FitBit Surge</td>
<td>✔️</td>
<td>✔️</td>
<td>No access to sensor data through API.</td>
<td>Not possible.</td>
</tr>
<tr>
<td>FitBit Charge</td>
<td>✔️</td>
<td>✘</td>
<td>No access to sensor data through API.</td>
<td>Not possible.</td>
</tr>
<tr>
<td>Jawbone UP24</td>
<td>✔️</td>
<td>✘</td>
<td>No access to sensor data through API.</td>
<td>Not possible.</td>
</tr>
<tr>
<td>Jawbone UP3</td>
<td>✔️</td>
<td>✘</td>
<td>No access to sensor data through API.</td>
<td>Not possible.</td>
</tr>
</tbody>
</table>

Table 2.5.1 - Smartwatch & Smartband comparison

[16, 17, 18, 19, 20, 21, 22, 23, 24]

2.5 Summary

An in-depth look at the current research which has been performed around touch, smartwatch and joint interactions along with research covering other hand posture detection shows the level of detail of which each area has been researched. Explorations of joint interactions have proved more effective and provided a wide range of possible interaction applications. Particularly the continuous interaction space which demonstrated the possibilities for joint interactions between above the display interactions and on-display interactions. However this system was limited by the equipment being used being both expensive and large.

Research into hand characteristics that can be determined from a variety of sources including camera based tracking and wrist-worn tendon sensors have shown what can be detected about characteristics of the hand and possible application scenarios of
said characteristics. Evaluations showed the pronounced effect of issues in detection of hand posture and issues surrounding careful positioning of sensors being frustrating to users.

Looking at the smartwatches and smartbands currently available the sensors required to perform Expressive Touch interactions on smartphones and tablets are present in the vast majority of devices. Enabling users to take advantage of expressive interactions with a device that many already own. However a current issue highlighted in the device table above is the lack of API access provided by many manufacturers for battery conservation reasons.

In conclusion the current research revolving around joint-interaction or cross-device interactions has not focussed on adding expressiveness to the touch interaction. Using the IMU data available from wrist-worn sensors which are readily available to augment the touch interaction will be explored in this project. Looking at the current interaction modalities available and taking inspiration from current expressive mediums to provide developers and users with new Expressive Touch interactions.
3. Interaction Design & System Design

3.1 Design Space

Development of the Expressive Touch interactions covered two distinct areas: discrete interactions and continuous interactions. Discrete interactions involved a specified sequence of events occurring. For interactions to be detected for instance a sweep interaction where the user would sweep in from one side of these screen make contact with the screen and sweep off the other side.

The interactions developed can be further subdivided into 3 distinct sub-sections of the interaction: before the touch interaction, during the touch interaction and after the touch interaction. The development of each interaction comes down to a set of underlying metrics and events which can be combined or used independently to manipulate objects on the screen or modify interactions with an application. Events are divided up into pre- during and post- touch, different metrics are available at each stage.

Pre-Touch

Sensor data from before the touch interaction can be used to determine instantaneous force of the wrist. This information can be determined from accelerometer data showing acceleration and deceleration data from the wrist. Other information available pre-touch includes wrist orientation (yaw, pitch and roll) from the gyroscope, however this information is not currently included in Expressive Touch as it would require user calibration before each interaction, which was considered an unacceptable requirement.

Combining this data with information already available from touch screens allows for calculation of touch force at the point of touch. Force is the only metric available pre-touch. The value returned from this is an arbitrary value denoting the force with which the screen was struck. Examples of this metric being used include: control of the force with which a ball or other object is thrown or altering the functionality of buttons such as a keyboard (where letters are capitalised if the screen is hit hard and lower case if the key is struck soft).

During-Touch

Continuous interactions during the touch interaction involved using metrics such as pitch & roll to inform a continuous interaction with the touch screen. For example, a user pressing and holding a button and twisting to confirm the interaction. These continuous interactions allow users to add further expressiveness to an otherwise static interaction. Providing new metrics such as pitch and roll during the touch interaction opens up a wide range of possibilities for extending the existing touch interaction.

Designing applications around these continuous interactions allows for much greater freedom of expression when: manipulating views, adjusting controls and more. In many
cases the application of such interactions provides a much greater level of dexterity to users.

Post-Touch

After the touch interaction sensor data again can be used to determine instantaneous force of the wrist. Acceleration off the screen can be calculated from accelerometer data from the sensor. Again gyroscope data can provide yaw, pitch and roll after the touch interaction but this information is not available at the moment. Looking at the end of the touch event, using data from the touch screen and window of sensor data after the touch event, allows us to determine the acceleration away from the screen which can be interpreted as flick force.
3.2 System Diagram

Smartphone or Tablet
Touch events from smartphone/tablet touch screens are passed to the application for processing. Touch event data recorded involves number of touch points and touch position for each point. This data enables tracking of multiple fingers as they move across the screen.

Application
The application running on the smartphone/tablet receives touch events and the sensor data from the Bluetooth connection with the smartwatch/smartbands IMU. Applications taking advantage of Expressive Touch pass the touch events and sensor data to the
interaction detector. The application then subscribes to callbacks for the required events from the interactions detector. These callbacks are then attached to relevant actions to be performed within the application.

**Interaction Detector**

The application analyses the touch data and sensor data with the aim to use the sensor data to inform new parameters to the touch interaction. The interaction detector takes two inputs, the IMU data from the wrist-worn sensor and the touch events from the device touch screen. The IMU data and touch events are monitored based upon a set of pre-defined interactions. If an interaction is detected from the supplied data each of the callbacks subscribed for the relevant interaction are fired. Providing information about the touch interaction that was not previously available for example using accelerometer data from the smartwatch or smartband IMU to determine touch and flick force.

**Bluetooth**

Wireless communication between two devices is a battery intensive task which can be served by a wide range of protocols.

- Wi-Fi provides high bandwidth but at a high level of battery drain. Also the bandwidth available is unnecessary for the amounts of data being transmitted here.

- Bluetooth Serial Port Profile (SPP) provide communication with lower bandwidth and higher latency, however due to manufacturer restrictions access to this protocol is sometimes restricted.

- Bluetooth Low Energy (4.0) is very similar to SPP however the packet size is restricted reducing the bandwidth and lowering the power consumption.

IMU data from the smartwatch or smartband can be streamed using BLE to a smartphone or tablet with low latency and acceptable battery consumption.

**Smartwatch/Smartband**

IMU data from smartwatches and smartbands which are used for a variety of functions can be used to interpret some characteristics of the hand and wrist from the accelerometer (force) and gyroscope (rotational changes). This data can be streamed to the smartphone or tablet for processing.

