Machine Learning Techniques For Implicit Interaction Using Mobile Sensors

Mohammad Faizuddin Md Noor

Submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy

School of Computing Science

College of Science and Engineering
University of Glasgow

July 2016

Abstract

Interactions in mobile devices normally happen in an explicit manner, which means that they are initiated by the users. Yet, users are typically unaware that they also interact *implicitly* with their devices. For instance, our hand pose changes naturally when we type text messages. Whilst the touchscreen captures finger touches, hand movements during this interaction however are unused. If this implicit hand movement is observed, it can be used as additional information to support or to enhance the users text entry experience. This thesis investigates how implicit sensing can be used to improve existing, standard interaction technique qualities. In particular, this thesis looks into enhancing front-of-device interaction through back-of-device and hand movement implicit sensing.

We propose the investigation through machine learning techniques. We look into problems on how sensor data via implicit sensing can be used to predict a certain aspect of an interaction. For instance, one of the questions that this thesis attempts to answer is whether hand movement during a touch targeting task correlates with the touch position. This is a complex relationship to understand but can be best explained through machine learning. Using machine learning as a tool, such correlation can be measured, quantified, understood and used to make predictions on future touch position. Furthermore, this thesis also evaluates the predictive power of the sensor data.

We show this through a number of studies. In Chapter 5 we show that probabilistic modelling of sensor inputs and recorded touch locations can be used to predict the general area of future touches on touchscreen. In Chapter 7, using SVM classifiers, we show that data from implicit sensing from general mobile interactions is user-specific. This can be used to identify users implicitly. In Chapter 6, we also show that touch interaction errors can be detected from sensor data. In our experiment, we show that there are sufficient distinguishable patterns between normal interaction signals and signals that are strongly correlated with interaction error. In all studies, we show that performance gain can be achieved by combining sensor inputs.

Acknowledgements

My deep gratitude goes first to my advisors, Simon Rogers and John Williamson, for guidance, understanding, patience, countless useful insights and most importantly for tirelessly motivating me throughout my PhD. Thanks also to Roderick Murray-Smith for his input and numerous valuable insights throughout the process.

My appreciation goes out to everyone I have shared an office space with over the past four years for moral support, stimulating conversations and advice. In no particular order, thanks to Edwin, Rebecca, Anna, George, Tommy, Dimitar, Joe, Collin, and Shimin.

Thanks to all my collaborators and everyone who participated in the experiments.

I thank my parents, Mak and Abah for their faith in me and allowing me to be as ambitious as I wanted. It was under their watchful eye that I gained so much drive and an ability to tackle challenges head on. Also, I thank my wife's parents, Anwar and Rasheeda. They, like me and my wife, are a couple who endured and survived the experience of graduate school and provided me with unending encouragement and support.

Finally, I would like to thank a very special person in my life, the only woman who I am in love with and will love forever - my wife, Farah. Her infinite support, encouragement, quiet patience and unwavering love were undeniably the bedrock upon which the past ten years of my life have been built. Her tolerance of my occasional boorish moods is a testament in itself of her unyielding devotion and love. Thanks for spending the most valuable ages of your life for my study and our family.

To the many people whom I have undoubtedly missed from this list, I apologise and once again thank you from the bottom of my heart.

For Mak, my wonderful wife Farah and my two children Zia and Nadine.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Interactions in mobile devices normally happen in an explicit manner, which means that they are initiated by the users. We touch, we speak, we hold and we gaze at our devices in order to interact. This requires a specific set of sensors to capture these interactions. Yet, users are typically unaware that they also interact implicitly with their devices. For instance, our hand pose changes naturally when we type in our text messages. Whilst capacitive sensors of the touchscreen capture touch position, hand movement during this interaction is unused. If this implicit movement is observed, it can be used as additional information to support or to enhance the users' text entry experience. This is the form of implicit interaction that this thesis is based upon.

Modern commercial devices are equipped with a variety of sensors. Although these sensors are specific to certain types of explicit interaction (e.g. microphone to sense voice, accelerometer to sense device orientation), they also can be potentially used for implicit interaction. The idea of leveraging existing sensors for implicit interaction is not new, in fact most of implicit interaction techniques using mobile devices are based on the concept of implicit sensing. However, the focus of this thesis is quite different; instead of using mobile sensors to derive new implicit interaction techniques, this thesis investigates how implicit sensing can used to improve existing, standard interaction technique qualities. In particular, this thesis will look into enhancing front-of-device interaction through back-of-device and hand movement implicit sensing.

This thesis approaches the investigation through machine learning techniques. It primarily looks into problems on how sensor data via implicit sensing can be used to predict a certain aspect of an interaction. For instance, one of the questions that this thesis attempts to answer is whether hand movement during a touch targeting task correlates with the touch position. This is a complex relationship to understand but can be best explained through machine learning. Using machine learning as a tool, such correlation can be measured, understood and used to make predictions on future touch position. Furthermore, this thesis also evaluates

the predictive power of the sensor data.

This thesis will present a number of implicit interaction techniques to enhance the way the user interacts with mobile devices. In particular, it will demonstrate how implicit sensing enables prediction of touch contact position ahead of time. Besides that, implicit sensing also can be embedded into standard touch interaction to identify users transparently. It also will show that touch interaction errors can be inferred from observing patterns in mobile sensors data. Moreover, this thesis also will show that the combination of multiple sensor inputs could yield performance gains in predictions. These techniques can be beneficial for mobile devices since a wide array of sensors are now a part of standard commercial devices.

We run a number studies to support our hypothesis. We show tha sensor inputs and recorded touch locations can be modelled to predict the general area of future touches on touchscreen. Furthermore, we show that data from implicit sensing from general mobile interactions is user-specific. This can be used to identify users implicitly. We also show that touch interaction errors can be observed from sensor data. In our experiment, we show that there are sufficient distinguishable patterns between normal interaction signals and signals that are strongly correlated with interaction error. In all experiments, we show that performance gain can be achieved by combining back-of-device and accelerometer.

1.1 Thesis Statement

The goal of this thesis is to examine the use of mobile sensors for implicit interaction to enhance standard mobile interaction. Due to the complex nature of interaction between human and mobile device, we use machine learning techniques to learn and model this relationship. Our assertion is that we can use machine learning to infer implicit behaviours which could improve the explicit interaction itself.

1.2 List of Contributing Papers

The work described in this thesis has led to two conference papers[1]. These papers are as follows:

[1] M.F. Noor, S.Rogers, J.Williamson. ”Detecting Swipe Errors on Touchscreens using Grip Modulation”, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* New York, New York, USA: ACM Press, May 2016.

[1]The work in the these papers was primarily carried out by the author. The exception is the designing and developing the hardware prototypes, which was implemented by colleagues, Andrew Ramsay and Stephen Hughes.

[2] M. F. Mohd Noor, A. Ramsay, S. Hughes, S. Rogers, J. Williamson, and R. Murray-Smith, "28 Frames Later: Predicting Screen Touches from Back-of-Device Grip Changes", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, New York, USA: ACM Press, April 2014, pp. 2005 – 2008

Chapter 5 of this thesis is based on the first paper whereby Chapter 6 is based on the second paper. Analyses given in this thesis are based on the expanded version of the work in the papers.

1.3 Overview of Thesis and Research Contributions

The contributions of this thesis are:

- A hardware design and acquisation methods to capture salient information from back–of–device (BoD) grip changes during touch interaction.
- A touch area prediction model based on Gaussian Processes which describes the non-linear relationship between BoD sensor inputs and front-of-device touch locations.
- A detailed study on identifiying user implicitly from the way the device is grasped.
- A detailed examination on the use of BoD grip and hand motion changes as manifestation of cognitive errors in swipe-based tasks.

The remainder of the thesis is organised as follows:

Background

Chapter 2 discusses the research literature which this thesis is built on. The chapter begins with presenting important distinctions between explicit and implicit interactions and relates it with other terms used closely in this area. Next, the main motivation behind implicit interaction is given. This includes explanation of the problems with standard explicit existing interaction is given where can be minimised using implicit interaction. Also given is several implicit interaction approaches used in the literature to overcome the constraints imposed by standard interaction.

Chapter 3 gives a detailed look on the proposed hardware design and sensor technologies for mobile implicit interaction, which is used as a data collection tool in this thesis.

Chapter 4 introduces the machine learning (ML) techniques used in this thesis which includes data pre-processing and exploratory analysis techniques. Also given in this Chapter is discussion on the approaches taken to evaluate prediction performance.

Contributions

Chapter 5 presents the use Gaussian Process regression to model general touch area from BoD and accelerometer sensor inputs. The results from this study show that touch locations are correlated with sensor inputs up to several hundred milliseconds before touch.

Chapter 6 examines the use of grip and hand motion changes as an implicit input to recognise swipe errors which can be detected accurately using Support Vector Machine (SVM) and Random Forest (RF) classifiers.

Chapter 7 shows the application of implicit interaction for identification purposes. In particular, it explores identification method using implicit sensing of hand grip and hand motion from two standard mobile interactions to identify users automatically. The results show that there is sufficient user-specific information in each input modality.

Conclusions

Chapter 8 summarises the work and contributions, provides potential avenues for future research and subsequently concludes this thesis.

Chapter 2

Background

Summary
This chapter provides an overview of the relevant research literature related to this work. This includes a review on explicit and implicit interactions and the relationship between these two. Also given is the main motivation behind implicit interaction, focusing on the standard interaction enhancements that can be achieved using implicit interaction. Several use of implicit interaction to overcome constraints imposed by standard interaction are given in the remainder of this chapter.

2.1 Introduction

This thesis presents approaches to predict interaction in mobile device using machine learning techniques. In particular, these are achieved through the use of implicit interaction at the back of mobile device with the purpose to improve the quality of standard explicit interaction. The implicit approach to interaction is not a new concept and has been well explored. This chapter begins with a view on the problems with standard interaction and efforts to improve it by focusing on implicit methods.

A thorough review of previous work on implicit interaction is given Section 2.2. This includes approaches, mobile sensing technology and development of hardware for implicit interaction. Also given is the machine learning techniques adopted by these approaches. Section 2.3 gives an overview of applications using implicit method for interaction. This looks into three application domains where implicit interaction excels in enhancing standard mobile interaction.

2.2 Explicit and Implicit Interactions

The terms explicit [96] and implicit interaction [61, 60] which have been used widely in the literature can be related to Buxton's foreground/background interaction model. It defines human-computer interactions into two general categories: (1) foreground interaction which refers to the activities that are *fore* to human consciousness (intentional activities) and (2) background interaction which refers to the tasks that occurs *behind* foreground activities [14]. Therefore based on this model, foreground interaction can be considered as explicit interaction since it involves deliberate action perform by the user to achieve his/her goal (i.e. typing, clicking). On the other hand, implicit interaction can be regarded as background interaction, since the interaction happens in parallel during explicit interaction. For instance, changes in grip strength (background/implicit) during typing (foreground/explicit).

In the context of mobile devices, Hinckley et al. view foreground interaction as deliberate user activity or while attending the device whereas background interaction as unconscious activity that occurs naturally. By using sensors, background interaction can be sensed and used to complement foreground interaction [52].

For instance, almost all user interactions with mobile devices depend on the screen. This follows the conventional desktop computer interaction model where a user interacts with the main display using an input device such as a mouse, keyboard or touchscreen, requiring direct attention from the user. However, passive or indirect interactions such as eye and hand movements occur simultaneously when interacting with the display. Therefore, based on the foreground/background interaction model established by Hinckley et al. [52], these indirect interactions can be measured using front-facing camera (eye movement) and accelerometer (hand movement) sensors which in turn can be used as an additional input to support touchscreen interaction. In other words, implicit interaction foregrounds the human while putting the technology in the background.

The distinction between explicit and implicit interactions can be determined by the level of *controlibility* that the user possesses during an interaction. For instance, interaction using keyboard, mouse or multi-touch touchscreen is considered as explicit interaction because the user has the control over what needs to be done (i.e. typing, pressing, tapping). However when the device senses user's physiology information such as heart and breathing rates as input command, this is considered as implicit interaction because the user has no control over these attributes. Figure 2.1 shows the continuum of interaction methods, from explicit to implicit.

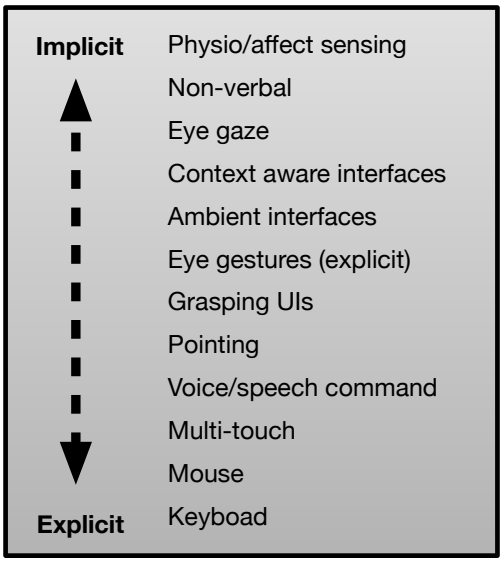


Figure 2.1: Continuum from explicit to implicit interaction.

2.2.1 Implicit Interaction in Mobile Devices

Unlike desktop computers, users use mobile devices in many different ways and environments. Users bring their mobile devices when they drive, when they go out running and when they go to bed. These situations create different interaction contexts that requires different actions from the user. However, users do not have the luxury to interact with mobile devices explicitly for all the time. Implicit interaction attempts to address this by learning the context and adapting the interaction to suit the current task or situation. For example it may be useful to activate a mobile device's voice operated mode when driving and turning off notification system at bedtime without having the user to do so. Therefore given an input (i.e. moving speed), the device has the ability to understand what needs to be done (i.e. activate voice operated mode). In particular, this assumes that the device has the *knowledge* of a certain *context* and use this as an additional input to achieve the goal. In other words, implicit interaction can be defined as an interaction that is based on two concepts: 1) perception (context) and 2) interpretation (knowledge) [116].

Sensors are fundamental to implicit interaction. Almost every implicit technique in the literature is based around the concept of implicit sensing – that is using sensors as an implicit input for interaction. Today, mobile devices are equipped with various of sensors and many research in this area have leveraged these sensors to create new implicit interaction techniques for mobile devices. For instance, phone's accelerometer has been the popular choice of input for many implicit interaction techniques based on movement and motion for instance in personal health [99] and activity tracking [77]. Similarly, eye-gaze interaction is achieved by leveraging phone's front-facing camera [37, 62] whereas microphones have been used to implicitly recognise speaker's emotion [97]. Apart from leveraging built-in sen-

sors, there are also techniques that adopt additional sensors to create new implicit interaction techniques. Chang et al. [19], Cheng et al. [23] and Cheng et al. [24] for instance augmented back and sides of a standard mobile device with capacitive sensors to enable hand grip-based implicit interactions.

2.3 Applications of Implicit Interaction in Mobile Devices

In this section we review related techniques for implicit interaction in mobile devices that can be found in the literature. In relation to this thesis, we focus the work on the implicit interaction techniques in three application domains: 1) authentication 2) touch interaction and 3) error detection. In addition, we also discuss about the benefits these techniques can offer as well as their limitations.

2.3.1 Authentication

In general, implicit authentication in mobile devices attempts to authenticate users without requiring the users to provide their identity proof such as PIN and password explicitly. Implicit authentication is designed mainly to minimise memory challenges and cumbersome time consuming verification procedures. There are a significant number of implicit authentication mechanisms that have been proposed previously for smartphone that are centred around the idea that all users behave differently. Based on this notion, much of the work in this area attempts to design an authentication mechanism that functions through user behavioural indicators. The following sections give an overview on techniques proposed for implicit method for authentication for mobile devices.

Gait

Identifying people from the way they walk was first introduced by Cutting and Kozlowski in the the late 1970s [33]. Since then, researchers have attempted various techniques to identify people from the way they walk both visually and physically. Ailisto et al. [1] and Liu et al. [81] for instance used an accelerometer attached to the users waist to identify people from their gait pattern. This technique is considered practical for smartphones because most of the modern smartphones have built-in accelerometers and it can be implemented without additional hardware. However there are many factors that can alter gait pattern such as footwear, ground surface and carrying load and therefore can significantly affect the recognition results [41]. Besides that, gait recognition system requires user to walk in order for the

system to authenticate. This can be a problem in a situation where the user has been seated for a long time.

Keystroke

Another approach that has been proposed for implicit authentication is keystroke dynamics. In general, this approach measures the keystroke latency or time span between two keystrokes. Over years, researchers have enhanced this technique by adding several other factors such as key hold time and error rates when performing the keystrokes [28, 90]. Zahid et al. [150] in their work have considered the use of on-screen keyboard perform keystroke dynamics analysis. Their system has to be trained with a minimum of 250 keystrokes in order to achieve a low Equal Error Rate[1] (ERR) of approximately 2%. Alternatively, Giuffrida et al. [42] used hand movement whereas Saevanee and Bhattarakosol [108] adopted finger pressure from keystrokes to authenticate users. Both achieved EER of 0.08% and 1% respectively. In another study, Angulo and Wästlund [2] proposed a keystroke authentication system by enhancing the existing graphical pattern lock found in most Android smartphones with biometrics. They considered two features; time taken to place the finger in the dot and the speed to move the finger from one dot to another. Their approach managed to achieve ERR of 10.39%. Albeit of the positive result, this technique is still considered partly implicit since it requires the user to draw the pattern on the screen explicitly in order to unlock and use the smartphone.

Location

The inclusion of a GPS sensor as standard in modern smartphones has motivated researchers to explore and design an authentication mechanism that could learn from user whereabouts. This approach generally assumes that users tend to visit same places every day, for instance going to an office or a bus station. However, this kind of authentication may not be suitable to be used on its own as users may travel or go to unfamiliar places. Instead Shi et al. [124] proposed an authentication method based on Gaussian Mixture Model (GMM) trained using multiple user's behavioural features acquired from smartphone sensors, user biometric information and smartphone applications usage. They demonstrated that by combining together users location information with other features allows them to better distinguish a legitimate user from an adversary.

[1] EER is a value where False Accept Rate (FAR) and False Reject Rate (FRR) are equal.

File System and Network Access

An authentication method based on the file system access and network usage has been proposed by Yazji et al. [148]. They investigated if a user can be identified from their network activities and file access behaviour, using K-means clustering to construct a model of the file system and network behaviour. They considered a client-server architecture in their approach where detection of anomalous events is performed on the server side. While this approach is able to distinguish between normal use and attack with an accuracy of approximately 90%, it however requires between 80 MB to 25 GB of training data observed over two weeks. Apart from that, they also reported that the system has a detection latency of five minutes which may not be reasonable for security concerned applications. The authors propose to use this kind of authentication in combination with other features.

Grasp

The Smart Gun project, used a piezo-resistive sensor sheet embedded in the grip of a prototype gun to measure the static pressure of the hand-grip when the gun is being held in order to identify its owner [121]. They used Support Vector Machines (SVM) and Likelihood-Ratio Classifiers (LRC), and reported their technique to have an Equal Error Rate (ERR) average of 3.5% using LRC and 5.7% using SVM. While this result may not be sufficient for guns, which are only ever used in life threatening situations, this technique could be adapted to work with mobile devices where performance is less critical. Similarly, the use of grip pressure sensor has shown to be able to differentiate between car drivers from the way they grasp and grip the steering wheel [22].

2.3.2 Touch Interaction

Analogous to desktop computing, touch interaction is an explicit interaction method that requires full engagement between user and the device. However, despite being the *de-facto* interaction method in today's mobile devices, it still possess some interaction issues. One of the common problems is hand/finger occlusion – a situation where the touch target is blocked by the user's finger/hands which impairs the accuracy of touch. Implicit interaction has been used to reduce occlusion problem. In general, this method is based around the idea of implicit sensing using mobile device's internal sensors such as accelerometer or additional sensors such as capacitive sensor to sense the user's *unconscious* actions that associated with touch (i.e. hand grips, grasps, tremors). The following Sections discusses on implicit approaches proposed to improve the quality of touch interaction using touchscreens.

Pre-Touch

This method uses information sensed before the actual touch in order to estimate the actual touch with the purpose to improve the quality of touch interaction. The pre-touch sensing method in mobile device was first demonstrated in *SmartPad* where they used position sensors based on capacitive sensing, embedded underneath the phone keypad to recognise finger proximity before the user actually presses the key [104]. *PreSense* keypad is an extension of *SmartPad* using a similar sensing technique to recognise which button is about to be pressed [103]. Both used the proximity information from the sensors to initiate content previews on the screen prior to touch to help the user to decide before pressing the key.

The use of capacitive-based touchscreens in modern mobile devices in theory, offers a huge prospect for pre-touch sensing. However, most modern touch-based devices use mutual-capacitance touchscreens resulting in weaker capacitance signal for pre-touch sensing purposes. Sony developed a capacitive touchscreen technology called *Floating Touch*[2] that runs both self-capacitance (stronger signal but only capable to sense only one finger) and mutual-capacitance sensors at the same time to measure finger proximity from the touchscreen whilst allowing multi-touch interaction. Besides *Floating Touch*, similar technology also is used by Synaptics' *ClearPad*[3] and Fogale's *Sensation*[4] touchscreens.

The application of pre-touch sensing using capacitive touchscreen has been used in *Air Hook* – an interface that estimates the distance of finger or stylus pen from the touchscreen [149]. This valuable information is then used to preload remote and local data before the user performs touch actions, allowing the device to response faster. Ostberg and Matic on the other hand used hover cursor interaction by leveraging *ClearPad* touchscreen in Samsung Galaxy S4 to assist with small-target selection [93]. Their method produces fewer selection errors when compared to standard touch target selection method. Hinckley et al. explored three interaction design strategies that can be offered by pre-touch sensing [53]. Using a prototype mobile device equipped with Fogale's *Sensation* touchscreen, they demonstrated how pre-sensing can be used to create dynamic user interface (i.e. present appropriate interface as fingers approaching the screen), as an additional information to support touch actions (i.e. distinguish between flick and select) and as hybrid touch and hover gesture for menus.

Besides capacitive touchscreens, other input sensors such as 3-dimensional capacitive sensors [142], infra-red (IR) sensors [131], proximity sensors [13, 73] and magnetic sensors [67] have been used to acquire approximation of finger distance from the touchscreen/device. Outside of mobile devices, pre-touch sensing also has been used in interactive tabletop systems [102, 25] and notebooks [27].

[2] http://developer.sonymobile.com/knowledge-base/technologies/floating-touch/

[3] http://www.synaptics.com/products/touch-controllers

[4] http://www.fogale-sensation.com/technology

Back–of–Device

Back-of-device (BoD) is an interaction method that provides the possibility of extending the small interaction space of mobile devices through the effective use of unused physical surfaces at the back and around the device. This can be especially useful for small devices where the selection of a target is extremely difficult [6]. BoD also can be very useful for interactions when the device screen is not in the user's sight. In the context of implicit interaction, BoD has been used to complement standard touch interaction methods.

Unlike touch, interaction with BoD systems is normally initiated using the palm of the hand. Therefore majority of the work on implicit BoD interaction uses information associated with palm such as hand grip and grasp. For instance Chang et al. [19] created a mobile user interface (UI) that takes hand grip as input. Through this UI, the phone application is launched based on a particular grip pattern. In order to achieve this, they used 64 capacitive sensors distributed at the back and sides of a mobile phone to capture hand grip patterns. Naive Bayes (NB) and support vector machine (SVM) classifiers were used to recognise grip patterns associated with 5 common mobile applications: phone dialler, text messaging, camera, video and game applications. Their recognition system was able to achieve average accuracy of 79% (NB) and 96% (SVM) in all applications.

Although the built-in accelerometer has been used to automate device screen orientation, in some situations, such as lying on one side, users may want to keep current viewing orientation regardless of device orientation. In order to allow finer control on screen rotation, Cheng et al. [23] proposed a screen rotation method based on hand grasp. Their intuition is based on the fact that users grasp the device differently in protrait and landscape orientations when lying on one side. Similar to technique used in grip-based UI [19], grasp data was acquired via 44 capacitive sensors laid on the back and sides of a phone protoype. Recognition of portrait and landscape grasps was done using SVM classifiers which they reported to achieve classification accuracy of 75% for portrait, 66% for left-landscape and 93.3% for right-landscape.

Another use of grasp input is for soft keyboard positioning and layout adjustments. Through this technique, grasp is used to determine position and layout by sensing the way the device is being grasped. For instance grasping with two hands may require split keyboard layout to better suit typing using thumbs. In this work Cheng et al. [24] used 46 capacitive sensors placed at both long edges of a protoype tablet to capture grasp patterns and compared typing performance between manual and automatic adjustments of the keyboard. Their method shows improvement in typing speed from 24.8 to 26.6 word per minute (WPM).

Bar of Soap on the other hand is a rectangular object developed by Taylor and Bove Jr. [132] to explore the use of an object as grasp-based user interface. The object which measures $11.5 \times 7.6 \times 3.3$ cm uses 72 capacitive sensors placed around the box to sense grasp. This study is similar to grip-based UI in [19], however rather than focusing just on grasps for

mobile phone applications, Bar of Soap explores other grasps of which users have previous experiences such as the *rubric* cube. Ball of Soap is a similar grasp sensitive object based on a small rhombicosidodecahedron with 62-sided Archimedean solid. The grasp recognition system is evaluated using two grasp manipulation studies: 1) rubric cube (Bar of Soap) and 2) pitch selection (Ball of Soap). They used six classifiers namely template matching, neural networks, K-nearest beighbours, parzen windows, Bayes and general linear discriminants (GLD). Their system was able to recognise individual user's grasp with over 90% accuracy, indicating that grasp has substantial information for recognition.

2.3.3 Error Detection

Perhaps, the biggest sources of errors in human computer interaction are indeed the users themselves [101]. There are a number of factors associated with user errors such as information overload, interface complexity as well as other cognitive factors. Reason [101] defined two types of error in HCI: 1) slip 2) mistake. Slip is when the action made is not what was intended whereby mistake is when the action is not appropriate. For instance using a dialler application to text someone is called a mistake, however calling or texting a wrong person is called a slip. Both types of error can happen during the course of interaction however according to Sternberg, slips are more prone to happen if a routine is deviated [126],

Detecting interaction errors has been of great interest to the HCI community. There is abundant empirical literature exploring methods to recognise and detect human errors from interaction [136, 137, 79, 34]. This is mostly based around the idea of observing a specific *indicator* that is highly correlated with interaction errors. Typically, this indicator is the result of a natural response (interaction) which is unknown (implicit) to the users. Examples of response being used are brain signal, breathing and heart rates. Studies have shown that errors can recognised by observing these indicators through the use of physiological as well as non-physiological sensors. The following sections give an overview on the approaches using both type of sensors.

Physiological Approach

Most established approaches to recognise errors in HCI are achieved through brain computer interface (BCI) [89]. When people make errors, there is a characteristic signal observable in EEG – the Error-related Negativity (ErrN). ErrN is a pattern that appears in the ongoing EEG signals when users have feedback about their response accuracy [54]. It is also known to appear when users are confused or unsure about their actions [115] (i.e. without explicit correct/incorrect feedback).

In HCI, ErrN signals have been shown to be useful in detecting errors in spelling [34, 87, 21]. Besides BCI spelling applications, ErrN has also been used to detect errors in diverse interactive tasks [136] [137]. These studies used a commercially available EEG headset to capture ErrN signals triggered from Flanker, button selection and pointing tasks. Using a logistic regression classifier, ErrN signals were classified with an accuracy of about 70%, 65%, 80% in Flanker, button selection and pointing tasks respectively. EEG-based BCI has also been used to detect ErrN during tactile human-machine interaction [79].

Various attempts have been made to bring BCI to mobile devices. *NeuroPhone* for instance is a mobile phone that allows user to interact with mobile application using P300 neuro signals from the EEG headset [15]. **?** on the other hand proposed a software platform for EEG oriented applications. Although EEG offers extra dimensionality of information, they however required end-user to wear additional peripheral which makes it impractical in almost any mobile context. Similarly, wireless EEG headband has been used to interact with the mobile phone dialing application [140]. Despite potential use in initiating interaction using the brain, no work has been done to study the presence of ErrN signals when interacting directly with a mobile phone.

Whilst using brain waves to detect error is well established, it is not a practical approach for most interactions. There is less work on physiological signals and body channels which might produce similar signals when a mistake is made (other than measurements such as GSR or heart rate modulations, which tend to be too slow to identify individual errors in an interaction [122]).

Non–Physiological Approach

Outside of BCI, there is less research on physiological indicators of error in human computer interaction, particularly in mobile contexts. There are sophisticated auto-correct mechanism which detect and correct typing mistakes on mobile device through various techniques such as keypress timings [29], touch position distributions [44] and geometric pattern matching [75]. Weir et al. on the other hand have used touch pressure as an uncertainty indicator before correcting the typing error [141]. Although these approaches can reduce typing time and error levels they can also be irritating [85].

Inertial sensors such as accelerometers and gyroscopes have been fundamental to mobile human activity recognition systems [100]. There have been attempts to study unusual or anomalous human dynamics from sensor data [16] to detect of unexpected events or accidents [80, 134, 4], although these focus on large scale events (falling over) rather than micro-movements related to touch screen gesturing.

2.4 Conclusion

This chapter has discussed the important distinction between explicit and implicit interactions. This includes the role of implicit interaction in mobile devices. A discussion on sensor-based approach to implicit interaction techniques has been given. Finally, literature related to the work that will be presented in the rest of this thesis has been reviewed, with particular focus on three application domains – where a number of examples from the literature were presented.

Chapter 3

Hardware and Experimental Tools

Summary
In this chapter, we give a review of the related work found in the literature that employ motion and capacitive sensing. We provide a brief discussion on the decision of using motion and back-of-device (BoD) capacitive sensing in our work. Finally, we provide technical specifications of our hardware prototypes, experimental user study tools and BoD sensor design at the end of this chapter.

3.1 Introduction

This thesis is based on studies on inferring interactions from two input modalities: 1) hand motion and 2) hand grip. In general, this involves modelling the data acquired from the sensors to make inferences from it. Acquiring hand motion data from a mobile device is a very straightforward procedure since majority of mobile devices today have built-in accelerometer and can be easily accessed via Application Programming Interface (API) provided by the mobile operating system (OS) such as Android and IOS. Although there are many variations of accelerometer used in mobile devices, these sensors typically measure acceleration along three orthogonal axes (x, y and z). Depends on the application context, accelerometer readings can be used as either 1) measurement of velocity and position, 2) 3-dimensional orientation sensor or 3) vibration sensor. In this chapter we focus on the use of accelerometer as a 3-dimensional orientation sensor to capture hand motion.

Accelerometer has been widely used to study motion-based interaction with mobile devices. Prior to using mobile phone internal accelerometer, researchers relied on the external accelerometers to measure hand motions particularly related to gesture. For example, *SoapBox* is a wireless sensor device equipped with accelerometers used to capture hand motion gestures for operating video cassette recorder (VCR), lighting system and 3D software appli-

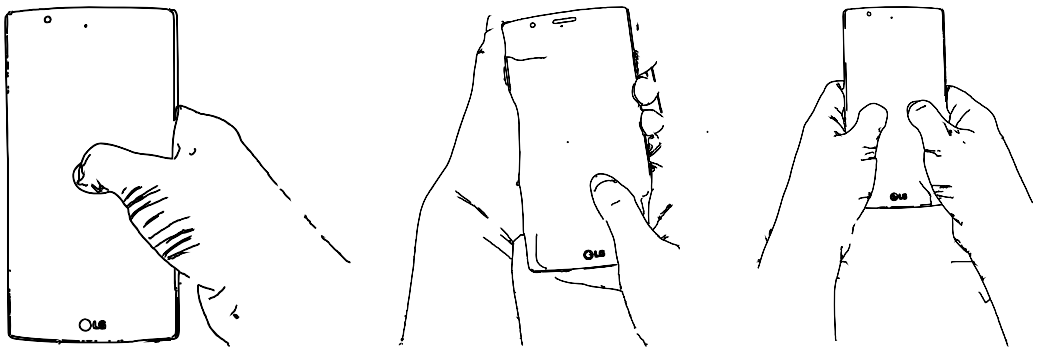
cation using hand gestures [65]. Similarly, to explore the variability and dynamics of hand motion gestures, Williamson and Murray-Smith used an external sensor pack mounted on the hand [143]. Apart from hand gestures, external accelerometer also is a popular sensor for studying whole body interaction. For instance external accelerometers have been used to recognise human daily activities such as walking, running and standing [100, 5].

The inclusion of accelerometer in mobile phones has accelerated many research efforts pertaining mobile interactions. Although the primary use of accelerometer is to sense device orientation, researchers have explored other potential uses of accelerometer such as for identification [57], gesture-based interaction [26] which can be used as input for game [59]. Fitness-based applications such as Google Fit[1] and Strava[2] leverage built-in accelerometer to track and recognise human activities. In health, accelerometer in mobile phone has been used to monitor cardiac outpatients [139] and many other home-based rehabilitation programmes [71, 138, 95]. Beyond mobile phones, accelerometer is an established input modality in wearable devices such as smartwatches and activity trackers. From this, many new interaction methods based on hand movements have been explored such as finger-based gestures [147, 153], office-space gestures [9] as well as for detecting anomalous activity [48].

Although most modern mobile devices come with multitude of sensors, none of these sensors are capable to sense hand grip. In nearly every case, users hold their mobile devices using one of the following basic grips: 1) one-handed use, 2) cradling with two hands and 3) two-handed use [127]. As shown in Figure 3.1, in all positions, hand will be gripping the back and sides of the phone. Based on these positions, it is sensible to place sensors at these locations in order to sense grips. In terms of sensing technologies, researchers have explored various sensors based on capacitive [70, 23, 24, 132, 112, 144], force/pressure [121, 22, 49, 76], optical [94, 145], light/infrared [47] and proximity [13] technologies to acquire grip related information. However, capacitive technology is expected to be more adaptable on mobile devices compared to other sensing technologies since it is a proven sensing technology (core sensing method on touchscreens) and relatively low in cost. Even though grip sensing is yet to be implemented in mobile devices, several phone manufacturers have begun to include **BoD** sensing as an additional input to complement standard touch interaction. For instance Oppo NI[3] use a touch sensitive panel at the back of the device to allow users to interact using index finger – this could help users to navigate single handed without the need to change their grip position (i.e. from single handed to cradling).

As mentioned earlier, sensor location is important in order to capture salient information pertaining hand grip. Although some devices (i.e. DG800 and Oppo N1) provide such sensing capability, they are purpose-specific and do not provide enough coverage to sense whole

[1] http://fit.google.com/
[2] http://www.strava.com
[3] http://www.oppo.com/en/smartphone-n1



(a) One-handed (b) Cradled (c) Two-handed

Figure 3.1: Typical mobile device grip positions [127].

hand grip. Due to this, we designed a capacitive sensing system that can be attached at the back and sides of a mobile phone. By placing capacitive sensors at these locations, we could sense a wider grip contact from the hand, thus allowing us to observe more grip variations which is essential in all of our experiments. For this thesis, we built two prototype systems with BoD capacitive sensing capabilities based on the actual smartphones. Section 3.2 gives an in-depth specification on how we achieved these systems.

Capacitive-based sensing is a popular technology in the HCI community, particularly when investigating grip and grasp based interactions. For instance Kim et al. used capacitive electrodes at the back and sides of their prototype device to acquire hand grip contacts, learn the patterns and subsequently map them to the intended mobile applications [70]. Similarly, Cheng et al. attached capacitive sensors at the back and sides of an iPod touch to sense and recognise grasp to automate screen orientation [23]. Same sensing method also has been applied on a tablet in order to make the layout and position of the virtual keyboard changes according to the way the tablet is grasped [24].

Besides augmenting standard mobile devices with capacitive sensors, researchers also used dedicated prototypes developed specifically for grasp-based sensing system. *Bar of Soap* [133] for example, is a rectangular box with capacitive sensors covering the back and sides of the box to sense grasps. Using this prototype, Taylor and Bove demonstrate how grasps can be used to differentiate different between camera, phone and remote control poses [133]. In the follow-up study, Taylor and Bove Jr. used *Ball of Soap* – another prototype with similar functionalities as *Bar of Soap* but in spherical form factor to measure the way users hold and manipulate objects [132]. This concept is similar to Tango – a graspable input prototype for 3D environments. However, unlike *Bar of Soap*, Tango measures hand grip contact pressure instead of capacitive contact of hand [76].

Besides capacitive sensing, researchers also have successful used force/pressure sensing technology to acquire hand grip information. *GripUI* for example is a mobile user interface that uses grip pressure to interact [49]. To acquire pressure from grip, Higuchi and

Okada used pressure sensors placed at both sides and top half at the back of the phone prototype. As mentioned previously, Tango [76] is a ball-like prototype that uses hundreds of pressure sensors to track grasps with the focus to recognise hand shapes. Tango offers a free-form whole hand interaction which was previously achievable only through glove-based approaches [130]. Force and pressure sensors also have been used as an input to biometric system. For instance, Shang and Veldhuis used piezo-resistive elements underneath the pistol grip to acquire grip patterns for identification purposes [121]. Apart from gun, similar grip-pressure based identification technique also can be used to identify car driver. To acquire driver's grip, the pressure sensors were embedded in the steering wheel [22].

The purpose of this chapter is to give technical specification of hardware prototypes including sensing technologies that we used in our work. As stated earlier, motion and capacitive sensing technologies were chosen due to their popularity and have been actively researched within HCI community. Moreover, **BoD** capacitive sensing also has been adopted by several phone manufacturers and is expected to be included in more mobile devices in the near future. We developed two prototypes with **BoD** capacitive sensing capability to capture hand grip information. For hand motion, we leverage the built-in accelerometer. The following section discusses about our first prototype system including work that employs this system. Our second prototype is discussed in Section 3.3.