**3.3 Research Methodology**

This project will propose a set of new interaction modalities for use on smartphones and tablets in conjunction with a wrist-mounted sensor such as a smartwatch/ smartband. The methodology used to develop the interactions will revolve around an evaluation driven design process. Through use of the design space outlined above an investigation into the sensor data will be performed to determine what characteristics of the hand movement can be interpreted from wrist-mounted IMU data.
Once these characteristics have been determined a range of interactions will be developed and evaluated which incorporate the hand characteristics into touch interactions in a meaningful and intuitive manner.

3.4 Interactions
Application of methodology outlined above resulted in the design and research of four Expressive Touch interactions which are detailed below.

**Touch Force**
At the point of touch there is a very limited set of data available to developers. Realistically the only information available to them is touch position. Force with which the screen was struck is a valuable metric which can be used a wide range of scenarios, for instance, piano and drum playing.

**Flick Force**
Flicking off the screen usually requires contact with the touch screen during the flick interaction. This requirement can often be limiting to expressiveness provided by this interaction especially when interacting with small screens where the real estate further limits this interaction. As such being able to determine flick force without contact with the screen provides an interaction with much greater range and freedom of expression.

**Roll**
During the touch interaction information about the twisting motion of fingers on the screen is unavailable. This has led to multi-touch interactions being developed to enable this type of interaction. While many of these interactions have become natural to users through extensive use of touch screen devices they lack the range or fine control offered by the equivalent single-touch twisting motion. Expressive Touch proposes taking advantage of the physiological constraint that users are unable to twist their finger without rolling their wrist to detect this interaction and provide information about the twisting of fingers on touch screens.

**Pitch**
When users interact with the screen the touch point is the only information captured by the screen as described above. However extending upon the roll interaction outlined above we can further decompose the pivoting interaction from a single touch point into pitch angles. Providing developers with a pitch angle allows for a simple and intuitive interaction for pivoting a 3D view.
Each of the interactions developed have been carefully designed to minimise false positive detections but to also be simple for users to perform. This was always carefully considered as it was important to ensure that the interaction being performed was not frustrating to the user or more difficult than the equivalent touch gesture.

### 3.5 Hand Model

The concept of a hand model was required to model some of the Expressive Touch interactions. Many of the interactions being developed assumed from the IMU data characteristics of the hand wearing the WAX sensor. As such the hand model is used to model such characteristics, specifically pitch and roll of the wrist.

The Madgwick code provides information which can be used to interpolate gravity. This vector can then be used to remove gravity from the accelerometer readings providing pure acceleration and using the gravity vector we can determine the sensors orientation with respect to a set up value. This can then be used to determine the pitch and roll of the sensor and hence the hand. [4]

Another consideration in development of the hand model was concerning the orientation of the device itself. The pitch and roll values should always be calculated relative to the orientation of the device being used, therefore allowing the user to perform gestures without having to consider the positioning of the device.

There are however some issues involved in development of gestures around the pitch or yaw of the hand. Considering the pitch of the hand, this can be completely detached from the movement of the wrist, the same applies to the yaw. You can move your hand without moving your wrist by a significant amount. Finger movement also means that interactions reliant on pitch may be difficult to capture accurately as the movement of the finger cannot be detected from the wrist.
3.6 Technologies

OpenMovement Axivity WAX9 IMU

The OpenMovement Axivity WAX9 sensor is a battery powered device which contains a 9-axis IMU with an accelerometer, gyroscope and magnetometer along with a barometer and thermometer. The device has a 56 day standby battery life and a 6 hour Bluetooth streaming battery at 50Hz sampling rate.

The device can be interfaced with through Bluetooth serial ports or Bluetooth Low Energy (4.0). Sampling through Bluetooth serial ports can return data from the sensor in text output which can then be parsed into the required values or more reliably and efficiently through a byte representation. Through the Bluetooth Low Energy (BLE) interface developers can subscribe to notifications or indications this terminology relates to how packet drops are handled. A notification is a packet sent by a BLE device to update subscribed clients of a change in the subscribed value, there is no acknowledgement sent by the receiver of this packet and therefore no resending of lost packets. An indication is similar to a notification except there is an acknowledgement of the packet sent by the receiver and missing packets are re-sent.

The BLE interface on the Axivity WAX9 requires developers to firstly write a value into a specific service to begin streaming of data. Then you can subscribe for notifications to a service which will provide updates of the sensor data through a single Bluetooth packet. This packet can then be read into an application and interpreted as the required sensor values. The reason for using notifications rather than indications for sensor data is if a Bluetooth packet is dropped then we no longer care about its contents as it would arrive out of sync containing old data. [28]

iOS Device

The application will be developed for iOS as it has a well developed and documented SDK and CoreBluetooth API. An initial issue with development for iOS related to Apple’s requirement that Bluetooth devices, that wish to communicate with an iOS device using Bluetooth Serial Ports need to be MFi certified devices. This is an expensive and arduous process that requires a special authentication chip installing in all such devices after a lengthy approval process. However Apple do not have such a restriction on devices that communicate using Bluetooth Low Energy to encourage the adoption of the technology and generation of cheap devices such as fitness trackers and smart home products that can interface with their products. [31]

As BLE was used to communicate with the OpenMovement Axivity WAX9 sensor the CoreBluetooth framework provided by Apple for BLE communications would be used extensively on the project. This framework was well documented and help was readily available making set up of the communication with the WAX9 sensor relatively easy.

Due to the universal application set-up provided by Xcode developing an application for one iOS device instantly made the application available across all iOS devices both
smartphones and tablets. During the project the primary development device will be Apple’s latest smartphone the Apple iPhone 6 Plus, however with very few adjustments the application could be adjusted to run on for instance an iPad with the only hardware requirement being support for BLE which has been available since the iPhone 4S and iPad 3. [29]

Language Choice

Apple’s recent launch of their new programming language Swift was something of personal intrigue to me. I decided to develop the application using this new language. While the language was new it is fully interoperable with Objective-C code therefore current frameworks and code examples already available can be easily used in the project. [29]

3.7 Development Process

The application for the demonstration of Expressive Touch was developed using the agile development methodology. The iterative development of the application and the interaction detection algorithms allows for gradual testing and improvement.

3.8 Demonstration Ideas

To demonstrate the power of Expressive Touch several demonstration scenarios were theorised. This first being a keyboard, iOS 8 allows developers to create 3rd party keyboards. This would allow users to perform a variety of actions that would previously have required multiple taps or gestures to complete.

The first could be a for accessing special characters on the keyboard, users could press and hold keys which would pop open a radial menu with options for special characters and numbers which were carefully selected for each key. Users could twist to select the option required and release to confirm. Further study could be performed, to potentially evaluate the capacity for this system improving typing rates as users become experts.

Other interactions were possible such as being able to flick off the delete key to delete a line of text. Users could also use a specified area of the keyboard to perform fine control of the cursor position. This is often something that many users struggle with and using a twist gesture to control the position of the cursor was extremely helpful to users in our tests.

Another potential application of Expressive Touch could be for presentation software. Users could define slide transitions which could be controlled through an expressive interaction. Users could potentially define interactions relating to specific transitions or could have fine control over the speed of transitions through a simple interaction such as twisting.

A maps demo application could be produced to demonstrate the pitch and roll with a continuous interaction. Users could pan, pitch and rotate the map view by pitching and
rolling their wrist, manipulating the map view. Google Earth and Apple Maps flyover
provide high quality textured 3D buildings which can be navigated currently using touch
interactions. However using these gestures to manipulate the map view can be
awkward and unintuitive. Being able manipulate a map view simply by pitching and
rolling the hand could provide a much more natural and intuitive interaction. [25, 26]

The applications of Expressive Touch in terms of gaming are broad, several
demonstration ideas were investigated for gaming. Taking advantage of the possibility
of absolute pitch and roll angles before the touch event which while not available in the
current Expressive Touch system could be expanded to support ideas such as a
football game similar to Flick Kick Football. [27] A game such as this where users can
bend the ball into the goal using their swipe direction but have no control over power or
shot style. Mapping the pitch and roll of the wrist to a foot essentially letting the user
use their finger as they would their foot, potentially provides a realistic and intuitive way
of playing football on a touch screen. Another gaming idea involving absolute pitch and
roll angles involved boxing games where the angle and hit force could be used to
perform different punches.
4. Implementation

4.1 Application Development

The Axivity WAX9 sensor is traditionally interfaced with using Bluetooth serial ports which are unfortunately restricted on iOS to only devices which are MFi approved. Instead the Expressive Touch application uses the Bluetooth Low Energy (BLE) interface of the WAX9 to stream data to the iOS device which does not require MFi certification. [30]

Data from the sensor is streamed to the iOS device and stored on the phone along with a time the data was received. The amount of data stored can be adjusted based upon the application, currently the limit is set to 10,000 for evaluation purposes. However the current Expressive Touch system only needs a maximum of 10 values. Initial testing has shown that even the latest iPhone 6 Plus recording 10,000 (roughly 1 minute of data) causes performance issues. This data is then shared across the various sections of the application, live data, interaction recorder, interaction detector etc.

The live data view uses CorePlot an external library for iOS for plotting data on a graph, the data for the graph is reloaded at regular intervals displaying the last 100 data values for performance reasons. [31]

An interaction detector screen was produced to show the current values being produced for force rotation and pitch changes. This screen is very useful for studying the IMU data based upon an interaction to be detected. During interaction detection sensor data and touch events were logged and the data could be exported in CSV format for further analysis. This proved extremely helpful when debugging issues with demos and evaluations. This information can be used to determine the effect of issues such as delay in data from the sensor and how the sensor data for each sensor looks when performing the interaction aiding development of the application of the required interaction.

The interaction detector for Expressive Touch tracks each feature in the sensor and allows developers to respond to events and retrieve information about hand roll, pitch and force. Features are distinct patterns in the IMU data which can be used to determine the type of actions being performed by the user. The rotation interaction calculates how much the finger has twisted on the screen using the gyroscope present in the wrist mounted IMU. Features such as detecting flicking off the screen are achieved by processing a window of data from the IMU from the point of lifting off the screen and a preset period for detection, this is needed as there is a delay in data being received by the iOS device.

One of the aims of this project was to produce a framework with which developers can incorporate Expressive Touch into their own application. This was achieved through the use of a self-contained class for interaction detection. The underlying idea behind class was to encapsulate the interaction detection through developers passing in the sensor
data and touch events which in turn would callback to developers subscribed callbacks when interactions occurred.

4.2 IMU Analysis & Sensor Fusion

Feature detection from the IMU data is based upon a window of data depending upon which feature is being detected. For instance the flick detection works by looking at each of the readings in a 100ms window after the touch is released. This type of window analysis is common in IMU data analysis, essentially the window of data is analysed for specific spikes or changes which can be inferred as or be used to calculate a specific interaction.

After investigation of possible interactions it was decided that Sebastian Madgwick’s sensor fusion code to determine the “best estimation” of the IMU orientation was required. This involved using an initial position vector and applying the quaternion produced from Madgwick’s algorithm to produce a transformed vector representing the IMU orientation. The angles between these vectors could then be decomposed into pitch and roll angles. As the WAX is wrist mounted the orientation information can be used to interpret the orientation of the hand allowing for a range of interactions involving hand orientation. [4]

In an ideal world acceleration data from the accelerometer would give you exactly the current acceleration of the wrist which can be interpreted into force. However gravity is constantly acting in one direction meaning that effectively there is always an extra 1 G in the accelerometer output affecting the results which could be spread across multiple axes in the output, as such we need to remove this. There are several techniques for estimation and removal of gravity, this projects uses the Madgwick estimation of gravity by decomposing the Madgwick quaternion into the gravity components x, y and z using this code (Figure 4.2.1). [4]

```swift
let gravx = 2 * (q.y * q.w - q.x * q.z)
let gravy = 2 * (q.x * q.y + q.z * q.w)
let gravz = q.x * q.x - q.y * q.y - q.z * q.z + q.w * q.w
grav = Vector3D(x: gravx, y: gravy, z: gravz)
```

Figure 4.2.1 - WaxData.swift 23-26

Once an estimation of gravity has been made then each accelerometer reading can be adjusted to remove gravity leaving the ‘pure’ acceleration of the wrist. Calculation of the touch force and flick force are both performed in a similar manner. An analysis of a window of data is performed 100ms before to the touch for touch force and after the touch ends to 100ms later for flick force. This window of data is analysed and the maximum magnitude vector is retrieved from that window and passed as an arbitrary value back to the developer to be used as a factor for view or object manipulation. In the case of the touch force the value returned is passed through 3 threshold checks (Soft, Medium & Hard) to determine which event to fire. A similar threshold is used for flick force to determine if a flick occurred or not.
4.3 Live Data

To analyse the data returned from the sensor in relation to touch events a live data view was created. The view presented the x,y,z values from each of the sensors on board the wrist-mounted IMU on a separate graph along with touch events. This allowed for analysis of the data being returned from the sensor to aid development of interaction detection algorithms. The data views were created using a library called CorePlot, the library created a graph within a provided UIView. [31] The graph could be customised to provide axes values, labels and data labels which were used to label touch points. A subscription to the sensor data cache provided updates to the graph when new data arrived from the sensor.

![Live data view](image)

Figure 4.3.1 - Live data view allows for analysis of data returned from sensor for feature detection.