3.2 Prototype 1

Our first prototype system consists of a Sony Xperia Mini Pro (SK7i) smartphone, SHAKE SK7 sensor pack, an external array of capacitive sensors on a flexible printed circuit board (PCB) and a plastic housing. The role of the smartphone in this system is to log data from the external capacitive sensors. This is achieved via bluetooth connection from SHAKE SK7 which acts as an interface to the capacitive sensors. For this prototype, we used 24 capacitive sensors pads positioned at the back and sides when attached to the plastic housing. As shown in [127], users tend to apply grip at this location when interacting with mobile phone. Figure 3.2 (a) shows each component of the prototype. When assembling the whole system, the SHAKE SK7 is kept at the bottom of the housing (see Figure 3.2 (b))

The external capacitive sensors have 8 bit dynamic range – each sensor gives proximity values between 0 (not touched) and 255 (touched). The capacitive sensors are quite sensitive to the way the prototype is held and also due to the gaps between the sensor pads and surface of the plastic housing. To overcome this, we calibrated each sensor by correcting the offset error prior to every experiment. The total size of this prototype is 65mm (H) x 150mm (L) x 25mm (D) and weighs approximately 145 grams. As can be seen in Figure 3.2 (b), the size of this prototype is slightly larger than the average sized modern smartphones, however we

(a)

(b)

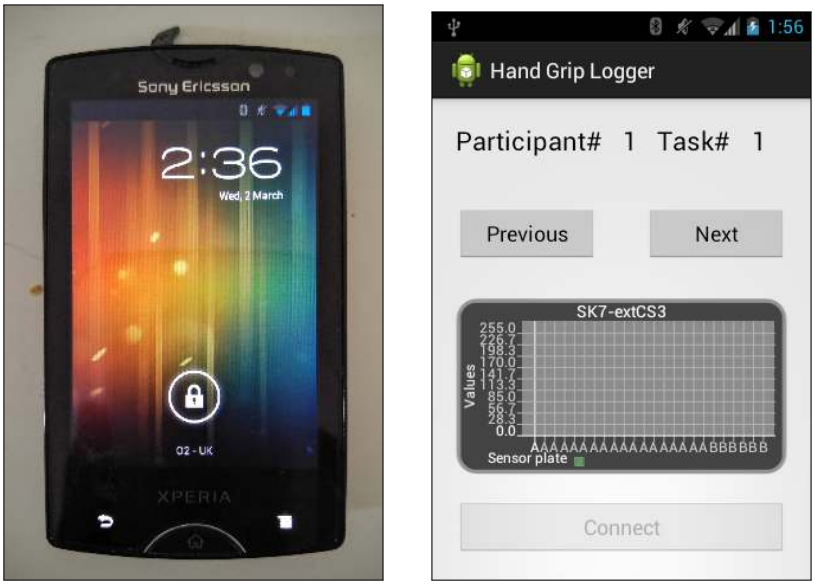
Figure 3.2: (a) Individual component of Prototype 1 (b) Prototype 1 when fully-assembled (SHAKE SK7 is hidden underneath the front cover).

feel it still relevant in our study since it has the 'bar' form factor of a mobile phone.

3.2.1 Sony Xperia Mini Pro

The Sony Xperia Mini Pro is a smartphone running Android version 4.0.4 (see Figure 3.3(a)). The dimension of this phone is 92mm (H) x 53mm (W) x 18mm (D) and weighs 136 grams. The phone diminutive size allows it to be fit inside the plastic housing easily. It has 76mm capacitive touchscreen display with resolution of 320×480 pixels. Internally, the phone is powered by 1GHz single-core processor with with 512 MB of RAM. The phone uses BMA250 accelerometer, a 3-axis accelerometer manufactured by Bosch Sensortec. It has acceleration range of $\pm$2g with 12-bit dynamic range.

An Android data logging application (Figure 3.3 (b)) was developed on the phone to log data from both external capacitive sensors and phone's internal accelerometer (see Figure 3.3(b)).



(a) Sony Xperia Mini Pro (b) Logging application

Figure 3.3: (a) Sony Xperia Mini Pro (b) Screenshot of data logging application.

Data from the capacitive sensors were transmitted from SHAKE SK7 via bluetooth connection, whilst data from the accelerometer was acquired directly from the Android API. In terms of sampling rate, the SHAKE SK7 is capable to stream data with frequency up to 256Hz whilst for the Android, we request the values from the API using the highest possible sample rate. In order to synchronise data streams from both sensors (accelerometer and SHAKE SK7), we fixed the sampling rate of the logging to 100Hz by writing values from both sensors to a file in comma separated values format (CSV) into phone's internal storage.

3.2.2 SHAKE SK7

Figure 3.4: SHAKE SK7 sensor pack

The SHAKE SK7 is an external sensor pack developed by SAMH Engineering Services that comes with various sensors, including accelerometer, gyroscope, magnetometer, digital compass and vibro-tactile motor. Shake SK7 is a valuable tool used by researchers in the areas of HCI and has been used in many work involving motion gestures [143], touchscreen

interaction [8] and in understanding human personality [110]. The sensor pack dimension is 43mm (L) x 32mm (H) x 18mm (D) and weighs 22 grams. SHAKE SK7 can communicate through bluetooth or emulated serial communication (COM) port via universal serial bus (USB).

In this thesis, we used SHAKE SK7 as an interface and controller to the external capacitive sensors. This is done by connecting the external capacitive sensors to a 4 pole 2.5mm socket (auxiliary socket) to allow SHAKE SK7 to communicate with the sensors using the I^2C protocol. We used bluetooth to communicate and transmit data from SHAKE SK7 to the logging application.

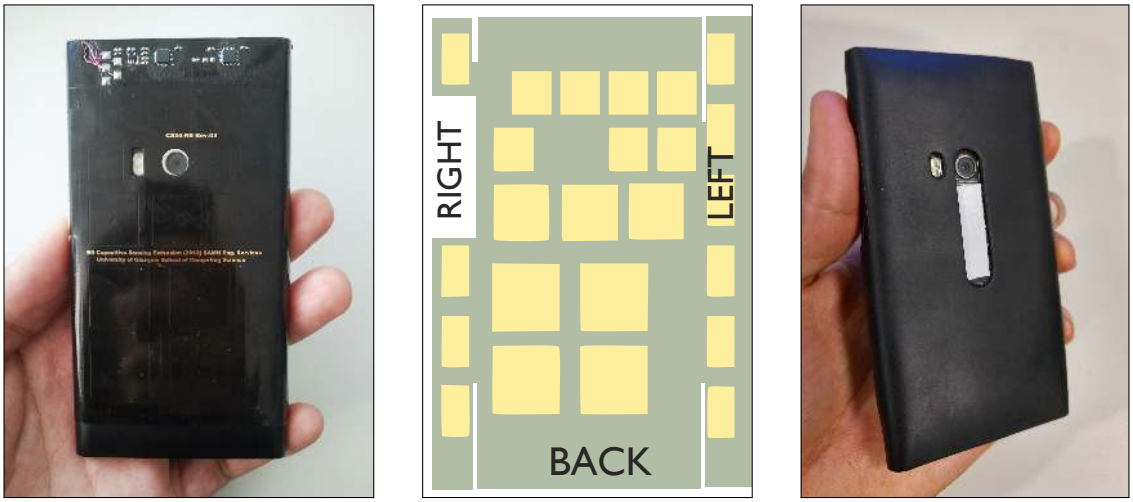
3.3 Prototype 2

Our second prototype is based on Nokia N9 smartphone. The device has dimension of 61.2mm (W) x 116.45mm (H) x 12.1mm (D) and weighs 135 grams. It has 99.1mm capacitive touch display with 854 x 480 pixel resolution. The phone runs MeeGO 1.2 operating system and is powered by a single-core 1GHz processor with 1 GB of RAM. It uses a 3-axis LIS3LV02DL STMicroelectronics accelerometer that has acceleration range of $\pm 2g$ with 12-bit dynamic range. The sensor has been pre-configured to sample data at 160Hz.

We added capacitive sensors around the device in order to sense hand grip. This is achieved by using an external array of 24 capacitive pads on a flexible PCB wrapped around the N9. Figure3.5 (b) shows the capacitive pads arrangement. This asymmetrical layout of sensor pads was chosen since we believe it could capture grip contacts from typical grip positions [127] effectively. As shown in Chapter 5, 6 and 7, this layout has shown to give good results in the predictive tasks. We expect similar results if a symmetrical layout version of ours is used. However different sensor layouts such as used in [70, 144] and [24] may have consequence on the results.

Unlike the approach we used in the first prototype, the external sensors are wired directly to the phone through the internal I^2C bus interface provided by the N9 (see Figure 3.5 (a)). As shown in Figure 3.5 (c), we fit a silicon case to the N9 to protect the sensor pads from grime/dirt which could affect the contact capacitance signals. The modified N9 with around device capacitive sensing adds an approximately 1mm of thickness to the original dimension however this does not affect the original form factor of the phone.

The external capacitive sensors use 16-bit data representation – each sensor gives contact capacitance value between 0 (no contact) to 65535 (full contact). In comparison to our first prototype, this range provides a much finer contact measurement and can also be used to represent grip pressure. Each sensor is calibrated prior to every experiment by correcting the offset error.



(a) (b) (c)

Figure 3.5: Nokia N9 with **BoD** sensing.

The acquisition of acceleration data from the built-in accelerometer is done by reading directly sensor values from the operating system file (a file handle to I^2C system call) via N9 Linux environment, which MeeGo is based on. The use of Linux as system backend in the N9 allows us to have full control over the operating system and access to various system files. To acquire capacitive data, we wrote a custom Linux kernel module to allow the N9 to communicate with the external sensors through the I^2C bus. This creates a file handle to which the data obtained from the sensors is stored. We used a Python logging application which we wrote on the N9 to read the data from both sensor system files. This process was done at 100Hz. The application then writes the data to a file (CSV format) and stores the file in the phone internal storage.

3.4 Capacitive Sensing Technology

Our BoD sensors are based on capacitive sensing technology that measures capacitance changes from single electrode sensors. The sensor electrode on the PCB comprises one plate of a virtual capacitor. The other plate of the capacitor is the user's finger, which is grounded with respect to the sensor input. This sensing method also is known as *transmission* or *loaded* method. As shown in Figure 3.6, the excitation source charges the plate of the capacitor and when human body such as finger (or other grounded object), comes close to the sensor electrode, the virtual capacitor is formed, with the user's finger acting as the second capacitor plate.

When a finger approaches the sensor, the total capacitance associated with the sensor changes. Therefore in order to register a sensor activation, the total capacitance needs to exceed a cer-

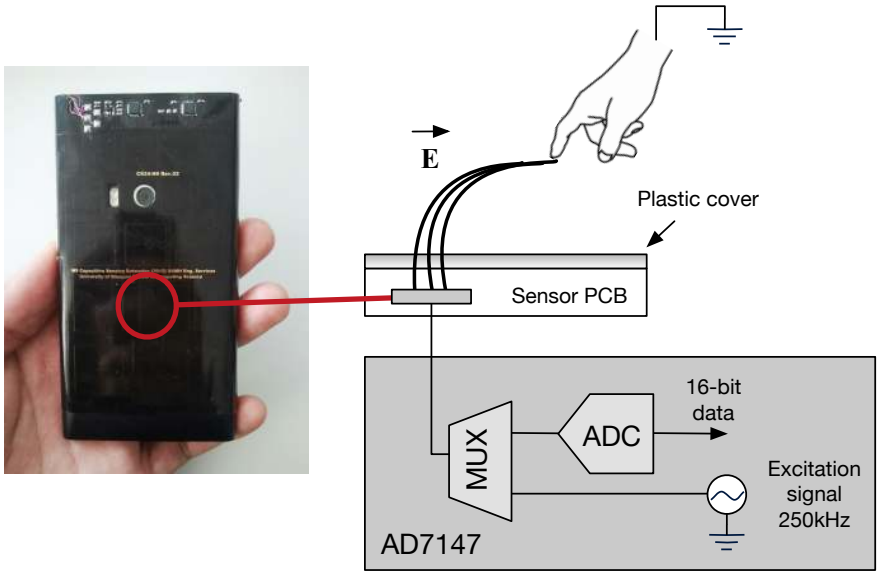


Figure 3.6: BoD capacitive touch sensing method. (Adapted by author from Analog Devices AD7147 data sheet [117]).

tain of threshold. Since it uses threshold for activation, this sensing method also can be effectively used to measure non-contact proximity, for instance when the finger is approaching the sensor (pre–touch sensing). Moreover, the sensor is grounded object activated, therefore it can be concealed under any non-conductive material (e.g. plastic housing, silicon case), making it a practical sensing solution for our work.

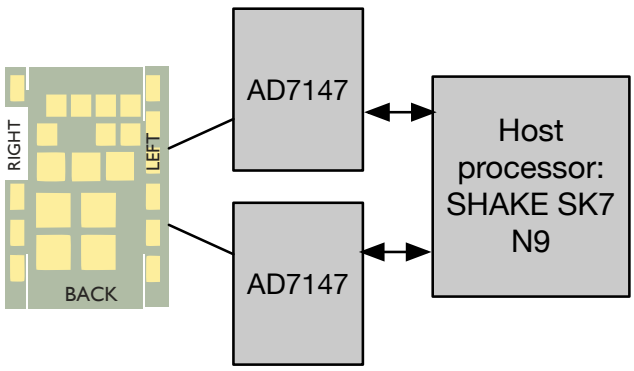


Figure 3.7: BoD capacitive sensing solution.

In our setup, we use Analog Devices AD7417 capacitance–to–digital converters (CDC) which contains the excitation source and a 16–bit ADC while the electrode is constructed on a flexible PCB as an external capacitance sensor. The AD7417 can interface up to 12 capacitance sensors, therefore we used $2 \times$ AD7417 to interface 24 BoD sensors in our prototypes. AD7417 uses I^2C interface to communicate with the SHAKE SK7 and N9. Our BoD sensing solution is illustrated in Figure 3.7.

3.5 Conclusion

In this chapter, description and technical specifications of both hardware prototypes used through out this thesis are given. We state the factors that influence our decision to use motion and capacitive sensing technology for implicit sensing. We focus on the technical aspect of the hardware by showing how back and around device grip sensing is added to our prototypes and techniques to acquire data from these sensors for logging purposes. Although external capacitive sensors were used in our prototypes, this does not affect the original form factor of the mobile device, thus making our prototypes still relevant in all the work presented in this thesis.

Chapter 4

Prediction Using Machine Learning

Summary Machine learning is one of the most important components in this thesis. In this chapter, we provide an in-depth discussion of machine learning techniques we used to model implicit interaction from mobile sensor data. We also discuss about exploratory data analysis we use to inform model choices. We discuss our core prediction techniques: classification and regression. Model selection and evaluation techniques, including performance metrics are given at the end of this chapter.

4.1 Introduction

This chapter provides references to the analysis techniques we use throughout this thesis. We use machine learning techniques to explore patterns in the sensor data, model them and subsequently make predictions using sensor data.

Data exploration is a preliminary analysis that seeks to understand about the sensor data. Depends on the goal, there are many exploratory analysis techniques can be used for this purpose. In this chapter we discuss about Principal Component Analysis (PCA) whose goal is to uncover important structures in the data and Canonical Correlation Analysis (CCA) where we are interested to understand relationship between two multivariate datasets.

Our machine learning framework involves four main components: 1) input feature 2) training 3) algorithm and 4) evaluation. This is illustrated in Figure 4.1. The first component involves identifying characteristics from the data that best represent the problems to be modelled. The second component involves the process of establishing a model by *learning* from a set of examples, which makes our learning tasks *supervised*, in that it uses labeled values or measurements to learn patterns in the data. The third component is selecting supervised learning algorithm to model the data. Depends on the task, this can be either classification or

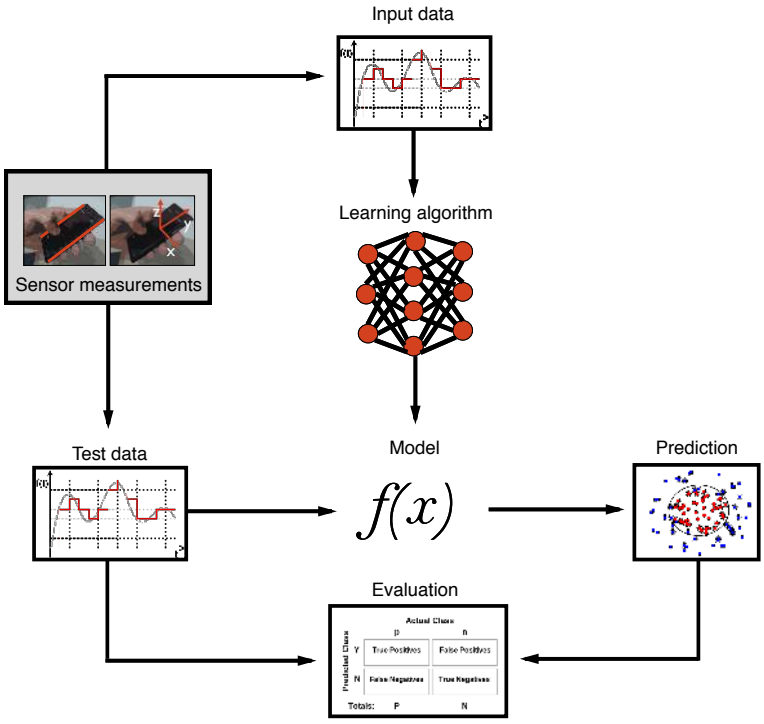


Figure 4.1: Machine learning main components.

regression problem, where the output takes discrete class labels in the former and the latter takes continuous values. We use both types of learning algorithm in this thesis.

Given the models, the next step is to make predictions on new and unseen data and evaluate how good is the model in predicting the unseen data (test data). This makes the final component of our framework. There are many performance measures can be used to evaluate the model. A detailed discussion about performance measures we use to evaluate the models is given in Section 4.6.

This chapter is organised as follows:

- Section 4.2 discusses the initial exploratory data analysis we took to gain insight into patterns that exist in the data.

- Section 4.3 and 4.4 describe two supervised methods we used to make predictions. Evaluation methods for model selection.

- Section 4.6 gives a review on all evaluation metrics we used to assess prediction performance in this thesis.

4.2 Exploratory Data Analysis

We perform exploratory data analysis (EDA) as the first step in our analysis in all of the work. EDA allows us to build an intuition for data which can be used as the basis for choosing machine learning techniques. This pragmatic approach was promoted by Tukey in 1977 [135]. EDA involves extracting important statistics from the dataset and visualising them. EDA helps to uncover and expose important underlying structures and could highlight outliers in the dataset.

In order to identify patterns and detect potential outliers in data we perform linear transformation using Principal Component Analysis (PCA). This transformation is described in Section 4.2.1. In Section 4.2.2, we show the techqnique to investigate the relationship between two multivariate input variables using Canonical Correlation Analysis (CCA). Finally in Section 4.2.3 we use Mutual Information Analysis as a basis to perform feature input selection.

4.2.1 Principal Component Analysis

Patterns in two and three dimensional data can be easily visualised by plotting them on a graph in their respective axis. However as the number of dimension increases, plotting them using their original axes is no longer possible. PCA is a technique to identify patterns that exist in data by highlighting their differences and similarities by *viewing* them from a particularly good angle. PCA can be a powerful tool for analysing high dimensional data.

In this thesis, we mainly deal with high dimensional datasets and thus PCA is a sensible technique to analyse and reveal the underlying patterns. Although there are other techniques that can be used to reduce dimensionality (e.g Factor Analysis), PCA is chosen because we are only interested in reducing dimensions of the data for visualisation and inspection purposes. PCA is used in Chapter 5 and 7 to explore the data in two-dimensional space.