4.4 Plane Model

Madgwick’s estimation of IMU orientation through sensor fusion provides a quaternion which can be used to model the IMU as a 3D model. To aid development of the interactions and to confirm the implementation of Madgwick’s a 3D model was produced taking in the quaternion and modelling it with a plane. A sample SceneKit project provided by Apple provided a model of a plane which could be easily modified to use the quaternion. [4]
4.5 Maps demo

Touch interactions with 3D views is traditionally an awkward interaction. It is exactly for this reason that the vast majority of successful smartphone and tablet games are in 2D or with a restricted 3D to enable simpler interaction. The 3D view interaction techniques offered by current platforms involve using a combination of multi-touch and single touch interactions which often feel disjointed and unintuitive. Manipulation of map views is an interaction that is often subject to a degree of frustration when attempting to pivot and rotate a map view.

As such an Expressive Touch map demo was produced to allow for manipulation of a 3D map view with building models using expressive interactions. This involved mapping the pitch and roll metrics during the touch interaction to the pitch and heading of the map camera respectively. This provided a dynamic and expressive interaction medium for manipulation of a 3D map view.
4.6 Controls demo

Settings and control panes on smartphones and tablets often require large amounts of screen real-estate to accommodate the fine level of control required on slider and dial controls. The controls demo was developed to demonstrate a control panel which is not limited by screen real-estate or at least reduces the limitation of this. Slider controls, a common value control, which require a large amount of screen real-estate can be replaced with a button which can be the size of a touch point. Users can then press down on the control they wish to set and twist their finger (roll interaction) to the required value.

![Figure 4.6.1 - Controls demo showing Expressive Touch control and image rotation.](image)

This demo shows how a control screen which previously required a large amount of screen real-estate such as Apple’s control centre on iOS can now be made much smaller by replacing the sliders with Expressive Touch value controls.

![Figure 4.6.2 - Apple Control centre showing large brightness and volume sliders, could be condensed with Expressive Touch controls.](image)
4.7 Expressive Touch Video

To demonstrate what is possible from Expressive Touch and illustrate each of the interactions and demonstrations described above a video was produced. This video is available from YouTube. [15, 33]

4.8 Interaction Detector

The implementation of Expressive Touch’s functionality comes down to a single class which processes the data and detects interactions the InteractionDetector. The purpose of the interaction detector was to encapsulate the functionality of Expressive Touch providing a single contact point for developers to pass data and receive subscription callbacks.

```swift
class InteractionDetector {
    var currentForce: Float
    var currentRotation: Float
    var currentPitch: Float
    var touchDown: Bool

    private var lastDataTime: NSTimeInterval!
    private var metricsCallbacks: Array<(data: Float!) -> Void>
    private var flickedCallbacks: Array<(data: Float!) -> Void>
    private var hardPressCallbacks: Array<(data: Float!) -> Void>
    private var mediumPressCallbacks: Array<(data: Float!) -> Void>
    private var softPressCallbacks: Array<(data: Float!) -> Void>

    private let dataCache: WaxCache
    private let touchForceFilter: Float = 0.1
    private let medForceThreshold: Float = 0.2
    private let hardForceThreshold: Float = 0.5
    private let flickThreshold: Float = 0.5

    init(dataCache: WaxCache) {
        self.dataCache = dataCache

        currentForce = 0.0
        currentRotation = 0.0
        currentPitch = 0.0
        touchDown = false

        metricsCallbacks = Array<(data: Float!) -> Void>()
        flickedCallbacks = Array<(data: Float!) -> Void>()
        hardPressCallbacks = Array<(data: Float!) -> Void>()
        mediumPressCallbacks = Array<(data: Float!) -> Void>()
        softPressCallbacks = Array<(data: Float!) -> Void>()
    }
    ...
```

This excerpt (Figure 4.8.1) shows the variable declarations and initialiser for the InteractionDetector. As you can see construction of the InteractionDetector simply requires a reference to the data cache where the sensor data is being stored for analysis.
At this stage a variety of constant thresholds are set for interaction detection. The values for these thresholds were set through experimentation and evaluation analysis.

```swift
func startDetection() {
    dataCache.subscribe(dataCallback)
}

private func dataCallback(data: WaxData) {
    currentForce = calculateForce(data)
    if (touchDown) {
        currentRotation = calculateRotation(data)
        currentPitch = calculatePitch(data)
    }
    lastDataTime = data.time
    fireMetrics()
}

func stopDetection() {
    dataCache.clearSubscriptions()
    clearSubscriptions()
}
```

The three functions shown here (Figure 4.8.2) are responsible for the processing of the sensor data as it comes back. Applications can start detection by calling the `startDetection()` function. This function then subscribes to updates from the WaxProcessor class which pushes new data that comes in from the sensor to all subscribers.

As you can see the `dataCallback(data:)` function shown is responsible for processing each bit of sensor data that comes back from the sensor. The sensor data is used to keep track of the instantaneous force, and if the user is currently touching down on the screen it also tracks rotational and pitch changes from the data. Finally it tracks the time the data was received followed by firing the metric callbacks which will be explained later.

The final function in this segment is the `stopDetection()`. This function should be called by developers before disposing of the InteractionDetection instance, it clears data callbacks and unsubscribes to the data cache callbacks. However if this method is not called it is called in the class destructor.

```swift
func touchDown(touchDownTime: NSTimeInterval) {
    touchDown = true
    let touchForce = calculateTouchForce(touchDownTime)
    let data = dataCache.getForTime(touchDownTime)
    data.touchDown(touchForce)
}
```
if (touchForce > hardForceThreshold) {
    fireHardPress(touchForce)
} else if (touchForce > medForceThreshold) {
    fireMediumPress(touchForce)
} else {
    fireSoftPress(touchForce)
}

func touchUp(touchUpTime: NSTimeInterval) {
    touchDown = false
    currentRotation = 0.0
    currentPitch = 0.0

    let data = dataCache.getForTime(touchUpTime)
    data.touchUp()

    NSTimer.scheduledTimerWithTimeInterval(0.1, target: self,
                                          selector: Selector("touchEndCallback:"), userInfo: touchUpTime, repeats: false)
}

func touchCancelled() {
    touchDown = false
}

@objc private func touchEndCallback(timer: NSTimer) {
    let touchUpTime = timer.userInfo as! NSTimeInterval
    let end = NSDate.timeIntervalSinceReferenceDate()

    let flickForce = detectFlick(touchUpTime, end: end)

    if (flickForce > flickThreshold) {
        fireFlicked(flickForce)
    }
}

...
private func calculatePitch(data: WaxData) -> Float {
    var totalPitch = currentPitch
    totalPitch += data.gyro.y * Float(NSTimeInterval(data.time - lastDataTime))
    return totalPitch
}

private func calculateForce(data: WaxData) -> Float {
    return data.getAccNoGrav().magnitude()
}

Figure 4.8.4 - InteractionDetector.swift 114-132

These functions (Figure 4.8.4) are used to calculate the metrics used during continuous interactions along with an instantaneous force calculation function. The rotation and pitch calculations take changes in the gyroscope x and y axes respectively and apply the rotational changes around these axes to the current rotation and pitch to produce a new estimation of the rotation and pitch relative to the initial touch. The force calculation takes the current magnitude of the acceleration data with the gravity removed using Madgwick’s quaternion. [4]

... func calculateTouchForce(touchDownTime: NSTimeInterval) -> Float {
    let data = dataCache.getRangeForTime(touchDownTime - 0.1, end: touchDownTime)
    var force: Float = 0.0
    for d in data {
        if d.getAccNoGrav().magnitude() > force {
            force = d.getAccNoGrav().magnitude()
        }
    }
    return force
}

private func calculateFlickForce(touchUpTime: NSTimeInterval, end: NSTimeInterval) -> Float {
    let data = dataCache.getRangeForTime(touchUpTime, end: end)
    var maxMag: Float = 0.0
    for d in data {
        if (d.getAccNoGrav().magnitude() > maxMag) {
            maxMag = d.getAccNoGrav().magnitude()
        }
    }
    return maxMag
}
...
These two functions (Figure 4.8.5) are responsible for calculating the touch force and flick force. The touch force calculation has evolved throughout the project through experimentation to come to this calculation. It takes a window of data from 0.1s before the touch event, it then takes the accelerometer magnitude with gravity removed with the maximum value. This is returned as an arbitrary reference to the touch force. It generally returns values between 0 and 2 which can then be used for manipulation of objects of variables by developers.

The flick force calculation works in much the same way, a timer is setup to call back to this method 0.1s after the touch has ended. Then it analyses the data from the touch end point to current time i.e. 0.1s of data looking for the maximum magnitude from the accelerometer data without gravity. Again this is returned as an arbitrary value denoting the force with which the screen was hit.

```swift
private func fireSoftPress(data: Float) {
    fireCallbacks(data, callbacks: softPressCallbacks)
}

private func fireCallbacks(data: Float!, callbacks: [(data: Float!) -> Void]) {
    for cb in callbacks {
        cb(data: data)
    }
}

func subscribe(event: EventType, callback:(data: Float!) -> Void) {
    switch event {
    case .Metrics:
        metricsCallbacks.append(callback)
        break
    case .Flicked:
        flickedCallbacks.append(callback)
        break
    case .HardPress:
        hardPressCallbacks.append(callback)
        break
    case .MediumPress:
        mediumPressCallbacks.append(callback)
        break
    case .SoftPress:
        softPressCallbacks.append(callback)
        break
    default:
        NSError(name: "InvalidEvent", reason: "Invalid event subscription.", userInfo: nil).raise()
        break
    }
}
```

The functions (Figure 4.8.6) show a sub-section of the event subscription system in the InteractionDetector which underpins the framework provided by Expressive Touch. The subscribe(event, callback) function shown is used to subscribe to a set of pre-defined
events which can be detected (metrics, flicked, hard, medium and soft presses). The event to subscribe to and a callback function reference are passed to the function, the subscribe function then pushes the relevant function onto a list of current subscriptions depending on the event being subscribed.

When a fire function such as the soft press function at the top of this block is called each callback in the list is called and data is passed to the callback where necessary.

4.9 Testing Strategy
As discussed in the Research Methodology in chapter 3 testing of these interactions will revolve around an iterative agile development process and user evaluation driven design and testing. Each interaction will be tested during the development process and evaluated highlighting any potential issues in the implementation of interactions.

Interactions which require threshold values to be set (touch force & flick force) will be determined using evaluation data for each interaction and highlighting potential threshold values which are further evaluated to ensure correctness.
5. Evaluations

5.1 Evaluation Techniques

When considering evaluations techniques initial thoughts moved towards comparing Expressive Touch to the equivalent touch interaction through both timing and accuracy of an expressive interaction and its equivalent touch gesture.

However this type of evaluation could be considered very selective when considering the breadth of touch gestures available. For instance comparing the Expressive Touch rotate interaction with the equivalent two finger twist to rotate interaction. This is a good example of where an expressive interaction may be considered easier and more intuitive to users. However there are many other possible gestures for rotation of objects such as panning across a corner of the object. Therefore any comparisons made between the equivalent touch interaction could not possibly take into account all possible touch interactions and therefore would be considered selective in its comparisons.

Therefore it was decided to evaluate the system with an approach similar to Hoggan et al. They evaluated participants ability to perform multi-touch rotation gestures. The authors evaluated the gestures by asking participants to rotate an object to a random angle within a range. They measured the accuracy of the users rotation gesture to the required angle and the time taken to rotate. [14]

5.2 Technical Evaluation

The evaluation consisted of 15 participants (age 30 ±11.2) in a lab based study, their gender and handedness were also recorded for evaluation purposes. Participants were presented with each evaluation in turn and were walked through what was expected of them. A random anonymous participant ID was generated for each evaluation which was also recorded to allow for analysis of the results based upon the information captured about participants.

To evaluate the system there was a range of interaction evaluation screens produced. The evaluations covered capturing a users ability to perform gestures by capturing metrics related to their use of the system. There were five of these evaluations specific tap force, categorised tap force, rotation, pitch and flick.
Specific Tap Force Evaluation

The following metrics were recorded:

- Tap Force (G)
- Requested Tap Force (G)

The specific tap force evaluation provided information about users ability to tap the screen with a specific level of force when feedback was given. The screen showed a yellow and red circle which was scaled to fill the yellow circle depending on the force with which the screen was hit.

The first stage of the evaluation gave participants some time to familiarise themselves with the system and the kind of force that they needed to strike the screen with in order to manipulate the red circle. The purpose of this step was to allow participants to assess and if necessary to adjust their hand movements to ensure proper capture of the touch interaction from the wrist-worn sensor. Once participants have familiarised themselves with the force levels they are presented with a blue circle denoting the force they need to hit the screen with. The participants were given 3 attempts at each force, at each attempt they were provided with feedback on how hard they struck the screen. After the 3 attempts the participants were asked to move onto the next random force level. There were 10 random force levels in each run.

The purpose of this evaluation was firstly to determine participants ability to perform taps of specific forces as opposed to defined categories of forces. The results from this evaluation would hopefully provide insight into the application areas for Expressive Touch force interaction. Another aspect which would hopefully be evident was participants ability to respond to feedback presented to them during the evaluation and their ability to adjust their taps.
The following metrics were recorded:

- Tap Force (G)
- Category Requested

The categorised tap force evaluation provided information about users’ ability to distinguish between a requested tap force. Participants were presented with colour coded action requests, ‘Soft’, ‘Medium’ and ‘Hard’.

Initially participants were asked to tap the screen ‘Soft’, then ‘Medium’ then ‘Hard’. After this they were asked to perform a further 9 taps of each force category in a random order. The purpose of the initial structure is to allow participants to develop what they deem to be the three levels of touch force. Potentially providing us with information about their ability to repeat their taps with the same level of force.