PCA transforms data by performing *linear* projection of data into a lower-dimensional space [10]. Linear projection here means that each projected dimension is a linear combination (sum of vectors multiplied by a scalar) of the original dimensions. The linear combination x_n can be defined as:

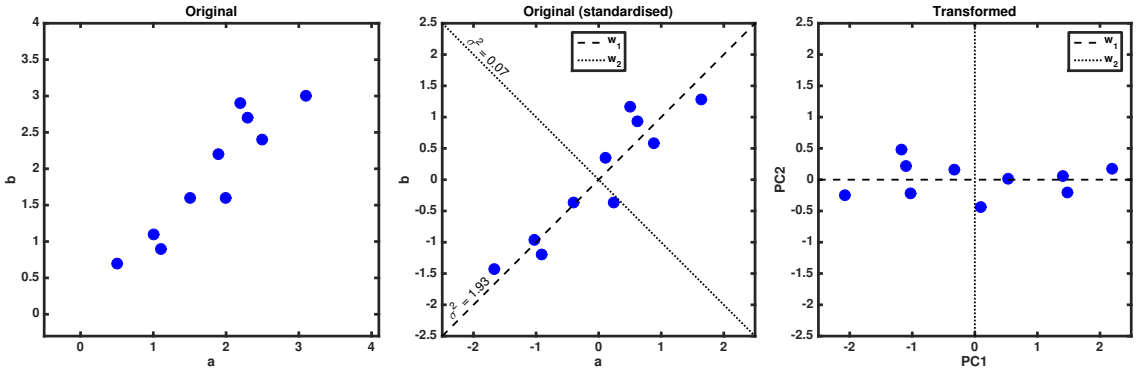
$$x_n = \mathbf{w}^\intercal \mathbf{y}_n$$

Therefore if we are interested in projecting $\mathbf{y}_n$ from M to D dimensions, there will be M-dimensional of D vectors $\mathbf{w}_d$ and the d-th element of the projection x_{nd} (where $\mathbf{x}_n = [x_{n1}, x_{n2}, \ldots, x_{nD}]^\intercal$) is therefore:

$$x_{nd} = \mathbf{w}_d^\intercal \mathbf{y}_n$$

The goal of PCA is to choose $\mathbf{w}_d$ such that the variance in x_{nd} is maximised. The first principal component $\mathbf{w}_1$ is the projection that captures the most variance. The second component $\mathbf{w}_2$ captures the most variance but is orthogonal to the first, that is $\mathbf{w}_1^\intercal \mathbf{w}_2 = 0$:

$$\mathbf{w}_i^\intercal \mathbf{w}_j = 0, \quad \forall j \neq i$$



(a) (b) (c)

Figure 4.2: Dataset $\mathbf{X}$ in the original (a) standardised (b) and transformed (c) space.

Covariance for $\mathbf{X}$ can be defined as follows:

$$\mathbf{C}_\mathbf{X} = \frac{1}{N}\mathbf{X}^\intercal\mathbf{X} \tag{4.1}$$

The next step is to find eigenvector $\mathbf{u}$ and eigenvalue $\mathbf{w}$ pairs from covariance matrix in 4.1.

The eigenvectors are then ordered according to its respective eigenvalues, from the highest to lowest. The eigenvalues show how much variance can be explained by its associated eigenvector and therefore, the highest eigenvalue indicates the highest variance in $\mathbf{X}$ observed in the direction of its eigenvector. Eigenvector with the highest eigenvalue is called *principal component* which represents the projection axes of the maximum variance.

Figure 4.2 (b) shows the first $\mathbf{w_1}$ (dashed) and second $\mathbf{w_2}$ (dotted) principal components for $\mathbf{X}$ along their respective variance. Accordingly, variance explained by each component also can be expressed in terms percentage of the total variance explained by each principal component:

$$\sigma_P^2 = \frac{\lambda_m}{\sum_{m=1}^{M} \lambda_m} \tag{4.2}$$

σ_P^2 also is called proportion of variance explained, which tells how much of variance is accounted relative to overall variance captured by each component.

To transform the data into new axes, simply multiply the data with the eigenvectors: $\mathbf{X}' = \mathbf{w}^\intercal \mathbf{X}^\intercal$. Figure 4.2 (c) shows the projection of $\mathbf{X}$ into new axes, namely the first and second principal components.

4.2.2 Canonical Correlation Analysis

Correlation analysis can be used to quantify the relationship between two univariate datasets (e.g. measuring relationship between body height and weight). On the other hand, multiple correlation analysis is normally used when the analysis involved of several independent variables and one dependent variable (e.g. relationship between temperature and humidity against mould growth rate). However when the analysis involves several independent variables and several dependent variables, multiple correlation analysis can no longer be used to measure the relationship. In order to measure such relationship, canonical correlation analysis (CCA) can be used.

CCA was developed by Hotelling [55] and has been widely used in many areas such as in economics [88] and medical studies [118]. Computation of CCA leads to the extraction of more than one set of linear combination of weighted sums (canonical roots) from each dataset. Specifically the number of canonical root will be equal the minimum number of dimension in either dataset. For instance if the dimension of the first and second datasets are 5 and 3, then the maximum number of canonical roots that can be extracted will be 3. When extracting more than one canonical root, each successive canonical root will explain additional variability of the two datasets, which means that they are uncorrelated with the previous extracted canonical root and decreasing in variability.

Computing Canonical Correlation

In order to measure the correlation, CCA finds new derived variables called canonical variates from each dataset and find the correlation such that it maximises the linear correlation between the two. Each canonical variate is formed through a linear combinations of weighted sum score of the variables. For example, consider datasets $\mathbf{X}$ and $\mathbf{Y}$ with M and N variables respectively.

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

where $M \leq N$. Let W and V be the canonical variates of $\mathbf{x}$ and $\mathbf{y}$ respectively. Therefore the weighted sum scores of W:

$$W_1 = a_{11}x_1 + a_{12}x_2 + \cdots + a_{1M}x_M$$
$$W_2 = a_{21}x_1 + a_{22}x_2 + \cdots + a_{2M}x_M$$
$$\vdots$$

and V:

$$V_1 = b_{11}y_1 + b_{12}y_2 + \cdots + b_{1N}y_N$$
$$V_2 = b_{21}y_1 + b_{22}y_2 + \cdots + b_{2N}y_N$$
$$\vdots$$

can be defined as (W_i, V_i) as the i^{th} canonical variate pair. Since $M \leq N$, there are M canonical variate pair that can be computed. a and b correspond to the weighted sum coefficients. The correlation between (W_i and V_i) can be determined by calculating their respective variance:

$$\sigma^2_{W_i} = \sum_{k=1}^{M}\sum_{l=1}^{M} a_{ik}a_{il}\sigma_{x_k x_l} \tag{4.3}$$

$$\sigma^2_{V_i} = \sum_{k=1}^{M}\sum_{l=1}^{N} b_{ik}b_{il}\sigma_{y_k y_l} \tag{4.4}$$

and find the cross-covariance between W_i and V_i as:

$$\sigma_{W_i V_i} = \sum_{k=1}^{M}\sum_{l=1}^{N} a_{ik}b_{il}\sigma_{y_k y_l} \tag{4.5}$$

The specific correlation between W_i and Vi is called *canonical correlation* which can be calculated by taking their covariance and divide it by the square root of the product of the variances:

$$\rho_i = \frac{\sigma_{W_i Vi}}{\sqrt{\sigma^2_{W_i}}\sqrt{\sigma^2_{V_i}}} \tag{4.6}$$

The goal of CCA therefore is to find linear combinations that maximises ρ. For the first pair (W_1, V_1) the coefficients $a_{11}, a_{12}, \cdots, a_{1M}$ and $b_{11}, b_{12}, \cdots, b_{1N}$ need to selected such that ρ_1 is maximised, subject to the constraint that variances of W_1 and V_1 are equal to 1.

CCA in general requires that all of the remaining correlations to be zero:

$$\sigma^2_{W_i} = \sigma^2_{V_i} = 1$$
$$\sigma^2_{W_1 W_i} = \sigma^2_{V_1 V_i} = 0$$
$$\sigma^2_{W_2 W_i} = \sigma^2_{V_2 V_i} = 0$$
$$\vdots$$
$$\sigma^2_{W_{i-1} W_i} = \sigma^2_{V_{i-1} V_i} = 0$$

In other words, the first pair of canonical variate accounts the highest variability between $\mathbf{x}$ and $\mathbf{y}$ and each successive pair will explain a unique additional proportion of variability.

4.2.3 Mutual Information Analysis

The next exploratory technique looks into relationship between two variables by measuring how much of *information* is communicated on average, in one random variable about another. For instance, let random variables X represents a roll of fair die and Y represents the parity of the outcome of the roll (0 = even, 1 = odd). In this example, the value of X could tell about Y and vice versa. Therefore here X and Y are said to share *mutual information*. However if X is the roll of a die and Z is the roll of another die, then X and Z do not share mutual information because X does not contain any information about Z and vice versa. X and Z are *statistically independent*.

This is the basis to mutual information analysis (MIA) which is based on information theory. Entropy is a measurement of *predictability* or *uncertainty* of a random variable. For example, if random variable X represents the outcome of tossing a coin (1 = head, 2 = tail) and random variable Y represents the number that comes up from rolling a die, X is said to have greater entropy than Y because predicting head or tail is much easier than predicting the outcome of rolling die which takes values 1 through 6.

If X takes values from set $\mathcal{X} = \{x_1, x_2, \cdots, x_M\}$ and is defined by probability $P(X)$, then entropy of random variable X can be formally defined as:

$$H(X) = -\sum_{x \in \mathcal{X}} P(x) \log_2 P(x) \tag{4.7}$$

When dealing with a pair of random variables simultaneously, the entropy can be obtained from the joint probability distribution of the random variables. For instance, if two random variables X and Y take values from set $\mathcal{X} = \{x_1, x_2, \cdots, x_M\}$ and $\mathcal{Y} = \{y_1, y_2, \cdots, y_N\}$ respectively, and their joint probability distribution is defined as $P(X, Y)$, therefore the joint entropy of X and Y is:

$$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P(x,y)\ log_2\ P(x,y) \tag{4.8}$$

and the average entropy of Y conditional on the value of X, averaged over all values in set $\mathcal{X}$ is:

$$H(Y|X) = -\sum_{x\in\mathcal{X}} P(x)\ H(Y|X=x) \tag{4.9}$$

Based on these two entropies, the chain rule for joint entropy defines that the total *predictability* about the value of X and Y is equal to the *predictability* about X combined with *predictability* of Y (average) given X. Therefore joint entropy of X and Y can be rewritten as:

$$H(X,Y) = H(X) + H(Y|X) \tag{4.10}$$

Based on the joint entropy equation (4.8), mutual information $I(X;Y)$ between X and Y can be defined as the relative entropy between joint distribution $P(X,Y)$ and product distribution of X and Y:

$$I(X;Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P(x,y)\log_2\ \frac{P(x,y)}{P(x)P(y)} \tag{4.11}$$

which means that how much of entropy X is reduced when *predictability* Y is increased:

$$I(X;Y) = H(X) + H(X|Y) \tag{4.12}$$

If X is perfectly informative (when information from Y is not required), then the mutual information between X and Y is simply the entropy of X:

$$I(X;X) = H(X) \tag{4.13}$$

In this thesis, we use MIA as a tool to measure the *informativeness* of sensor inputs towards prediction tasks. MIA is a sensible choice of exploring data, since it could capture the non-linearity properties in our dataset. For instance, in Chapter 7 we used MIA to estimate the usefulness of a particular $\mathbf{BoD}$ sensor to determine the correct user identity.

4.3 Classification

In classification, the goal is to predict a category. For instance given a set of n training points, $\mathbf{x}_1, \ldots, \mathbf{x}_n$ with class labels $t_n = \{-1, +1\}$ which describes which class object n belongs to. The goal of classification is therefore to assign $\mathbf{x}$ to either class -1 or $+1$. When there are only two classes, this is called *two-class* or *binomial* classification. When there are more than two classes involved, this problem is called *multi-class* classification. We only deal with two-class classification problem in this thesis.

4.3.1 Support Vector Machines

Support Vector Machines (SVM) was developed by in the 1990's and is considered as one of the best general purpose classifiers today. SVM is chosen as the classifier for classification tasks in this thesis because it has the ability to handle high-dimensional data. Another reason is due to fact that SVM can separate non-linear data through the use of kernels which gives flexibility in the decision boundaries. Lastly, SVM is perhaps one of the popular classifiers used in HCI, therefore by using SVM it is easier to compare our results with the results from other work in the literature.

Fundamentals of SVM

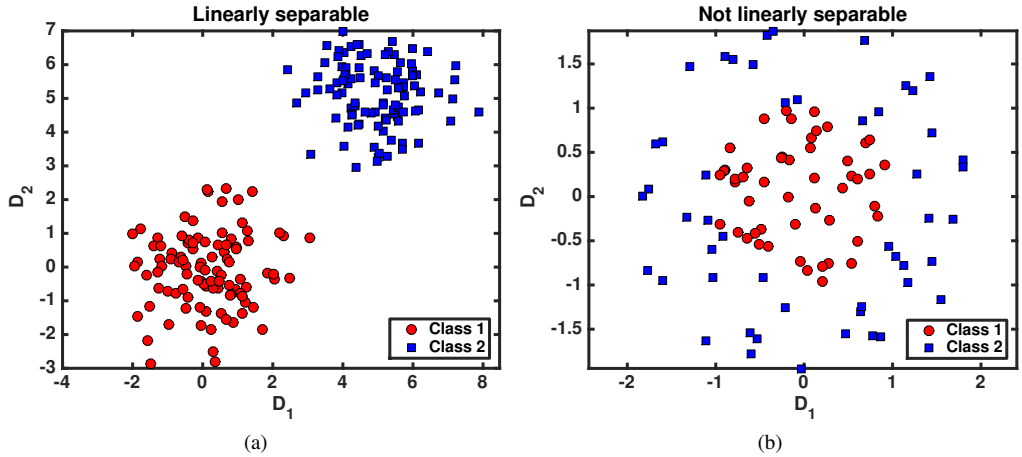


Figure 4.3: Example of two-dimensional (a) linear (b) not linearly separable datasets.

In classification, we are typically presented with a dataset $\mathbf{X}$, composed of N elements $\mathbf{x}_n$ in D dimensions in either class $t = 1$ or $t = -1$. Therefore our training data can be represented in the following form:

$$\mathbf{X} = \{\mathbf{x}_n, t_n\} \text{ where } n = 1 \ldots N,\ t_n \in \{1, -1\},\ \mathbf{x} \in \Re^D$$

As shown in Figure 4.3 (a), the two classes (denoted by different marker and colour combination) in two-dimensional dataset $\mathbf{X}$ can be separated by a straight decision boundary line (or a hyperplane if dimension $D > 2$). Figure 4.3 (b) shows an example of when dataset $\mathbf{X}$ is not linearly separable, where drawing a straight line is not possible. There are infinite possible straight lines (hyperplanes) that can be drawn on the graph in Figure 4.3 (a) to separate the two classes. Therefore the goal here is not only to find the best hyperplane that correctly separates but also maximises the orthogonal distance between the closest data points from both classes to the hyperplane. In other words, the objective of SVM is to orientate the separating hyperplane as far as possible from the closest data points from both classes.

In SVM the decision line or hyperplane that separates the classes is formally defined as:

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{4.14}$$

where $\mathbf{w}$ is normal to the hyperplane, b is the bias term and $\frac{b}{\|\mathbf{w}\|}$ is the orthogonal distance from the hyperplane to the origin. From this equation, the objective of SVM is to choose variables $\mathbf{w}$ and b so that the training data $\mathbf{X}$ can separated by a hyperplane based on the their class labels as follows:

$$\mathbf{w} \cdot \mathbf{x}_n + b \geq +1 \quad \text{for } t_n = +1 \tag{4.15}$$

$$\mathbf{w} \cdot \mathbf{x}_n + b \leq -1 \quad \text{for } t_n = -1 \tag{4.16}$$

$$t_n(\mathbf{w} \cdot \mathbf{x}_n + b) - 1 \geq 0, \text{ for all } n \tag{4.17}$$

As shown in Figure 4.4, the distance from H_1 and H_2 to the separating hyperplane H_0 is defined as γ_1 and γ_2 respectively, where $\gamma_1 = \gamma_2$. The combination of these two is known as *margin*. In order to determine the optimal orientation of the hyperplane, SVM needs to maximise the margin. Data points that fall onto H_1 and H_2 are known as *support vectors* (circled data points) which can be be formally defined as:

$$\mathbf{w} \cdot \mathbf{x}_n + b = 1 \quad \text{for } H_1 \tag{4.18}$$

$$\mathbf{w} \cdot \mathbf{x}_n + b = -1 \quad \text{for } H_2 \tag{4.19}$$

SVM margin $(d_i + d_2)$ can be defined as $\frac{1}{\|\mathbf{w}\|}$ and to maximise this margin requires applying the constraint in equation (4.17):

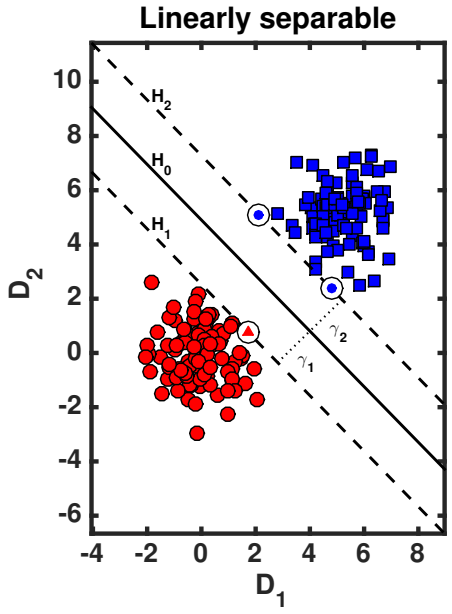


Figure 4.4: Linearly separable SVM classification example. H_0 is the decision hyperplane whereas circled data points correspond to support vectors.

$$\max \|\mathbf{w}\| \qquad \text{subject to} \qquad t_n(\mathbf{w}\cdot\mathbf{x}_n + b) - 1 \geq 0, \text{ for all } n \tag{4.20}$$

This constraint optimisation is equivalent to minimising $\frac{1}{2}\|\mathbf{w}\|^2$ which makes it easier to solve via quadratic programming (QP), therefore equation (4.20) is equivalent in determining:

$$\min \frac{1}{2}\|\mathbf{w}\|^2 \qquad \text{such that} \qquad t_n(\mathbf{w}\cdot\mathbf{x}_n + b) - 1 \geq 0, \text{ for all } n \tag{4.21}$$

To solve equation (4.21) requires allocating a set of Lagrange multipliers α, where $\alpha_n \geq 0,$ for all n which gives the new objective function N_P:

$$\begin{aligned} N_P &\equiv \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} \alpha_n (t_n(\mathbf{w}\cdot\mathbf{x}_n + b) - 1) \\ &\equiv \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} \alpha_n t_n(\mathbf{w}\cdot\mathbf{x}_n + b) + \sum_{n=1}^{N} \alpha_n \end{aligned} \tag{4.22}$$

The goal of the optimisation is to find the parameters $\mathbf{w}$ and b that minimises, and α that maximises equation (4.22). This is achieved by partially differentiating N_P with respect to $\mathbf{w}$ and b and setting these derivatives to zero:

$$\frac{\partial N_P}{\partial \mathbf{w}} = 0 \implies \mathbf{w} = \sum_{n=1}^{N} \alpha_n t_n \mathbf{x}_n \tag{4.23}$$

$$\frac{\partial N_P}{\partial b} = 0 \implies \sum_{n=1}^{N} \alpha_n t_n = 0 \tag{4.24}$$

Substituting equations (4.23) and (4.24) in equation (4.22) gives a new objective function definition which must be maximised with respect to α_n rather that $\mathbf{w}$:

$$N_D \equiv \sum_{m=1}^{N} \alpha_n - \frac{1}{2} \sum_{n,m=1}^{N} \alpha_m \alpha_n t_m t_n \mathbf{x}_m \cdot \mathbf{x}_n \tag{4.25}$$

The new objective function N_D is known as the *Dual* form of the *Primary* N_P which needs to be maximised subject to the following constraints:

$$\alpha_n \geq 0, \text{ for all } n, \quad \sum_{n=1}^{N} \alpha_n t_n = 0$$

To solve the optimisation problem, we use `libSVM` solver which is based on Sequential Minimisation Optimisation (SMO) [11] to compute the optimal α_n and substituting this in equation (4.23) gives $\mathbf{w}$. Any $\mathbf{x}$ that satisfies equation (4.24) is a support vector $\mathbf{x}_s$ which can be defined as:

$$t_s(\mathbf{x}_s \cdot \mathbf{w} + b) = 1$$

and substituting in equation (4.23) gives:

$$t_s(\mathbf{x}_s \cdot \sum_{m \in S} \alpha_m t_m \mathbf{x}_m + b) = 1$$

where S corresponds to the set of support vector indices, which is determined by finding indices n where $\alpha_n > 0$. By multiplying both sides of the equation with t_s and define $t_s^2 = 1$ from equations (4.18) and (4.19) gives:

$$\begin{aligned} t_s^2(\mathbf{x}_s \cdot \sum_{m \in S} \alpha_m t_m \mathbf{x}_m + b) &= t_s \\ b = t_s - \sum_{m \in S} \alpha_m t_m \mathbf{x}_m \cdot \mathbf{x}_s & \end{aligned} \tag{4.26}$$

Prediction on new data $\mathbf{x}_{new}$ is made by evaluating the following equation:

$$
\begin{aligned}
t_{new} &= \text{sgn}(\mathbf{w} \cdot \mathbf{x}_{new} + b) \\
&= \text{sgn}\left(\sum_{n=1}^{N} \alpha_n t_n \mathbf{x}_n \cdot \mathbf{x}_{new} + b\right)
\end{aligned} \tag{4.27}
$$

Soft Margin

As shown in Figure 4.4, dataset $\mathbf{X}$ is linearly separable by the hyperplane. This is a *hard margin* SVM. However in cases that are not fully linearly separable, the constraints set in equation (4.15) and equation (4.16) are relaxed to allow misclassification of training points. This is achieved using *soft margin* SVM, where a positive slack variable ξ_n, $n = 1, 2, \cdots, N$ is used, thus the constraint in equation (4.17) now becomes:

$$
t_n(\mathbf{w} \cdot \mathbf{x}_n + b) \geq 1 - \xi_n \quad \text{where } \xi_n \geq 0, \quad \text{for all } n \tag{4.28}
$$

The idea of *soft margin* SVM is that it applies a penalty to the training points that sit on the incorrect side of the decision plane and increases it as the distance from the hyperplane (margin) increases. Therefore by adapting objective function in equation (4.21) the new objective function is:

$$
\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N} \xi_n \quad \text{subject to} \quad t_n(\mathbf{w} \cdot \mathbf{x}_n + b) - 1 + \xi_n \geq 0, \text{ for all } n \tag{4.29}
$$

where parameter C controls the trade-off between slack variable penalty and size of the margin. In other words, C determines to what extent that the misclassifications need to be avoided. A large C value denotes that smaller margin is preferred and vice versa if a small C value is used.