This evaluation aimed to provide data to show not only if a user could distinguish between the three categories of touch force but if they could accurately repeat a touch force. Information about participants’ ability to distinguish between the three touch forces proposed could also be used to form a calculation of thresholds between the three proposed tap forces by looking at the spread of data from the evaluations for each force.
Rotation Evaluation

The following metrics were recorded:

- Rotation Range Max & Min (°)
- Requested Angle (°)
- Actual Angle (°)
- Time to Complete

The rotation evaluation was developed to assess participants' ability to perform the rotation interaction, measuring time and accuracy of participants when asked to rotate an object to a specified angle. The first stage of the evaluation involved asking participants to demonstrate their range of rotational range by pressing down with their index finger and rotating clockwise as far as they were comfortable then back as far as they were comfortable anti-clockwise. The purpose of this was to provide an insight into the average freedom of movement that users had when performing the rotation interaction. This information could be used to inform development of applications involving the rotation interaction to ensure that users are not asked to perform interaction beyond average ranges.

Once the rotational range of participants had been obtained, the evaluation generated two random angles within their range of movement and set one image to this angle and another placeholder for the image to the second angle. Participants were then asked to rotate the image on top to the angle illustrated using the rotation interaction in one movement (without lifting off the screen). The time from the touch down until the touch up was recorded along with the final angle. Participants were asked to perform this 10 times with random angles at each stage.
The reason for only allowing one motion to rotate the image was to determine how participants reacted to the task. Angles to rotate at the extremes of their range would require what we call ‘priming’ of the hand to achieve the angles which would hopefully provide interesting insight into participants willingness to think about each interaction before starting.

Pitch Evaluation

The following metrics were recorded:

- Pitch Range Max & Min (°)
- Requested Angle (°)
- Actual Angle (°)
- Time to Complete

The pitch evaluation provided information about participants ability to perform the pitch interaction. Participants were asked to pitch a slider to a requested value illustrated by a fixed slider. Various aspects of the interaction including accuracy, range and time to complete were recorded in the evaluation. The first stage in the evaluation asked participants to demonstrate their range of pitching by holding their finger on the screen, pitching upwards as far as they could then pitching downwards as far as they could. The purpose of this stage of the evaluation was to provide information about the freedom of movement of participants when performing the pitch evaluation. This information could be used to inform the design process when implementing the pitch interaction into applications.

After the maximum and minimum ranges of pitch had been determined the evaluation then selected a random pitch angle and set a second bar to illustrate the value to pitch to. Participants were then asked to pitch to the requested angle in one motion without
lifting their finger off the screen, once they had reached the requested angle they lifted off the screen. This was performed 10 times.

The purpose of this evaluation was to determine the accuracy and speed with which participants could pitch to a specific value. This information would provide an insight into participants' competency when performing the pitch interaction. Application of this evaluation data would aid development of applications using the pitch interaction. Ensuring that users were able to perform the required interaction both in terms of freedom of movement to perform the interaction, and range of movement required to complete any such interaction.

Flick Evaluation

The following metrics were recorded:

- Flick Force (G)
- Requested Flick/No Flick

The final evaluation was developed to investigate participants' ability to distinguish between a flick off the screen and lifting off without flicking. Participants were asked to press onto the screen after which they were asked to either flick off the screen or lift off without flicking. The flick evaluation did not request participants pull away from the screen in any specified motion rather let the participant decide how they would like to pull away from the screen. The reason for this was to provide a range of values which could potentially be used to determine a threshold for determining whether a flick interaction had occurred or not after the touch up event. Participants were presented with 10 colour coded requests to either flick off or lift gently off the screen without flicking. At each stage in the evaluation participants the force with which participants flicked off the screen was recorded.
5.3 Results

Specific Tap Force Evaluation

The graph (Figure 5.3.1) details the results for the specific tap force evaluation, each participant attempted to tap the screen 10 times with randomly selected force levels illustrated by a blue circle. They were given 3 attempts at each force. Attempts are categorised by colour to show that users in general improved in their attempts as they progressed.

Another observation to be made from the results is that participants seemed to be better at tapping with softer forces than harder ones, the results at lower force levels were closer to the requested force line in yellow. However the results are more spread as the requested force increases.

Overall the results of this evaluation show that participants struggled to hit the screen with a specific force. While there is a correlation between the tap force and requested force, (0.653) it is a weak correlation which only improves slightly as the attempt number increases.
Categorised Tap Force

The graph (Figure 5.3.2) shows the results from the categorised touch force evaluation. The data shows that in general users could distinguish between Soft and Hard touches but struggled with a Medium touch.

During the evaluations it was observed that participants often hesitated with Medium press requests but were quick to react to both Soft and Hard presses. The data from the experiments showed that there was very little consistency even amongst each participant in the force with which they struck the screen with a Medium press.

Taking into account the results from this experiment, it would seem that the Medium press in Expressive Touch should be removed. Which while restricting the parameters from which developers can target interactions will reduce user confusion and hesitation when attempting to perform expressive interactions.
Rotate Evaluation

The graph (Figure 5.3.3) shows the time users took to rotate the given object against the angle they were asked to rotate by. This data gives us a rather interesting insight, the angle participants were asked to rotate did not affect the time to rotate. This means that if a user was for instance asked to rotate an object by 180° and 10°, both these operations take a similar amount of time.

Another interesting observation that can be made from the underlying data from participants was the range of movement participants had in one complete rotation which was captured at the beginning of the evaluation. The average rotation range for participants was 227° (-86° to +141°) which allowed participants a large degree of freedom when performing the rotation interaction. The application of their range of rotation was also influenced by how participants prepared themselves for the interaction. Some participants ‘primed’ their hand before the interaction if they noticed the angle to rotate was at the extremity of their range in either direction by starting with their hand rotated in the opposite direction to that of travel, thereby giving themselves a much greater range of movement than simply starting in a standard palm down and neutral position.

During the evaluation participants were asked to wear the sensor on their right wrist, regardless of their handedness. As such the rotational data captured considered clockwise movement of the wrist as positive in value and anti-clockwise movement as negative in value. Therefore we can see from the data that participants had a much lower range of counter-clockwise rotation this is due to bio-mechanical constraints restricting counter-clockwise movement on the right wrist and clockwise movement on the left wrist.
The graph (Figure 5.3.4) provides an analysis of participants' ability to complete the rotation task being asked of them. More specifically, the graph shows how accurate users were in their rotation. It shows that the vast majority of participants were accurate to within 5 degrees when rotating an object, showing not only the accuracy of the roll detection itself but also demonstrating the ability of participants with little experience to effectively use Expressive Touch. However, something that needs to be considered when analysing these results is that participants were not asked to be as accurate as possible. During the experiments, it was observed that some users were very precise with their placement of the object at the requested whereas others were less careful in their placement.
Pitch Evaluation

As you can see from the graph (Figure 5.3.5), the pitching evaluation showed the same pattern for the most part with the angle to pitch having little effect on the time to complete. However the data is more spread the closer to zero the angle to pitch is, this was something that was observed during the evaluations. Participants struggled in general to be accurate when dealing with fine angles which were requested.

Evaluation of the pitching ranges reveals that users had quite a restricted range of movement when pitching with an average range of only 52° (-6° to +46°). The reason for this restricted range of movement would appear to be due to the non-uniform manner in which participants pitched their wrist with their hand. For instance it was observed in the evaluations that many participants pitched upwards by moving their wrist and hand but then pitched downwards by moving only their hand, thus their downwards motion was not captured by the sensor. Therefore applications involving the pitch interaction need to carefully consider this on average restricted range of movement demonstrated by participants.
As you can see from the graph (Figure 5.3.6), while most participants achieved the requested angle within roughly 5 degrees, however many were outside of this range. This graph shows that while most participants were accurate many struggled to achieved the required angles due to the issues outline previously, relating to the non-uniform manner in which participants performed the pitch interaction. The results of this evaluation must also be taken into account in applications of Expressive Touch as participants were able to pitch the view but setting to a specific value is not an appropriate application of this interaction. An example of a good application of this interaction would be the maps demo developed in this project which uses the pitch to manipulate the view but specific setting of the pitch is not important.
The graph (Figure 5.3.7) shows the results from the flick force evaluation, with the 20th and 80th percentile highlighted. As you can see from the results participants were able to distinguish between what they considered a flick and no flick. There was very little overlap in the evaluation data between the flick and no flick operations which shows a high probability that the system can be used to distinguish between these two interactions. Looking at the results from this evaluation a threshold of around 0.5 would appear to be suitable for flick detection.

During the evaluations several participants pointed out that they felt this interaction was already available to them through a swipe across the screen and they felt that this interaction was unnecessary. However the flick interaction allows for a much greater freedom of movement that did not require contact with the screen to detect. This interaction can also be more easily performed on smaller screens due to the lack of contact area available to perform a swipe gesture. After discussing this with participants many agree with this justification and after having performed the interaction in the evaluation could see its potential.
7. Discussion & Conclusions

7.1 Aims Evaluation

To explore the integration of wrist worn movement sensing and tablet devices.
A detailed exploration of the possibilities for the integration of inertial measurement data from a wrist worn-sensor into tablet device interaction was performed. Initial investigations looked at what can be inferred about the posture and movements of the hand from the sensor data, then investigating possible application scenarios when combining this data with touch events from smartphones and tablets. Investigations into the technical requirements of integrating wrist worn movement sensors into the touch interaction with smartphones and tablets was also performed. Detailing the current sensors available and their possible applications in relation to Expressive Touch.

To investigate possibilities for adding expressiveness to touch interactions using data from the sensor.
A set of Expressive Touch interactions were developed which considered the current touch interactions and looked to add expressiveness to each basic touch interaction. The developed interactions carefully considered what information was missing from current tablet and smartphone screens and looked to provide this information from the sensor data. Using real-world examples of where expressiveness is important to the touch interaction, such as painting, the interactions proposed provide a new set of metrics which allow for a more expressive interaction to be implemented. An important consideration during the development of Expressive Touch interactions was concerning the accuracy of the interaction, both in terms of detection of the interaction and of the metrics returned from the interaction.

To develop a system for the application of this research to other platforms or applications.
Throughout the development process Expressive Touch was produced in such a way that it could be easily be applied to other applications and data sources. The sensor data is encapsulated in a custom wrapper allowing for any data source to be used simply by updating the data cache with the required sensor data.

The event subscription system outlined above forms the other part of the framework system. Developers can subscribe to a defined set of interactions, when Expressive Touch detects an interaction the subscribed callbacks are called. This provides a very clean and precise way to use Expressive Touch.

7.2 Evaluation Issues

During the evaluations informal interviews were carried out with participants. These conversations covered how the participants felt about carrying out the various interactions being evaluated and provided some valuable feedback on Expressive...
Touch. When initially trialling the evaluations with some users, it was immediately apparent that users struggled with the force evaluations. This was mainly due to many users tapping the phone without moving their wrist i.e. the motion came from the hand movement only, therefore no or little force was recorded. In the categorised force evaluation there is no feedback to the user of the force with which they hit the screen. Therefore participants did not realise they were not recorded their intended force. However if I changed the order of the study asking participants to do the specific tap force evaluation first which did provide immediate feedback participants quickly adjusted their interaction to cause the desired effect.

Other issues participants faced during the study involved the pitch evaluation. The first step of this evaluation asked users to demonstrate their range of pitch, this caused a few issues. First and foremost being that participants often lost their touch point when pitching upwards when their nail hit the screen, this was an unintended and frustrating reality to the pitch interaction. Another issue participants faced was the seeming lack of response to their pitch motion when interacting using Expressive Touch. Throughout the pitch evaluation users are provided with feedback to show the interaction being recorded by the device. However users often found that the system did not record their downwards pitch motion, from observing each participant in the study it was discovered that this seemed to be due to users pitching their wrist in a non-uniform manner. Users would pitch upwards with their wrist but when pitching downwards they would pivot their hand only, resulting in on-screen feedback which did not match their apparent movement.

7.3 Information Overload

The interactions proposed here provide a wealth of information for developers to use in their applications allowing users to add expressiveness to their touch interaction. This information can be freely used across their applications for manipulation of elements which developers feel is relevant. However a careful consideration that developers must take into account is an information overload for users, for instance users accidentally performing an interaction.

A demonstration of this situation was evident in the Maps demo, users were able to pitch the 3D map view by pitching their wrist as well as change the map heading by rolling their wrist. Something that was observed was several users inadvertently pitching the view when rotating to the extremes of their rotation ranges. The reason for this was due to physiological constraints from rotating your wrist, close to the point of maximum rotation users tend to pitch their wrist to achieve a small amount of extra rotation. This causes inadvertent pitching to be detected.

Accidental performance of each of the proposed interactions is potentially an issue for implementation of actions based upon said interactions. A destructive action such as deleting an email being mapped to a hard press for instance could lead to accidental deleting of items if users inadvertently tapped the screen hard. As such careful
consideration of the application of expressive interaction must be taken to ensure that where possible the effects of accidental interactions are reduced.

A proposed solution to some of the accidental interaction performance issues outlined above would be to provide a threshold for the performance of continuous interactions. This would ensure that for instance a rotation interaction would only occur if the user rotated past a defined threshold. This would reduce accidental rotation of objects at the expense of reducing the expressiveness and responsiveness provided by the rotation interactions. This is key to the development process striking the correct balance between usability and expressiveness. Reducing expressiveness in some aspects may increase usability of an interaction and vice versa.