Following the form in equation (4.25), the task now is to find the maximum of the following quadratic programming problem:

$$
\max_{\alpha} \sum_{m=1}^{N} \alpha_n - \frac{1}{2} \sum_{n,m=1}^{N} \alpha_m \alpha_n t_m t_n \mathbf{x}_m \cdot \mathbf{x}_n
$$

$$
\text{subject to } \; 0 \leq C \leq \xi_n \; \text{ for all } \; n \; \text{ and } \; \sum_{n=1}^{N} \alpha_n t_n = 0
$$

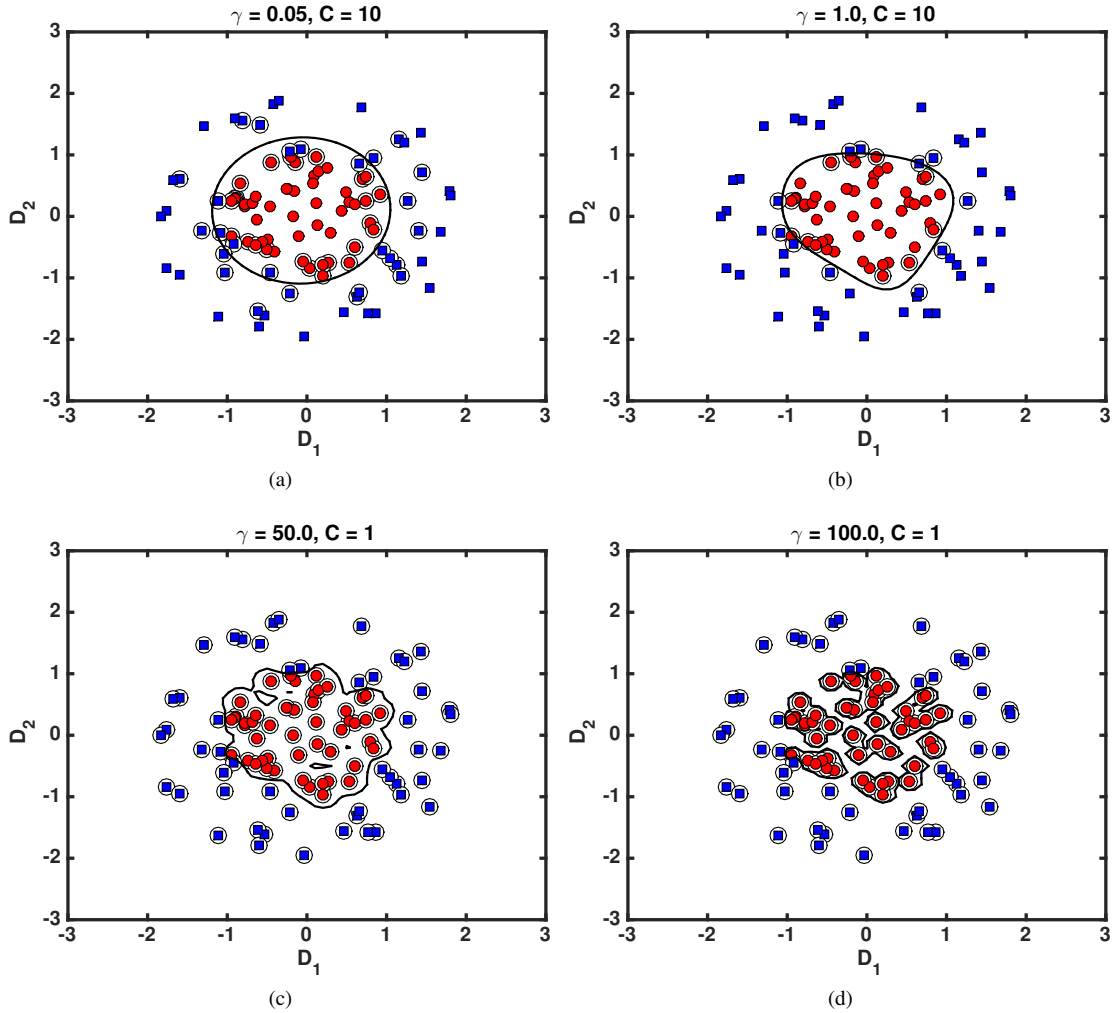


Figure 4.5: Decision boundary and support vectors for sample dataset in Figure 4.3 (b) as the γ parameter for Gaussian RBF kernel is increased.

where the upper bound of α_n is now set to C which implies the upper bound of any training point can have to influence the decision hyperplane.

Kernel Method

Figure 4.3 (b) shows a dataset where linear decision boundary would not be appropriate. To handle non-linearly separable cases, SVM uses *kernel trick*. In all of our work, we approach our classification problems using this method.

Classification using kernel method involves transformation of data from into a new space $\phi(\mathbf{x}_n)$. SVM operates exclusively on the inner products of the data $(\mathbf{x}_n^\intercal \mathbf{x}_m)$ therefore after the transformation, the inner products in the new space needs to be computed: $\phi(\mathbf{x}_n)^\intercal \phi(\mathbf{x}_m)$. This transformation is achieved using positive definite kernel function $k(\mathbf{x}_n, \mathbf{x}_m) = \phi(\mathbf{x}_n)^\intercal \phi(\mathbf{x}_m)$ [**?**]. The function used for transformation is called as *kernel function*.

In this thesis, we use Gaussian kernel function in all of our work which is defined as follows:

$$k(\mathbf{x}_n, \mathbf{x}_m) = \exp\{-\gamma(\mathbf{x}_n - \mathbf{x}_m)^\intercal(\mathbf{x}_n - \mathbf{x}_m)\} \tag{4.30}$$

where γ controls the complexity of the function. Gaussian kernel also is referred as Radial Basis Function (RBF). Therefore by incorporating kernel function (4.30) into equation (4.25), the objective function (soft margin) now becomes:

$$\max_{\alpha} \sum_{m=1}^{N} \alpha_n - \frac{1}{2} \sum_{n,m=1}^{N} \alpha_m \alpha_n t_m t_n k(\mathbf{x}_m, \mathbf{x}_n)$$

$$\text{subject to} \quad 0 \leq C \leq \xi_n \quad \text{for all} \quad n \quad \text{and} \quad \sum_{n=1}^{N} \alpha_n t_n = 0$$

and the kernelised decision function version of equation (4.27) for prediction:

$$t_{new} = \text{sgn}\left(\sum_{n=1}^{N} \alpha_n t_n k(\mathbf{x}_n, \mathbf{x}_{new}) + b\right)$$

The combination use of soft margin SVM with kernel function requires the selection of optimal C and γ parameters simultaneously. To select these parameters, we use cross-validation method which is explained in Section 4.5.

As shown in Figure 4.5, the SVM is able to separate the two classes (blue and red) after applying transformation on the data using RBF kernel function. Classification in high dimension space can be imagined as 'raising' the red circles and then separate them from the blue squares with a plane (hyperplane). The gamma parameter γ, controls the shape of the peaks of this 'raise' where a small gamma value correspond to a pointed bump, while a large gamma corresponds to a broader bump in the higher dimension.

Using `libSVM` [17] implementation which uses SMO-type algorithm, an RBF SVM has training computational complexity of $O(dn^2)$, where d is the number of features and n is the number of training/testing examples. On the other hand, the time complexity for prediction is $O(dn)$. This makes it possible to implement in a mobile phone, where the models can be trained offline (i.e PC) and use them in a mobile phone for real-time prediction.

As mentioned earier, the main advantages of SVM is the ability to handle non-linear data, highly scalable and works well with high-dimensional data. However, there is no method for determining hyperparameters without performing cross validation.

Composite Kernel

Besides building classifiers using kernel from each data type, we also are interested in building classifiers using composite kernel from both data types. This is achieved by constructing separate kernel for each data type and combine both kernels additively by applying kernel weight a to each kernel. The composite kernel K is given by:

$$K = aK_i + (1 - a)K_j$$

where $0 \leq a \leq 1$ with 0 representing only the K_j kernel and 1 representing only the K_i kernel. Although it is possible to combine both data types into same kernel, however this would make it hard to assess the relative contribution each data type on the classification performance. To select parameter a we fix the optimal kernel γ parameter for each data type and use cross-validation method to choose the optimal composite kernel weight a. We used this method in Chapter 7 and 6 when combining both accelerometer and back-of-device kernels.

4.3.2 Random Forest

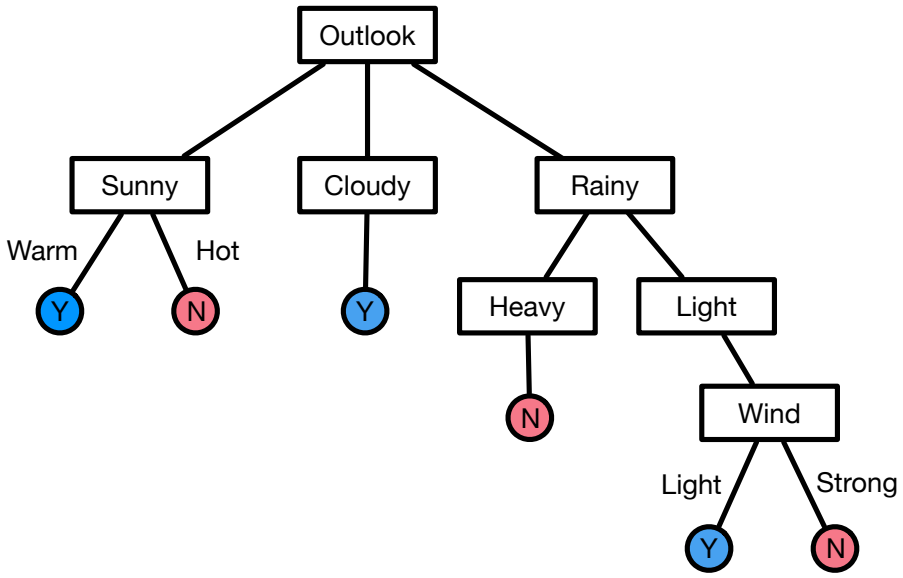


Figure 4.6: This decision tree classifies whether the weather is suitable for outdoor cycling. An example is classified by sorting through the box (leaf node) based on the variables and returns the associated class **Y**es or **N**o.

A decision tree is a hierarchy of rules used to classify data into classes or categories. An input is entered at the top and as it traverses down the tree the data gets divided into smaller and smaller sets. Consider the example in Figure 4.6 where the tree is used to decide whether or

not to go out cycling based on the weather conditions. For instance, if the outlook is sunny and the temperature is less than or equal to 30 degree, then it is probably good to go out cycling. However, decision trees are prone to overfitting and therefore does not generalise well.

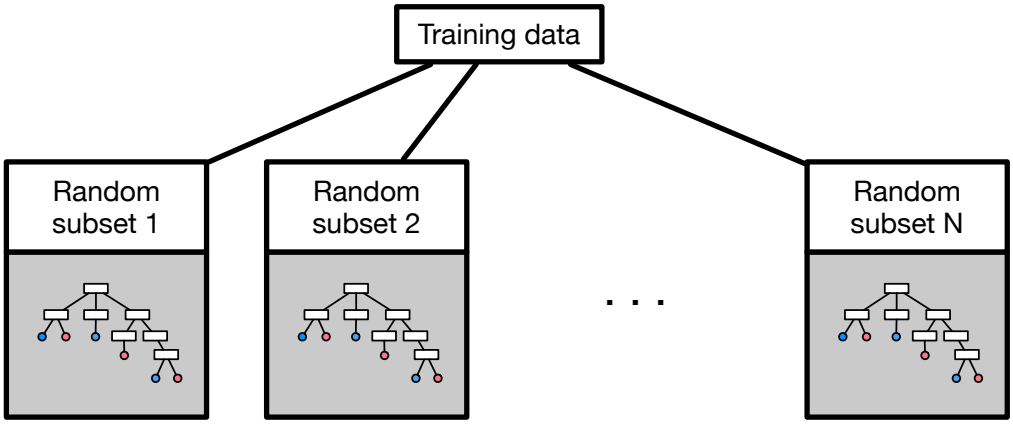


Figure 4.7: A random forest is formed by combining trees constructed from N random subsets of the training data.

Random Forests (RF) on the other hand is an extension of decision tree classifier based on an ensemble learning method. Developed by Breiman [12], this method constructs multiple decision trees (predictors) at training time to form a classifier. It based on the principle that a set 'weak predictors' can be grouped together to form a 'stronger predictor'. In order to so, RF adds some randomness when constructing the trees. As shown in Figure 4.7, this is achieved by combining trees constructed from random subsets of the training data through bootstrap aggregating. This stabilises the predictions thus reducing variance and overfitting. To make a prediction, RF averages the predictions from all subsets (trees) in the forest.

Just like decision trees, RF have low bias and by using more decision trees to build RF, it minimises the variance (discussion on tradeoff between bias and variance is given in Section 4.5). Moreover unlike SVM, RF has only one parameter to tune which is the number of trees and therefore faster to construct. Due to this simplicity, RF is a very popular classifier choice in many applications. We use RF in Chapter 6 to classify hand grip data.

4.4 Regression

Similar to classification, regression is concerned with inferring the values of one (or more) response variables, for a given set of predictor variables that have not yet been observed. In this thesis, we used regression in Chapter 5 to predict pair of touch coordinates (pixels) on touchscreen.

4.4.1 Gaussian Processes

In contrast to standard regression techniques which assume that the input data has some underlying parametric function (e.g. $f(x) = mx+c$), Gaussian Processes (GPs) on the other hand approaches the problem differently. GPs are a non-parametric regression technique because instead of trying to fit parameters of the selected basis functions (e.g. linear, cubic), GPs define how all data points are correlated via a specific covarince function typically by taking the features, $\mathbf{x}$, as input.

Using GPs, the goal is to represent a function $f(x_n)$ based on a set of training examples so that prediction can be made for unseen x by exploiting the correlation between the traning points and test. The function value of x_n, $f(x_n)$ is it self a random variable, f_n. Here the function output for $f(x_n)$ is f_n. The main assumption made by the GP is that the collection of function outputs f_n for all possible subsets subset of $\mathbf{x}$ input values x_n are consistently jointly Gaussian distributed.

GP is fully specified by a mean $\mu(\mathbf{x}_n)$ and covariance functions $k(\mathbf{x}_n, \mathbf{x}_m)$ which means that function f is distributed as a GP with mean function μ and covariance function k. The mean and covariance functions are used to compute the elements of the mean vector and covariance matrix that define the Gaussian distribution. In most applications there is no prior knowledge about the mean function μ of a given GP, it is commonly assumed that the mean function to be zero: $\mu(\mathbf{x}_n) = 0$ [10].

By using zero mean function means that the entire GP is defined by the covariance function. The covariance function $k(\mathbf{x}_n, \mathbf{x}_m)$ can be any general function that takes two arguments, such that $k(\mathbf{x}_n, \mathbf{x}_m)$ generates a positive definite covariance matrix $\mathbf{K}$. A popular choice of covariance function is squared exponential:

$$k(\mathbf{x}_n, \mathbf{x}_m) = \sigma \exp\left(-\gamma ||\mathbf{x}_n - \mathbf{x}_m||^2\right). \tag{4.31}$$

where $\sigma > 0$ is the signal variance and $\gamma > 0$ is the lengthscale. This covariance function also is referred as Radial Basis Function. From equation (4.31), it shows that the covariance between the inputs is really close to one $(k(\mathbf{x}_n, \mathbf{x}_m) \approx 1)$ if they are close to each other $(\mathbf{x}_n \approx \mathbf{x}_m)$ and decreases exponentially if the distance increases. Lengthscale hyperparameter γ controls the smoothness of a function. Small lengthscale value means that function values can change quickly (rough) or slowly (smooth) if large value lengthscale is used. Signal variance hyperparameter σ is a scaling factor that determines the variation of function from their mean. Small value corresponds to the functions that stay close to the mean where large value allows more variation to the functions.

The choice of γ and σ parameters can significantly influence the performance of the GPs. In this thesis, all GPs hyperparameters are optimised by maximising the marginal likelihood of

a GP. The output of the GP is a Gaussian distribution of functions expressed in terms of mean and variance. The mean value can be interpreted as the most likely output of the function while variance is the measure of confidence.

Prediction with Gaussian Processes

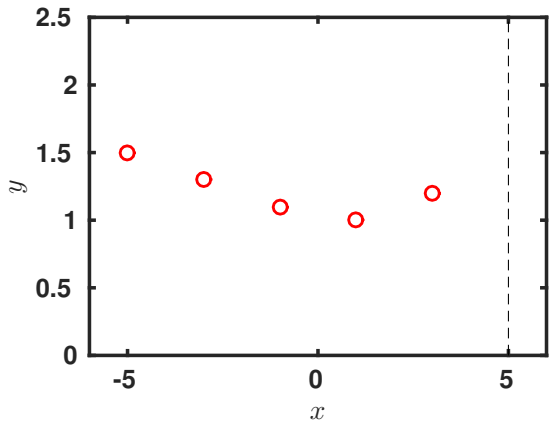


Figure 4.8: Some observation data (red markers) with the prediction point (dashed line).

Prediction using GPs is no different with typical prediction problem in standard regressions, that is given a set of N observations $\mathbf{y} = [y_1, y_2, \cdots, y_N]^\intercal$ at certain time points $[x_1, x_2, \cdots, x_N]^\intercal$ such that $y_n = f(x_n)$ what is the best estimate of y^* at a new time point x_{N+1} ? To illustrate this, Figure 4.8 shows some training data (red circles) and the x values where the prediction going to take place (dashed lines). $*$ is used to differentiate between data at the existing points and data at the points where the GPs will be predicting.

Formally the training set can be organised into function input vector $\mathbf{x} = [x_1, x_2, \cdots, x_N]^\intercal$ and function output vector $\mathbf{y} = [y_1, y_2, \cdots, y_N]^\intercal$ while the prediction set can be organised into function input vector $\mathbf{x}^* = [x_1^*, x_2^*, \cdots, x_L^*]^\intercal$ and function output vector $\mathbf{y}^* = [y_1^*, y_2^*, \cdots, y_L^*]^\intercal$ respectively, where $y_n = f(x_n)$, $y_n^* = f(x_n^*)$ and L corresponds to the number of testing points in the prediction set.

A GP model implies that the function values at all input points (training and test) comes from the GP, thus the joint distribution of the finite vectors $\hat{\mathbf{y}} = [\mathbf{y}, \mathbf{y}^*]^\intercal$ is given by a Gaussian distribution allowing us to easily condition on the traning observation/function values.

In order to perform GP regression, the covariance matrices of all possible combinations must be calculated using covariance function in Equation 4.31. A covariance matrix for training data:

$$\mathbf{K} = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_N) \\ k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_1, x_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_N, x_1) & k(x_N, x_2) & \cdots & k(x_N, x_N) \end{bmatrix} \tag{4.32}$$

a covariance matrix for testing data:

$$\mathbf{K}^* = \begin{bmatrix} k(x_1^*, x_1^*) & k(x_1^*, x_2^*) & \cdots & k(x_1^*, x_L^*) \\ k(x_2^*, x_1^*) & k(x_2^*, x_2^*) & \cdots & k(x_1^*, x_L^*) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_L^*, x_1^*) & k(x_L^*, x_2^*) & \cdots & k(x_1^*, x_L^L) \end{bmatrix} \tag{4.33}$$

and a $N \times L$ cross–covariance matrix:

$$\mathbf{R} = \begin{bmatrix} k(x_1^*, x_1^*) & k(x_1^*, x_2^*) & \cdots & k(x_1^*, x_L^*) \\ k(x_2^*, x_1^*) & k(x_2^*, x_2^*) & \cdots & k(x_1^*, x_L^*) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_N, x_1^*) & k(x_N, x_2^*) & \cdots & k(x_N, x_L^*) \end{bmatrix} \tag{4.34}$$

As mentioned earlier the key assumption in GP modelling is that the data follows Gaussian density. Therefore assuming zero mean, the GP model can be written as follows:

$$\begin{bmatrix} \mathbf{y} \\ y^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{R} \\ \mathbf{R}^\intercal & \mathbf{K}^* \end{bmatrix}\right). \tag{4.35}$$

This model tells that all elements (function values at the training and prediction points) are covary. The goal of prediction consists in estimating the mean value and the variance of y^*. Given the model in Equation 4.35 the first step now is to determine the conditional probability $p(y^*|\mathbf{y})$, that is given $\mathbf{y}$, how likely is a certain prediction for y^*?. This probability distribution is Gaussian and can be written as follows:

$$p(y^*|\mathbf{y}) = \mathcal{N}(\mathbf{R}^\intercal \mathbf{K}^{-1}\mathbf{y}, \mathbf{K}^* - \mathbf{R}^\intercal \mathbf{K}^{-1}\mathbf{R}). \tag{4.36}$$

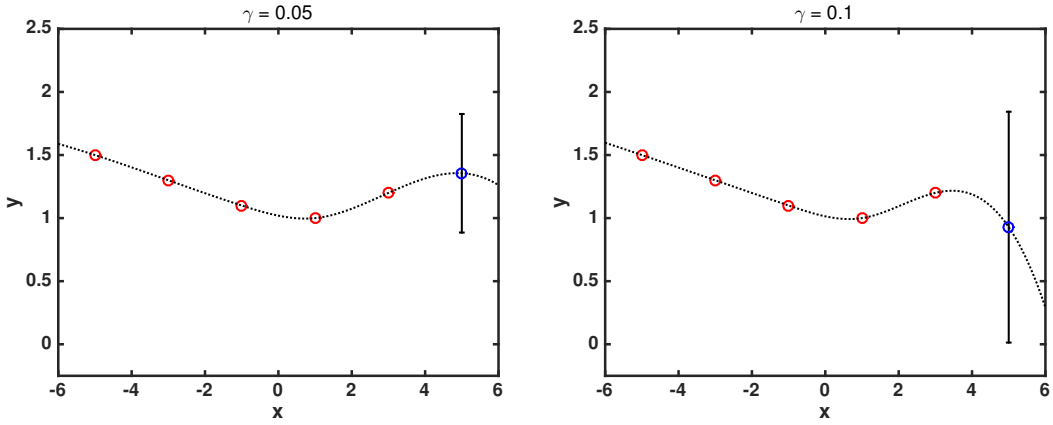
The best estimate for y^* therefore is the mean of this distribution:

$$\bar{y^*} = \mathbf{R}^\intercal \mathbf{K}^{-1}\mathbf{y}. \tag{4.37}$$

and the variance captures the certainty of this estimation:

$$\mathrm{Var}(y^*) = \mathbf{K}^* - \mathbf{R}^\intercal \mathbf{K}^{-1} \mathbf{R}. \tag{4.38}$$

Figure 4.9 (a) shows the distribution of y^* at x^* with blue circle represents the mean value and error bar showing plus and minus one standard deviation. Figure 4.9 (b) shows the distribution of predictions when using larger γ hyperparameter value. From these two GP models, (a) is likely to be chosen due to low variance in its estimation on y^*.



(a) (b)

Figure 4.9: Prediction of y^* at x^* using two different lengthscale γ values.

Often the data are noisy (e.g. measurement errors) therefore each observation in $\mathbf{y}$ can be modelled as being affected by some independent Gaussian noise ϵ with variance σ^2:

$$t_n = f_n + \epsilon_n$$

where $p(\epsilon_n) = \mathcal{N}(0, \sigma^2)$, which means that t_n is the true function value with additive Gaussian noise. Conditioning $\mathbf{t}$ on $\mathbf{y}$ gives the following distribution:

$$p(\mathbf{t}|\, \mathbf{y}, \sigma^2) = \mathcal{N}(\mathbf{y}, \sigma^2 \mathbf{I}_N)^1 \tag{4.39}$$

In most cases, we are interested to predict $\mathbf{y}^*$ rather than the corrupted $\mathbf{t}^*$ at a set of input $\mathbf{x}^*$. To do so, we need to construct the joint density similar in Equation (4.35) by replacing with new prior covariance:

$$\begin{bmatrix} \mathbf{t} \\ y^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma^2 \mathbf{I}_N & \mathbf{R} \\ \mathbf{R}^\intercal & \mathbf{K}^* \end{bmatrix}\right). \tag{4.40}$$

Thus the mean distribution to best estimate y^*:

[1] $p(\mathbf{t}_n|\, \mathbf{y}_n = f_n)$

$$\bar{y^*} = \mathbf{R}^\intercal(\mathbf{K} + \sigma^2\mathbf{I}_N)^{-1}\mathbf{t}. \tag{4.41}$$

In contrast to SVM, a GP has training computational complexity of $O(n^3)$ while the complexity for prediction is $O(n)$, where n is the number of training/testing examples. In terms of implementation, it is possible to implement this on mobile phone, since only the prediction part is implemented on mobile phone whilst the GP models are trained offline (i.e PC).

The primary advantage of GP is the fact that it is probabilistic. In contrast to SVM, the probabilitistic nature of GP allows direct hyperparameter optimisation and model-selection without the need of time-consuming cross validation. However GP does not scale as well as SVMs. In other words, GP tends to be slower as more training points is used.

4.5 Performance Evaluation

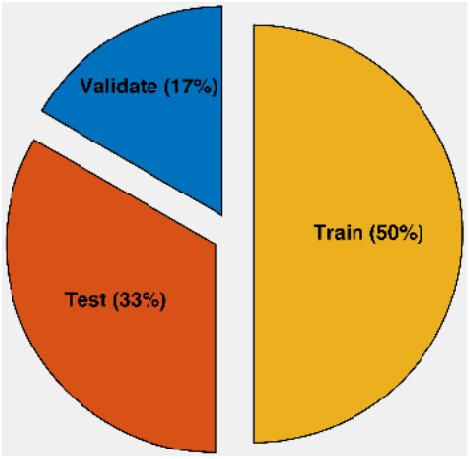


Figure 4.10: Example of data partitioning.

All classifiers must be evaluated prior to selecting the final model. This is to check for classifier's generalisation ability by estimating how well the classifier can classify new and unseen data. This involves partitioning the dataset into two independent sets: *training* and *testing*. Further split is made on the training set, normally in 75% (training) 25% (testing) ratio for *validation* purposes. Example of the partitioning is shown in Figure 4.10 where two thirds of the data is used for training where 25% of this is for model validation. The remaining is used for testing the model. The partition is to ensure that the final model does not lead to optimistically biased conclusion, that is when the model is trained and tested on the same data.

Bias and variance tradeoff in general is a model selection problem that emphasises on balancing between error due to bias and variance with respect to classification error:

$$classification\ error = error\ due\ bias + error\ due\ variance$$

Classification error due to bias is the error (on average) that is substantially different from the expected true value while classification error due to variance is the amount of error from one training set differs from the errors from all training sets. Both high bias and high variance model leads to poor classification results. In general, a model with high bias is too simple for the particular problem therefore will *underfit* data. On the other hand, a model with low bias and high variance is too complex thus will *overfit* data.

Underfitting can be evidenced by high error in both validation and training processes. Low error in training and high in validation on the other hand is a sign of overfitting. The error in validation is normally equal or slightly higher than the error in training for a good fit model. Whilst underfitting can be avoided by increasing classifier complexity (i.e linear to polynomial), overfitting on the hand can be estimated through cross-validation (CV).

In this thesis we used two variations of CV to evaluate our models. The first variation is called holdout method or 2-fold CV where data is randomly partitioned into two independent training and validation sets using a particular ratio, for instance two-third of data is used for training and one-third of data for evaluation (see Figure 4.10). To estimate overall performance of the model, the CV process is repeated for m times with different holdouts and average of the performance from m holdouts is taken.

The second variation of CV is called k-fold. In this variation, data is randomly divided into k disjoint (non-overlapping) subsamples of equal size and use the $k-1$ subsamples as training set and k subsample as evaluation set. The CV process is repeated for k times (the folds) to train and test the classifier. This gives k performance results which are then averaged.

CV can be used to select the best parameters for a classifier by estimating the generalisation error. When more than one parameters need to be optimised (i.e. SVM kernel parameters), *grid-search* method can be used. In this technique, the model is trained and evaluated using CV for different combination of parameters which lies on a grid of parameters. The best parameters for the particular model are then selected.

4.6 Performance Metrics

Depends on the problem, the performance of a classifier can be reported using different metrics. The following sections give an explanation each of the performance metric we used in this thesis. Although there are many other evaluation metrics exist, we opted the following as these are the most common metrics used in the literature.

4.6.1 Accuracy

Accuracy is the most simplistic approach to measure performance of a classifier. It measures how accurate the classifier classifies data in terms of number success (correct) classifications. Accuracy is expressed as ratio of correct classifications over all classifications. In binary classification, all correct classifications from both classes are called *true positives* and *true negatives* whereby all incorrect classifications from both classes are called *false positives* and *false negatives*. Therefore, accuracy can be defined as:

$$accuracy = \frac{true\ positives + true\ negatives}{true\ positives + false\ positives + true\ negatives + false\ negatives}$$

Accuracy assumes that instances from both positive classes are balanced, that is 50% of the instances are from either class. However in real setting (as in all of our experiments), the number of instances from each class is unlikely to be balanced. For example if 99% of the instances in a dataset belong to class A and 1% belong to class B, it is possible to achieve 0.99 accuracy by always predicting class A. The classifier in this case appears to produce high classification accuracy, however in reality it would not be able to predict any instances from class B.

One of the techniques to overcome this is to balance the class by resampling the dataset to even-up the classes. In our case, we always resample the majority class as many as the number of instances are there in the minority class by randomly selecting instances from the majority class. For instance, if class A (majority) and B (minority) contain 1000 and 250 training examples respectively, the training/test set will have all examples from class B and 250 examples selected randomly from class B.

4.6.2 Confusion Matrix

		Predicted class		
		Positive	Negative	Total
Actual class	Positive	TP	FN	P
	Negative	FP	TN	N
	Total	P'	N'	

Table 4.1: Binary classification confusion matrix.

As mentioned in Section 4.6.1, there are four possible outcomes from a binary classification. When positive and negative instances are classified correctly, they are known true positive (TP) and true negative (TN). Otherwise they are known as false positive (FP) and false

negative (FN). These four outcomes can be tabulated in a table of called confusion matrix (also known as contingency table) to show the relationship between these instances. Table 4.1 shows a confusion matrix for a binary classification. The rows represent the number of actual class labels and columns represent the number of predicted class labels in the test data. The diagonal of confusion matrix is the number of correct classifications while the opposite is the number of incorrect classifications or errors (confusion) made by a classifier.

As shown previously, accuracy is the proportion of correct classifications, therefore from the confusion matrix this is equivalent to:

$$accuracy = \frac{TP + TN}{P + N}$$

Besides accuracy, there are several additional metrics that can be calculated from confusion matrix:

- True positive rate (hit rate) is the proportion of positives classified correctly:

$$TP_{rate} = \frac{TP}{P}$$

- True negative rate is the proportion of negative instances classified correctly

$$TN_{rate} = \frac{TN}{N}$$

- False positive rate or false alarm/accept rate (FAR) is the proportion of incorrectly classified negative instances:

$$FP_{rate} = \frac{FP}{N}$$

- False negative rate or false reject rate (FRR) is the proportion of incorrectly classified positive instances:

$$FN_{rate} = \frac{FN}{P}$$

Each metric derived from confusion matrix is used to describe performance of a classifier in different contexts. For instance, in Chapter 7 we used confusion matrix to evaluate classifiers in terms of their false accept (FP_{rate}) and false reject (FN_{rate}) errors when predicting user identity.

4.6.3 Receiver Operating Characteristics Analysis

Receiving operating characteristics (ROC) analysis is a technique to evaluate performance of a classifier using a graph by depicting relative tradeoffs between false positive and true

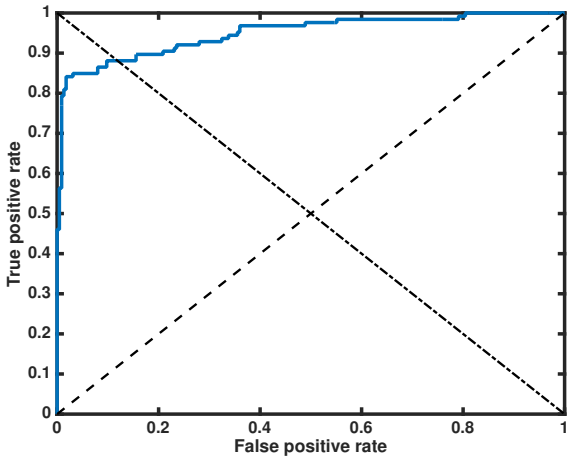


Figure 4.11: A ROC graph constructed from a binary classification outputs using SVM classifier. The diagonal dashed line ($y = x$) denotes the random performance while the intersection of the curve with opposite line ($y = 1 - x$) denotes the Error Equal Rate (EER).

positive rates of a classifier. It plots the cost (FP_{rate}) of classifying data at a particular performance (TP_{rate}). In other words, the goal of ROC analysis is to visualise the effect of TP_{rate} on FP_{rate}.

Consider a ROC graph for a SVM classifier in Figure 4.11. The x- and y-axis correspond to false positive rate and true positive rates of a binary classification. In SVM, an instance will be classified as positive if the *score* (signed distance from the decision hyperplane) is above a certain threshold (zero if using hard margin) or negative if otherwise. Comparing this with the actual class label would produce either true positive (correctly classified) or false positive (incorrectly classified) which is then used to compute TP_{rate} and FP_{rate}. These two values constitute a ROC point on the ROC graph ($FP_{rate} = x$, $TP_{rate} = y$). The blue curve (ROC curve) as shown in Figure 4.11 is the combination of all ROC points plotted together, sorted.

Bottom-left corner of the graph ($FP_{rate} = 0, TP_{rate} = 0$) denotes a threshold where the classifier never classifies anything to positive class whereby top-right corner ($FP_{rate} = 1, TP_{rate} = 1$) signifies a threshold where the classifier never classifies anything to negative class. A perfect classifier would have a ROC curve that hits top-left corner of the graph ($FP_{rate} = 0, TP_{rate} = 1$).

The diagonal dashed line ($y = x$) in Figure 4.11 corresponds to ROC points or ROC curve of a random classifier. For instance, if a random classifier guesses the positive class 70% correct of the time, it can be expected to classify 70% of the positive instances correct but at the same time increases the false positive rate to 70%. Therefore a random classifier is estimated to produce ROC points that sit on the diagonal line ($TP_{rate} = FP_{rate}$) on the ROC graph. The opposite of the random performance diagonal line ($y = 1 - x$) is Error Equal Rate (EER), which is the performance at the classification threshold that gives equal FR_{rate} and FP_{rate}. That is at threshold where number of misclassifying positives as negatives and

negatives as positives are at the same rate. The lower the EER value, the higher the accuracy of the classification.

In most cases, there will more than one classifiers to be compared during model selection process. In order to provide a single scalar value for comparison, area under the ROC curve (AUC) is used. AUC is a portion of the area of the unit square therefore it has a value between 0 and 1.0. Unlike accuracy, AUC is insensitive and more robust in class imbalance situations. One drawback of using AUC from ROC analyses is that it does not generalise to the multiclass setting. However this can be solved by turning multiclass setting into several binary classification problems. ROC analysis and AUC are used in Chapter 7 and Chapter 6 when evaluating SVM classifiers. This is due the fact that most of our classifications problems involve class imbalance situation.

4.6.4 Root-Mean-Squared Error

In contrast to classification, regression gives an estimation of the prediction (continuous output) rather than class membership. In order to quantify the error from regression problem, mean-squared error (MSE) can be used. MSE in general measures the error in terms of *residuals* of the fitted regression line to data points. MSE is computed by taking the mean of the squared distance between each test point and its corresponding predicted value. Formally this is:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

where n is the number of points, y_i is the predicted value obtained from regression and $\tilde{y}_i$ is the actual value. From MSE, root-mean-squared error (RMSE) is computed by simply taking the square root of the MSE:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2}$$

RMSE is thus the distance on average of a data point from the fitted regression line, measured along a vertical line ($x-$axis). RMSE has the same units as data plotted on the vertical axis ($y-$axis). We use RMSE when reporting performance of GP regression in Chapter 5.

4.7 Wilcoxon Sign-Rank Test

Statistical hypothesis testing is used to determine whether there is enough evidence to ”reject” a hypothesis about the results. For instance, prior to every experiment, we guess that sensor data does not have any predictive power. This guess forms a ”null” hypothesis. The experiments are performed to acquire sensor data to find evidence and potentially reject the ”null” hypothesis and accepting an ”alternative” hypothesis (i.e. sensor data has predictive power).