7.4 Expressive Touch as a Product

This project has shown through user evaluations that information from a wrist-worn sensor can be used to augment the touch interaction with tablets and smartphones providing an expressive interaction to users. The system used data streamed from a wrist-mounted IMU which are present in many modern smartwatches, therefore it is reasonable to assume that the technology is available for the system to be deployed as a product.

However there are several practical and technical limitations that will be outlined which must be overcome before the system can be considered a viable consumer system. While the smartwatch market can be considered a rapidly growing and evolving market current technology limitations involving battery life and API access restrictions cause issues. The wrist-worn IMU from the OpenMovement project has very limited battery-life lasting only a matter of hours while streaming data. Production smartwatches also have a limited battery life usually requiring recharging every night to remain functional. Constant streaming of IMU data from these watches will impact their battery life.

However this could be kept to a minimum through techniques such as only streaming data when an Expressive Touch application is in view on the device. It would appear that due to this exact issue Apple’s new Apple Watch provides a very limited API with no direct access to the internal IMU. While it is safe to assume this information will become available on future products it is unlikely that a constant stream of data will be considered viable for battery life. [12]

A practical limitation of Expressive Touch is that many users prefer to have their watch on their non-dominant hand but in general interact with their devices using their dominant hand. Therefore the IMU data from their smartwatch would provide no insights as to the movement of the hand they usually use to interact with their devices. However a reasonable assumption could be that the user would adjust their operation for Expressive Touch applications to take advantage of the system.

Other physiological limitations involving practical applications of Expressive Touch revolve around a user’s movement of their wrist with respect to their hand. While some users tap the screen by moving their wrist and hand others will move only their hand or
finger thus restricting the IMU data being recorded. The same applies to flicking off the screen. Rolling of the wrist is a motion which in general is not susceptible to this issue as users are physically unable to roll their finger without rolling their wrist. However the same cannot be said of the pitch gesture with many users struggling to make their apparent pitch movement match what was shown on the device.

7.5 Future Developments

The future development possibilities of Expressive Touch are broad. The system proposed here provides a framework for the development of more complex and expressive interactions. However I will highlight some of the most interesting examples for development in the future.

One issue with the current Expressive Touch implementation is the lack of multi-touch support, the reason for this was simply to reduce the immediate complications with involving multi-touch interactions and the fact that many of the expressive interactions proposed can replicate multi-touch interactions. However the system could be expanded to support multi-touch, this expansion could be achieved through modification of the current InteractionDetector to work with a ‘touch map’ rather than single touch up and down events. The updated system would keep track of a set of touches. This would enable a whole new set of possible interaction modalities by augmenting the touch interactions further through the use of multi-touch gestures.

Another benefit of moving to a system that supports multi-touch would be to allow multiple users with multiple sensors or even the same user with a sensor on each hand to interact using Expressive Touch. However an apparent issue with this would be assigning which touch events to which hand an hence which sensor, recent work by the Digital Interaction Group at Newcastle University TOUCHI would suggest that this is possible. They suggest that using the IMU data to match the on-screen touch gestures with the IMU data provided a high level of accuracy provided their was ample movement involved when the user was interacting. [11]

During the evaluation many participants performing the pitch and roll evaluations asked about the possibility of also being able to pitch and roll the device thus achieving an ever greater range of movement. Possible future expansion of Expressive Touch to involve the device gyroscope for rotational changes would be an interesting avenue for exploration. This type of integration could also be applied to the force detection systems taking the devices accelerometer data to get acceleration changes relative to the device rather than just the IMU. An example of the advantage of this would be if you were using the system in a car or train, movements of the transport would affect the accelerometer readings. Removing any data that also shows on the devices IMU would show the acceleration of the wrist relative to the device.

Expanding this further another possible research path would be to investigate the possibilities for calculating the absolute pitch and roll angles of the hand relative to an initial position rather than the current setup of measuring changes relative to the touch
point. Using the devices IMU data and the wrist-worn sensor IMU data together to work out an assumed pitch and roll of the wrist would open up another avenue of possibilities for future applications allowing for response to touches based not only on the force but the pitch and roll of the hand on contact. This could be compared to some extent to some of the gestures proposed by Chen et al with Duet. However Expressive Touch would provide much more detail about the hand posture. [2]

Also from purely a development aspect the system could be made more ‘friendly’ for developers to interact with by following more closely the repository design pattern for data storage and access. Currently developers wishing to use Expressive Touch would need to update certain aspects of the system to work with other sensors. Following of the repository design pattern would have enabled through the use of interfaces (protocols in Swift) and dependency injection such as Unity to produce a more robust and simple system for developers to interface with. Providing interfaces for the sensor data and data processing would enable developers to utilise the underlying functionality of Expressive Touch simply by adding in a library and implementing the required interfaces or protocols.

7.6 Conclusions
The system proposed in this project demonstrates how inertial data from a sensor on the wrist can be used to augment the touch interaction on a smartphone or tablet touch screen. The evaluations performed show that participants were able to perform the various interactions proposed in this project with a high degree of accuracy in most cases.

Also discussed was the potential for this system to be used in the real world, discussing both the technical and practical issues which are currently faced by the introduction of this system. Careful consideration of the implementation of the expressive interactions proposed here needs to be taken to ensure that the system has the desired response to users. Particular consideration of users range of movement needs to be taken into account when designing interfaces using continuous interactions. Providing accurate and simple feedback to users and ensuring that scaling of metrics returned from the system provide the correct feedback is also of great importance. Force is a particularly good example of this, participants in the evaluation mentioned that they struggled with the tap force feedback as they felt it did not represent what they felt they had tapped the screen with. Therefore careful adjustment of the scaling is important to ensure users feel the connection between their actions and the feedback.

The novel interactions proposed in this project outline a broad and exciting prospect for the possibilities of future interaction techniques with smartphones and tablets through utilisation of smartwatch IMU data. Further research into this system outlined in the sections above could result in a comprehensive and interactive system that has real-world applications without users requiring specialist hardware to use the system.
References


10. Digital Interaction Group, Newcastle University. (Submitted for publication). SwipeID: Using IMUs to Identify Supervisors on Touch Devices.
11. Digital Interaction Group, Newcastle University. (Submitted for publication). TOUCHI: Online Touch User Identification through Wrist-worn Inertial Measurement Units.


Appendices

A - Evaluation Participants
Evaluation data is made available on GitHub for analysis. [32]


B - Specific Force Category Evaluation
Evaluation data is made available on GitHub for analysis. [32]


C - Categorised Tap Force Evaluation
Evaluation data is made available on GitHub for analysis. [32]


D - Roll Evaluation
Evaluation data is made available on GitHub for analysis. [32]


E - Pitch Evaluation
Evaluation data is made available on GitHub for analysis. [32]


F - Flick Evaluation
Evaluation data is made available on GitHub for analysis. [32]


G - Code
All project code is hosted on GitHub with documentation and setup instructions. [32]