In this thesis we are mainly interested to investigate if there is statistically significant improvement in performance when two different classifiers are used (i.e single vs composite kernel or back–of-device vs accelerometer). When comparing performance results from two different classifiers, we use the non-parametric Wilcoxon Sign-Rank Test. By using this test, we assume that the data (prediction results) are not normally distributed.

The Wilcoxon Sign-Rank Test concerns the population median of the difference scores between two datasets. Thus the testing hypotheses (null H_0 and alternative H_1) need to be first stated:

H_0 : The median difference is zero

H_1 : The median difference is positive at $\alpha = 0.05$

4.7.1 Computation

The rank is computed by finding the paired difference: $d_i = x_i - y_i$ where x_i, y_i are the pairs from two predictions. Next the difference scores d_i are then ranked by ignoring the signs (i.e. assign rank 1 to the smallest, rank 2 to the next etc.). The sign of the rank is then assigned according to the actual sign of d_i. The test statistic for the Wilcoxon Signed-Rank W is computed by summing up the positive and negative signed d_is:

$$W^+ = \sum_{i=1}^{n} +d_i$$
$$W^- = \sum_{i=1}^{n} -d_i$$

From this summation, the null hypothesis H_0 is true if both W^+ and W^- are similar. Otherwise the alternative hypothesis is true if the number positive ranks are more than the negative

ranks ($W^+ > W^-$). To determine whether W supports null or alternative hypothesis by using tables of critical values, such that if W is less than or equal to the critical value, H_0 will be rejected or if W exceeded the critical value, H_0 will not be rejected. In this thesis, the whole computation has been simplified using $MATLAB$ `ranksum` function, with alpha significance level set at 0.05.

Chapter 5

Predicting Screen Touch Locations From Hand Grip and Motion Changes

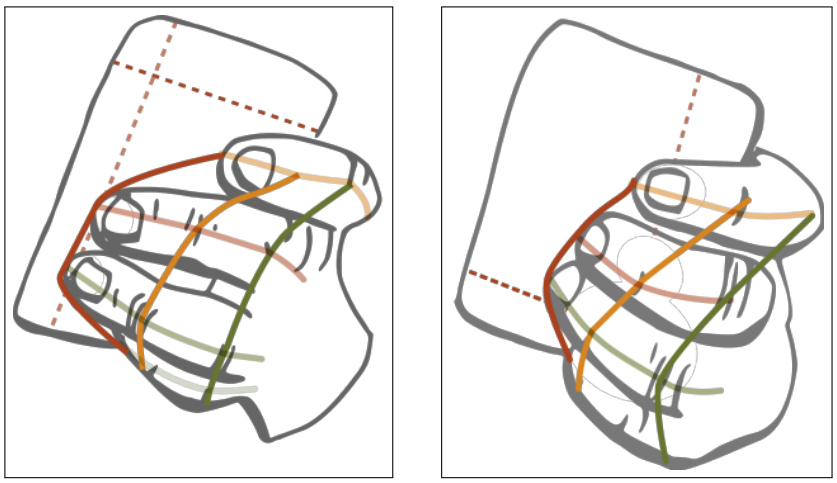
Part of the work presented in this chapter has won Honourable Mention award at the Conference on Human Factors in Computing Systems (CHI) 2014.

Summary: This chapter demonstrates the use of implicit sensing of hand grip and motion changes for touch location estimation on a touchscreen. Specifically, we show that using back–of–device (BoD) hand grip and accelerometer, touch locations on the touchscreen can be predicted before touch contact. We also show that time of touch contact can be predicted accurately using both input modalities. Each hand grip sensor is analysed to measure its relevancy in predicting touch locations and we show the locations of these sensors. At the end of this chapter, we demonstrate our technique as a touch location predictor implemented in the real product.

5.1 Introduction

Although touch has been a *de facto* mode of input for modern mobile devices, is still subject to a number of inherent challenges. Common issues such as input space constraints and accuracy have motivated researchers to propose complementary interaction techniques that not only enhances but also extends the capabilities of standard touch in mobile devices. This includes finger hover tracking for gesture-based interaction [151, 27], full-finger pose estimation to improve touch accuracy [106, 74], BoD [6] and around-device interaction (ADI) as an extension to touch input space [146, 67, 13].

These techniques employ sensors as an additional input modality to complement touch. In ADI for instance, common touch actions such as zooming, dragging and swiping can be *ex-*



(a) (b)

Figure 5.1: Grip changes as the the thumb targets different areas. Thumb target shown as dashed crosshairs. Notice the changes in the contours of the phalanges and finger tips (solid lines) as the thumb moves from top left (a) to bottom right (b) target.

pressed using motion gestures such as tilting and pivoting, thus freeing up the display from touch [51]. Researchers have also used hand capacitive profiles in order to determine hand postures, which brings extra interaction dimensions to conventional touch screens [112]. This is particularly useful in one hand operation as well as an effort to minimise the encumbrance effect – a situation where user's interaction is hindered with another task such as carrying typical objects (e.g. shopping bags, books) during interaction [91].

In this chapter we present an implicit sensing technique to enhance standard touch by predicting the location of screen touches from BoD hand grip and motion changes sensed using BoD capacitive sensors and an accelerometer respectively. This is based on the observation that when holding a phone single-handed, it is impossible to target with the thumb across the whole screen without changing the grip position and device pose, as illustrated in Figure 5.1.

In order to make inference of touch locations, we use Gaussian Process (GP) regression to find mappings between the two sensor inputs capacitive readings and accelerations, and touch locations. We use these mappings to estimate the touch contact location and touch contact time of fingers *before* they reach the touchscreen. By modelling these interesting dynamics, a number of enhancements can be potentially fitted into standard touch interactions. We outline three possible applications of this technique:

1. ***Content Pre-Loading***
   Off-device, cloud-based processing offers many opportunities for mobile interaction. One of the key issues that impeding the success of cloud-based processing in mobile environment is execution latency. This is evidenced by the existence of numerous techniques to accelerate execution of cloud-based applications to avoid latency [68,

45]. Despite fast connections, retrieving content over a wireless link may be prone to network latencies of 100–200ms and this can increase linearly with data size [78]. This level of delay is very noticeable, and can disrupt the rhythm of an interaction. For example, content in a webpage will be loaded when initiated *explicitly* by a user – tapping the associated button/link. However, given the fact that mobile phones have less computational power and poorer internet connections than other computers, the content may not appear immediately. Therefore through prediction of touch contacts, the interface components that the user was about to touch can be identified and content pre-loaded ahead of time. The accuracy of this touch prediction determines how much of content needs to be downloaded. For instance high accuracy denotes prediction of the exact touch location. This may involve downloading content for a single UI component. On the other hand, lower accuracy can be interpreted as estimation of larger touch location area that accommodates more than one UI components. This flexibility shows the feasibility of the content pre-loading.

2. ***Feedback Responsiveness Enhancement***
   Another potential application of touch contact prediction is enhancing auditory and tactile touch feedback. Existing feedback solutions can increase user confidence that touches have been registered but introduce latency of their own (e.g. audio buffering delay). A delay of 30 ms between touch and response is a common scenario. By predicting touch contact times, audio or vibro-tactile feedback can be queued to trigger exactly on the predicted touch contact time. This requires high predictive accuracy but only at the fraction of a second immediately preceding a touch. Whilst this approach is appealing, it needs to be balanced with the errors of predicting contact times of a touch when no touch happens.

3. ***Assistive User Interface***
   Although it is possible to interact with mobile phones single-handedly using a thumb, this however is limited to the display area reachable by the thumb of the hand that is holding the device. This reachable area is known as the thumb's functional area and is dependent on hand size and relative to display size [7]. In other words, the larger the display the smaller the functional area. Most mobile devices now come with display sizes larger than 5.0 inches (12.7 centimetres). It can be hard to reach targets located outside of thumb's functional area without changing the position of grip (e.g. top-left area with right hand thumb). By using prediction of touch contacts, the user interface can adapt to hand size by enlarging the predicted target area and assist users in reaching the target. This is particularly useful for users with below average adult hand size or when two-handed interaction is not possible (e.g. carrying a bag on the other hand).

5.1.1 Research Goals

The remainder of this chapter is organised around several research questions, which we answer via several experiments:

RQ1 ***Is there any observable relationship between touch position and hand grip / hand pose?***
This is the preliminary investigation that acts as the foundation of this study. In order to investigate and understand the relationship, we measure the correlation between touch locations and sensor input (hand grip and hand pose). This provides us an early insight about the relationships and structures that exist in the dataset which could help us to make decision on the touch prediction strategy. The result of this experiment is presented at the beginning of Section 5.3 and Section 5.3.1.

RQ2 ***Can we predict screen touch locations better than random using hand grip / hand pose?***
The main goal of this chapter is to measure to the extent to which touch locations can be predicted better than random from sensor input. We also compare prediction performance between hand grip and pose. We report prediction performance of our technique in Section 5.3.2.

RQ3 ***Can we predict time of touch contact better than random?***
Apart from predicting touch locations, we also are interested to see if contact time can be predicted from sensor input before the actual touch better than random. This is essentially the same process as RQ2 but using contact time as output. Results for contact time prediction is shown in Section 5.3.3.

RQ4 ***Do we need all sensors to predict screen touch locations better than random?***
Whilst predicting touch locations using BoD sensors is interesting on its own, we are intrigued to see if the prediction can be made better than random when the number of sensors is reduced, that us when using a subset of BoD sensors. Furthermore we also investigate the location of BoD sensors that can effectively capture grip patterns to predict touch location. We show results from this analysis in Section 5.3.4.

5.2 Experiment

We run an experiment to answer each of the research questions. In the first experiment, the goal is to investigate if an observable relationship exists between the sensor inputs $\mathbf{x}$ and output $\mathbf{y}$ (touch locations in (x, y) coordinates) (RQ1). Prior to evaluating the relationship, we first identify the structure in hand grip and hand pose data during the touch. We

compute Principal Component Analysis (PCA) on both M dimensional sensor inputs: BoD $\mathbf{x^c} = [\mathbf{x}_1^c, \mathbf{x}_2^c, \ldots, \mathbf{x}_M^c]$, where $M = 24$ and accelerometer $\mathbf{x^a} = [\mathbf{x}_1^a, \mathbf{x}_2^a, \mathbf{x}_3^a]$ (where $\mathbf{x}_1^a, \mathbf{x}_2^a$ and $\mathbf{x}_3^a$ correspond to the accelerations along 3-axis accelerometer) at touch contact time to investigate if structures such as clustering exist in the data. PCA is a linear transformation of data that allows us to inspect the differences and similarities of the dataset by viewing them from a different angle. Please refer the more in-depth description of PCA in Chapter 4.2.1.

5.2.1 Canonical Correlation Analysis

Next we measure the correlation between sensor inputs: BoD $\mathbf{x^c}$ and accelerometer $\mathbf{x^a}$ (from now onwards, we will use $\mathbf{x}$ when referring to either one of these sensor inputs) and touch location $\mathbf{y} = [x, y]$ (where x and y are the x– and y–coordinates in pixels). In other words, we would like to understand the relationship between pairs of variables: sensor values $\mathbf{x}$ and touch location $\mathbf{y}$. Given that $\mathbf{x}$ and $\mathbf{y}$ are two multi-dimensional variables, it requires a multivariate analysis technique to explain the relationship. For that purpose, we use Canonical Correlation Analysis (CCA) [55].

In this case, CCA computes two derived variables (canonical variates) $W_\mathbf{x}$ (weighted sum scores of sensor input variables) and $V_\mathbf{y}$ (weighted sum scores of touch location variables) and find the correlation between these two variables. The goal of CCA therefore is to find linear combination that maximises the correlation between $W_\mathbf{x}$ and $V_\mathbf{y}$. For further explanation on CCA, please refer to Section 4.2.2.

In short, CCA generalises multivariate correlation between $\mathbf{x}$ and $\mathbf{y}$ to bivariate correlation between $W_\mathbf{x}$ and $V_\mathbf{y}$. The relationship between these two pairs of variables can tell how well touch location $\mathbf{y}$ can be predicted from sensor input $\mathbf{x}$. The correlation between these two variables can be visualised on a graph by plotting $W_\mathbf{x}$ against $V_\mathbf{y}$. Results from CCA can provide an insight into the level of performance to be expected from the regression task. This is shown in Section 5.2.2.

5.2.2 Touch Location and Time of Touch Contact Predictions

The main goal of this study is to predict the intended touch location $\mathbf{y}$ using BoD (hand grip) and accelerometer (device pose) sensors inputs $\mathbf{x}$ (RQ2). Furthermore, we also are interested to find if time of touch contact $\mathbf{t}$ can be predicted from both sensor inputs $\mathbf{x}$ (RQ3). This is naturally viewed as a regression task, that is we are interested to find how well the set of sensor input $\mathbf{x}$ variables can predict touch locations $\mathbf{y}$ and time of touch contact $\mathbf{t}$. To make predictions, Gaussian Process Regression (GP) is used. GP is a flexible, non-parametric approach to regression analysis [98]. A detailed review of GP is given in Section 4.4.1.

In order to define a GP, we first define a prior mean regression function (in our case; $f(\mathbf{x}) = 0$) and a prior covariance function that defines the smoothness of the regression function. We train a separate, independent GP for each coordinate axis and time of touch contact. We use the squared exponential (SE) covariance function throughout this chapter in the following form:

$$k_{SE}(\mathbf{x}_n, \mathbf{x}_m) = \sigma \exp\left(-\gamma ||\mathbf{x}_n - \mathbf{x}_m||^2\right).$$

where the lengthscale hyperparameter γ determines the smoothness of the function whereby signal variance hyperparameter σ^2 controls the variation of functions from their mean. SE covariance function is chosen because it and can be integrated againsts a wide range of functions, making it a sensible choice for this task. We use the MATLAB implementation of GP from GP for Machine Learning (GPML) toolbox[1] for all the work described in this chapter.

To evaluate prediction errors, we use root-mean-square error (RMSE) in millimetres, that is the distance on average between the fitted regression line to the actual data points (detailed explanation of RMSE can be found in Section 4.6.4). As a basis for comparing prediction errors, we use RMSE of always of guessing the centre of the screen as a baseline to compare our predictions with random predictions. In addition to the baseline, we also compare our predictions with the standard touchscreen button size to put the performance into a realistic context. In other words, what would like to know what would the performance mean in the actual product. We use Android standard button size which is 48 density-independent pixel (dp)[2] or 7.62 mm according to N9 screen density.

5.2.3 Sensor Selection

Although we use 24 capacitive sensors at the back and sides of the prototype mobile phone, we argue that some of these sensors may have no predictive power at all, that is they may have no effect on the outcome of the prediction. Moreover, from a design point of view, the fewer sensors the better. This is based on the observation that when targeting the touch screen, some sensors remained untouched. Therefore our final goal in this study is to determine the relevancy of each $\mathbf{BoD}$ sensor in the prediction tasks and select only the relevant sensors to make predictions (RQ4). Similar to the technique in R3, we use a GP in this experiment.

To determine the relevance of the sensors we use a GP Automatic Relevance Determination (ARD) SE covariance function. Using this function, an individual hyperparameter length-scale σ_m is learnt for each sensor x_m. Here σ_m determines the relevancy of x_m. If σ_m has

[1] http://www.gaussianprocess.org/gpml/code/matlab/doc/
[2] http://www.google.com/design/spec/layout/metrics-keylines.html

a very large value, then the covariance will become almost independent of x_m thus making it irrelevant. ARD SE covariance function for M-dimensional input can be rewritten as follows:

$$k_{SE}(\mathbf{x}_n, \mathbf{x}_m) = \sigma \exp\left(-\sum_{d=1}^{M}(x_{nd} - x_{md})^2\right)$$

By removing irrelevant sensors, we can propose a minimum number of sensors required to predict screen touch locations efficiently. From a design point of view, the fewer sensors the better. Moreover, the experiment can reveal the location of the relevant sensors at the back of the prototype phone – valuable information that can be used by hardware designers.

5.2.4 User Study

We carried out a preliminary user study in order to answer all research questions we stated previously. The goal was to collect hand grip and pose examples from human subjects. To do so, we use a prototype mobile device based around a Nokia N9 smartphone. A brief technical specification of this prototype is given in Section 3.3. From the examples collected, we seek to measure the following:

1. Relationship between sensor input and touch locations.
2. Performance of sensor input as touch target and time of contact predictors.
3. Relevancy of sensors to the predictions.

Participants

We recruited 20 participants locally (12 males and 8 females) aged between $25 - 40$ (mean = 28.9, sd = 4.92) to participate in our experiment after gaining ethics approval on June 28th 2013. The small population size is chosen since this is a pilot study and has shown to be adequate to capture a plausible range of grip contacts [144, 24, 43]. All of our participants were current smartphone users and therefore had experience in operating touch screen device. We had 17 right and 3 left handed participants in total.

5.2.5 Data Acquisition

As mentioned earlier, we used the N9 prototype smartphone in this study. We wrote custom software in Python that ran on the phone to coordinate the acquisation of touch–grip/pose

Figure 5.2: During targeting experiment, participants were required to touch (tap) random target (crosshair) displayed on the screen using their thumb, single handed.

examples from both capacitive and accelerometer sensors. The sampling rate of both sensors was set at 100 Hz. As shown in Figure 5.2, the software displayed touch targets (crosshairs) on the screen that the participants had to touch using their thumb, single handed. Information pertaining to session and remaining targets was displayed at the top of the screen.

Every participant was given the same touch targets. We used unique random target coordinates (x, y) generated uniformly using Python built-in pseudo-random number generator library[3]. A half second delay was used between targets to encourage participants to return to a rest pose before the next target was shown. Audio feedback was given if the participant touches the target correctly. A correct touch requires a stable thumb contact within the minimum target area for at least 500 ms. We used a circle with 1 cm diameter (98.8 pixels) as target area in this experiment.

From each participant, we recorded timestamps (milliseconds), targets and touch locations (x, y coordinates in pixels) and BoD capacitive and accelerometer sensors readings into the phone's internal storage for subsequent off-line analysis.

In this experiment, participants were required to interact with the N9 prototype by touching/tapping the random targets shown on the touch screen using their only their thumb, single handed. We used 250 unique touch targets in this experiment. In order to ensure that we are not observing only a single grip pattern but a range of plausible grips/poses, we split the experiment into 5 different sessions in which we collected 50 touch actions using unique targets in each session, for each participant. Participants were required to switch their hand after each session, therefore each hand has an equal number of unique touches. There was a 5 minute break given between sessions to minimise the *repetition effects* – a condition where

[3] `https://docs.python.org/2/library/random.html`

participants perform targeting task without conscious effort as a result of repetitive movements over time [72]. In total, we recorded 500 touch actions in total from both hands, from all participants.

In this study we were not interested in how the participants initially picked up the phone, therefore the recordings began when the phone was held by the participants. Every participant was required to start the experiment with their dominant hand. Participants were seated on a chair the whole time during the experiment, facing a table where they can put the phone down when switching hand/session. Participants were debriefed prior to performing the experiment. In particular we explicitly mentioned that touch accuracy and speed were not being evaluated to ensure participants performed the experiment comfortably. They also were given a trial run to allow them to familiarise with the system. Overall, the experiment took between $15 - 20$ minutes (mean $= 17.50$, sd $= 1.28$) to complete.

5.2.6 Data Preprocessing

Prior to analysing our data, all targets and touch locations were rescaled such that they were in a unit square centred on the origin. The origin used by the phone's coordinate system was in the upper right corner of the screen in portrait mode. This rescaling was based on the device screen size, such that a target at the rightmost edge of the screen would have an x coordinate of 0.5, and a target at the bottom of the screen an y coordinate of 0.5. In the case where we predict the intended locations from BoD capacitive and accelerometer sensor values, this make the zero mean function of the GP correspond to the centre of the screen. Predicting the centre of the screen in the absence of data is a slightly more sensible assumption than predicting the upper right corner. We transform the coordinates back to a millimetre scale when computing prediction errors.

Feature Extraction

The raw signals from both BoD capacitive and accelerometer sensors were noisy and sampled at a slightly different rate. Therefore before proceeding to the analysis stage we first reduced the amount of data by downsampling the raw signals from both sensors to 50 Hz to equate the sample rate of both signals. Furthermore, the BoD capacitive sensors in our prototype also were occasionally affected by noise spikes. In order to remove these spikes, we ran a simple length 3 median filter over the downsampled signals.

The main goal of this study is to estimate touch locations using sensor values. In particular, we are interested to find how well the GP can learn from the sensor values to estimate touch locations, by varying the time before the finger touches the screen. To investigate this, we divided the timeseries into blocks of 100 ms interval up to 500 ms before touch contact. 0 ms

denotes the point of touch. For example, a 100 ms block = data from 0 ms – 100 ms before touch contact, a 200 ms block = data from 0 ms – 200 ms before touch contact and so on. We then extracted the mean from each block from each sensor as a feature. This provides us with a fixed-length 24–dimensional BoD ($\mathbf{x}^c$) and 3–dimensional accelerometer ($\mathbf{x}^a$) feature vectors for each timeseries block. By taking the mean values, we were only observing static components of hand grip and pose. We then normalised the data by rescaling the values linearly into a new minimum–maximum range after observing the minimum and maximum values from the original data.

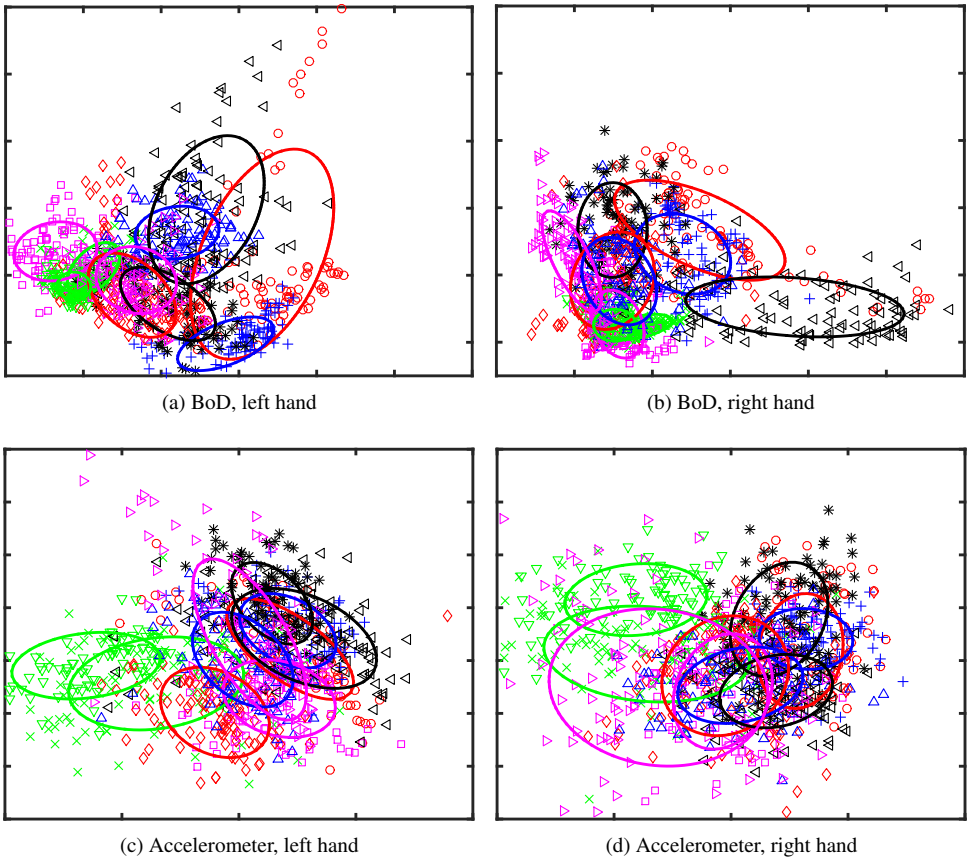
We also extended our feature vectors by adding the first and second derivates of both sensor (before normalisation), estimated using a Savitzky-Golay filter [113] (4^{th} order, 600 ms width), resulting in 74– and 11–dimensional feature vectors respectively. Savitzky–Golay filter is a low–pass filter that performs a least squares fitting on a set of points in the signal to a polynomial and take the central of the fitted polynomial as the new smoothed point. This filter was chosen over commonly used moving average filter due to its ability at retaining the shape of the original signal. The primary goal of this extension is to investigate if the derivatives of the original signal contain important information which has non-linear relationship with touch location. By extracting these additional information, it is hoped that the estimation errors can be improved.

5.3 Results

We use the non-parametric Wilcoxon Signed Rank Test with $p-$value threshold of 0.05 to test the significance level of the results. We begin the analysis by exploring the structure in our data by applying a linear transformation to the data using Principal Component Analysis (PCA). Prior to that, we standardised the feature data using standard scores (subtract sample mean from each observation then dividing by the sample standard deviation; mean=0, std=1) to ensure that each dimension is on the same scale.

Figure 5.3 shows the first (x-axis) and second (y-axis) principal components for each modality and from each hand for 10 participants selected randomly. Distinct participant clusters can be seen in both sensor plots for both hands, suggesting that every participant may have different ways of holding when interacting (targeting) with the mobile device. This also suggests that any predictive model based on hand grip and hand pose may need to be user–specific.

As shown in the Figure 5.4, the first and second principal components of BoD feature from the right hand account 27.5% of the variance of the original feature whereby for left hand, only 29.6% of the variance is accounted by the first and second principal components. Moreover, both plots also show that the variances decay slowly suggesting that the information



(a) BoD, left hand (b) BoD, right hand

(c) Accelerometer, left hand (d) Accelerometer, right hand

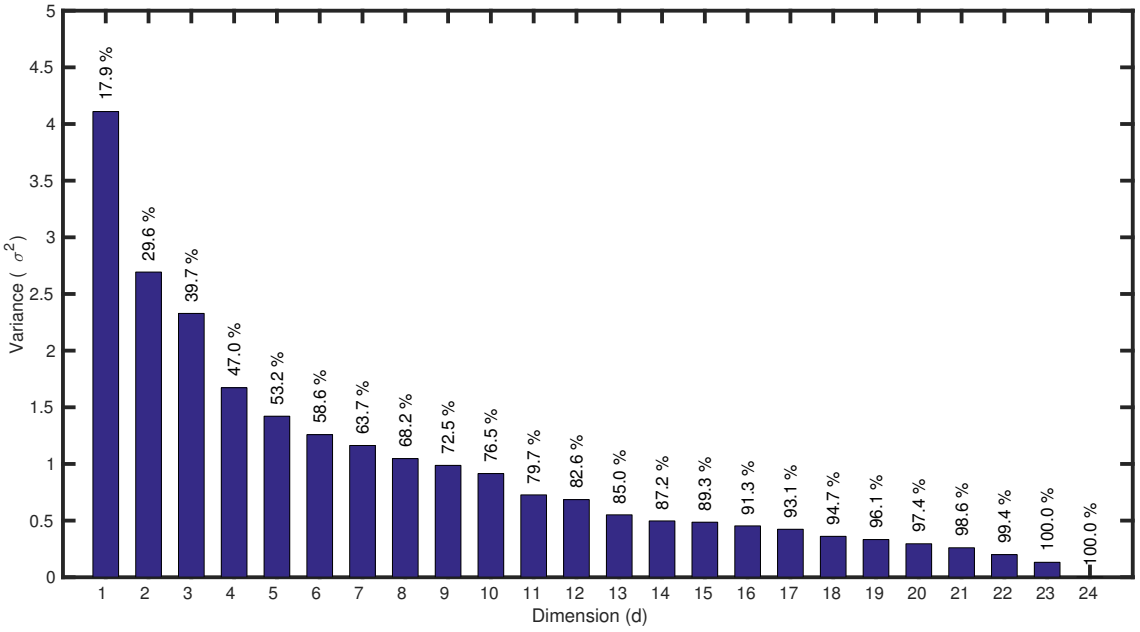
Figure 5.3: First and second principal components of BoD (a, b) and accelerometer (c, d) features from 10 random participants. Each colour/symbol combination represents one participant. The ellipses correspond to the Gaussian for each participant. For simplicity, only 100 observations are shown from each participant.

pertaining to hand grip sits beyond the first and second principal components, indicating that many components are required to capture most of the variance.

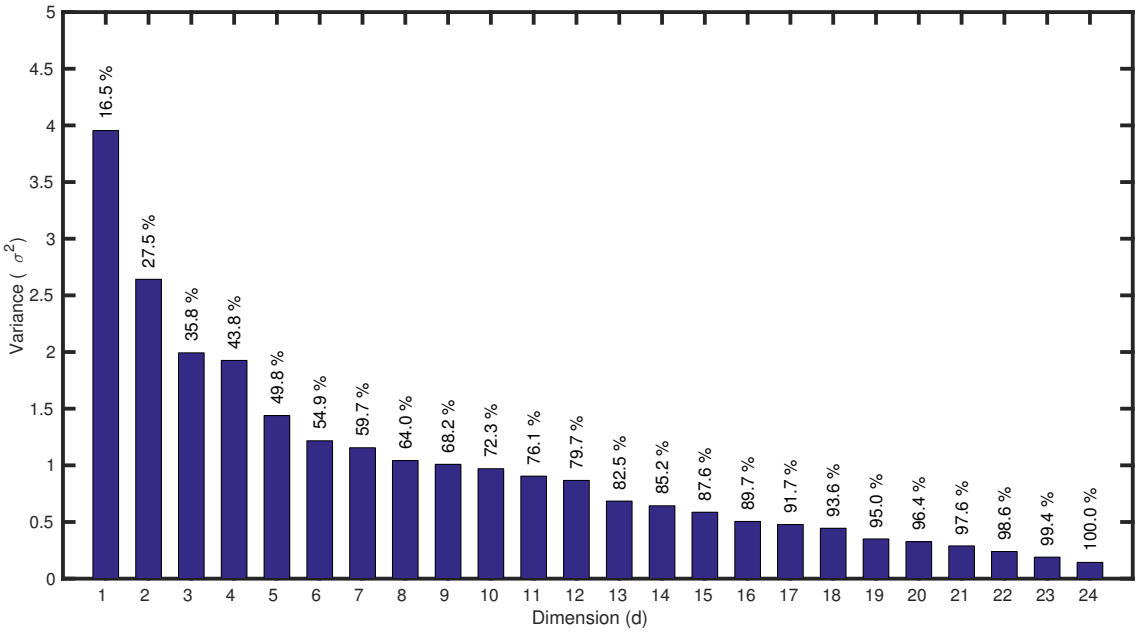
Besides per-participant analysis, we also performed per-hand transformations on the feature set. This is to see if hand grip and pose differs between hand. In order to do this, we perform PCA on 100 random hand grip and pose examples of both hands pooled from 10 random participants. From Figure 5.5 both **BoD** and accelerometer features show distinct groupings between right and left hand. This suggests that it is possible to determine handedness using either of the sensor input.

5.3.1 Correlation Analysis

To investigate the relationship that exists between touch position and **BoD** and accelerometer features, we used Canonical Correlation Analysis (CCA). As mentioned in Section 5.2.1, CCA measures the correlation between $\mathbf{x}$ and touch location $\mathbf{y}$ by finding the optimal linear combinations of variables from each modality such that the correlation is maximised. Simi-



(a) Left hand

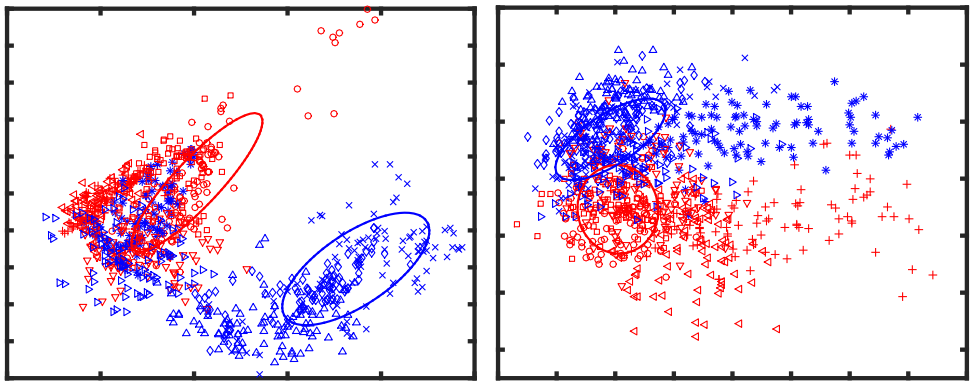


(b) Right hand

Figure 5.4: Variances explained (in percent) for each principal component for BoD feature. It is clear that the variances decay slowly, indicating that the information is dispersed across many principal components.

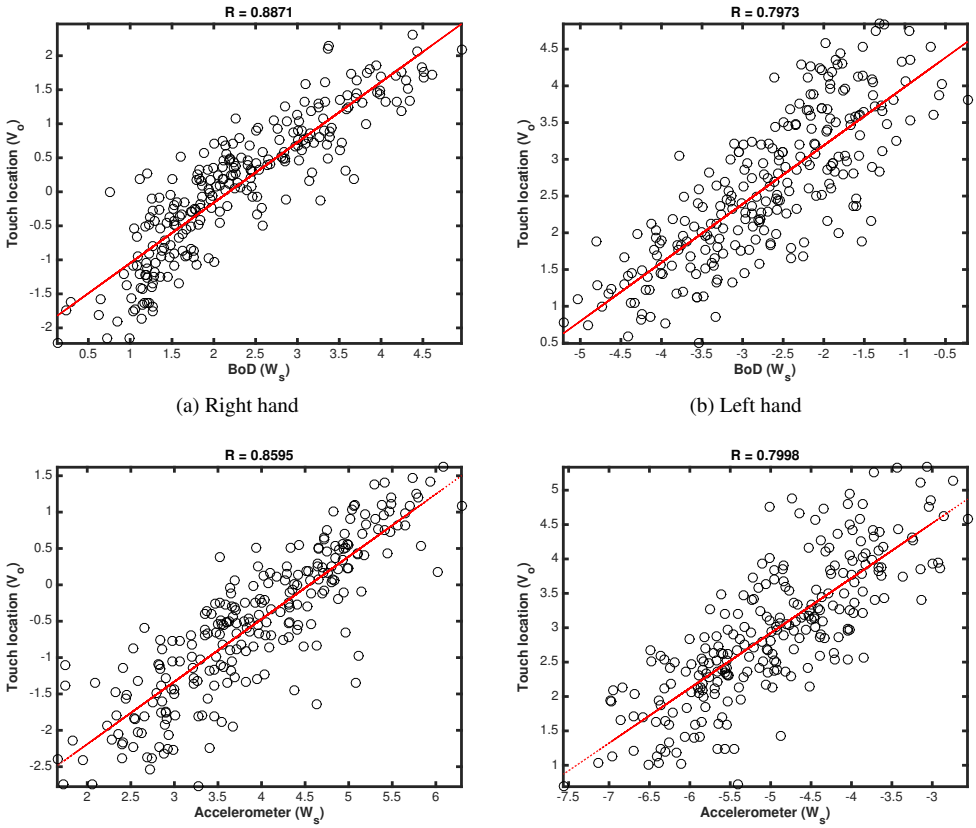
lar to the data treatment we used in PCA, we standardised our feature set before measuring the correlation.

This analysis attempts to answer if sensor inputs and touch locations are correlated (RQ2). We also would like to test if there is a clear correlation in both user-specific and pooled settings. To do so, we performed CCA in individual and pooled settings. In the former, we used observations from one participant selected randomly and the latter was done by pooling



(a) BoD (b) Accelerometer

Figure 5.5: First and second principal components of **BoD** (a) and accelerometer (c) features from 10 participants. Red and blue markers represent right and left hand data respectively and each symbol represents one participant. The ellipses correspond to the Gaussian for each hand. For simplicity, only 100 observations are shown from each participant.

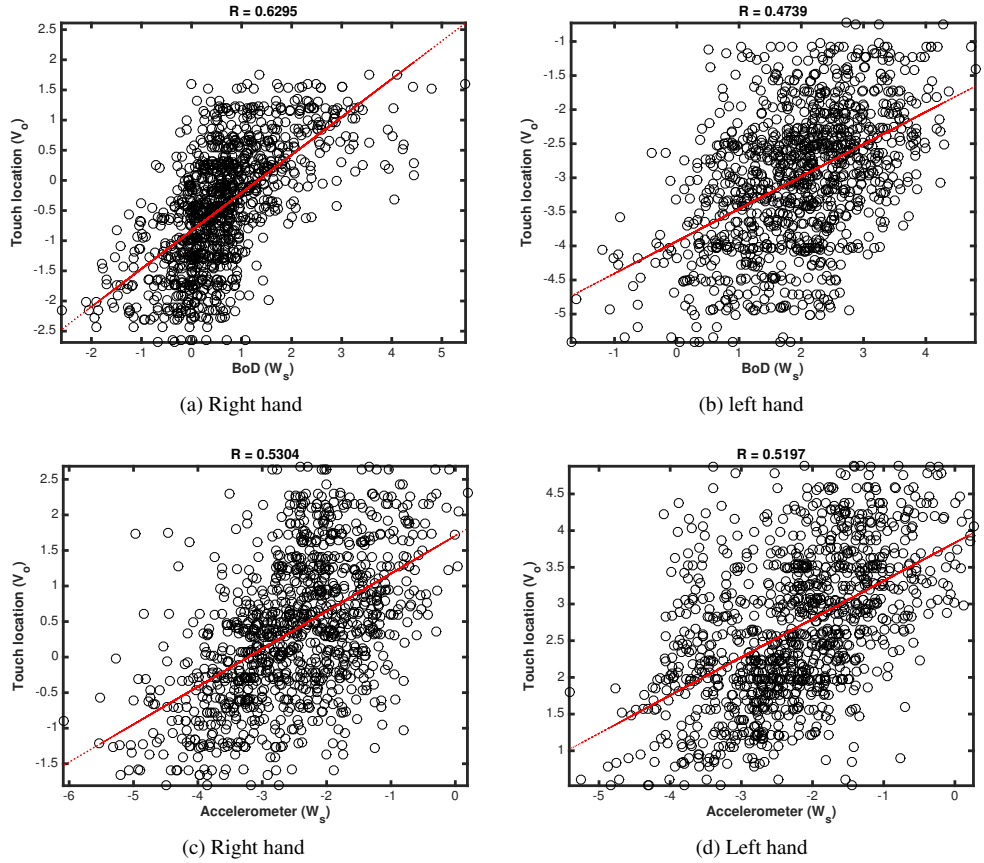


(a) Right hand (b) Left hand

(c) Right hand (d) Left hand

Figure 5.6: Examples of the first canonical component using individual data from one user selected randomly for each hand and for each sensor. x and y axes correspond to the first canonical variate W_s and V_o respectively. Red line corresponds to the least-squares line.

100 random observations from all participants. We show the first and canonical components (the most correlated) from this experiment for both input modalities and for both hands in



(a) Right hand (b) left hand

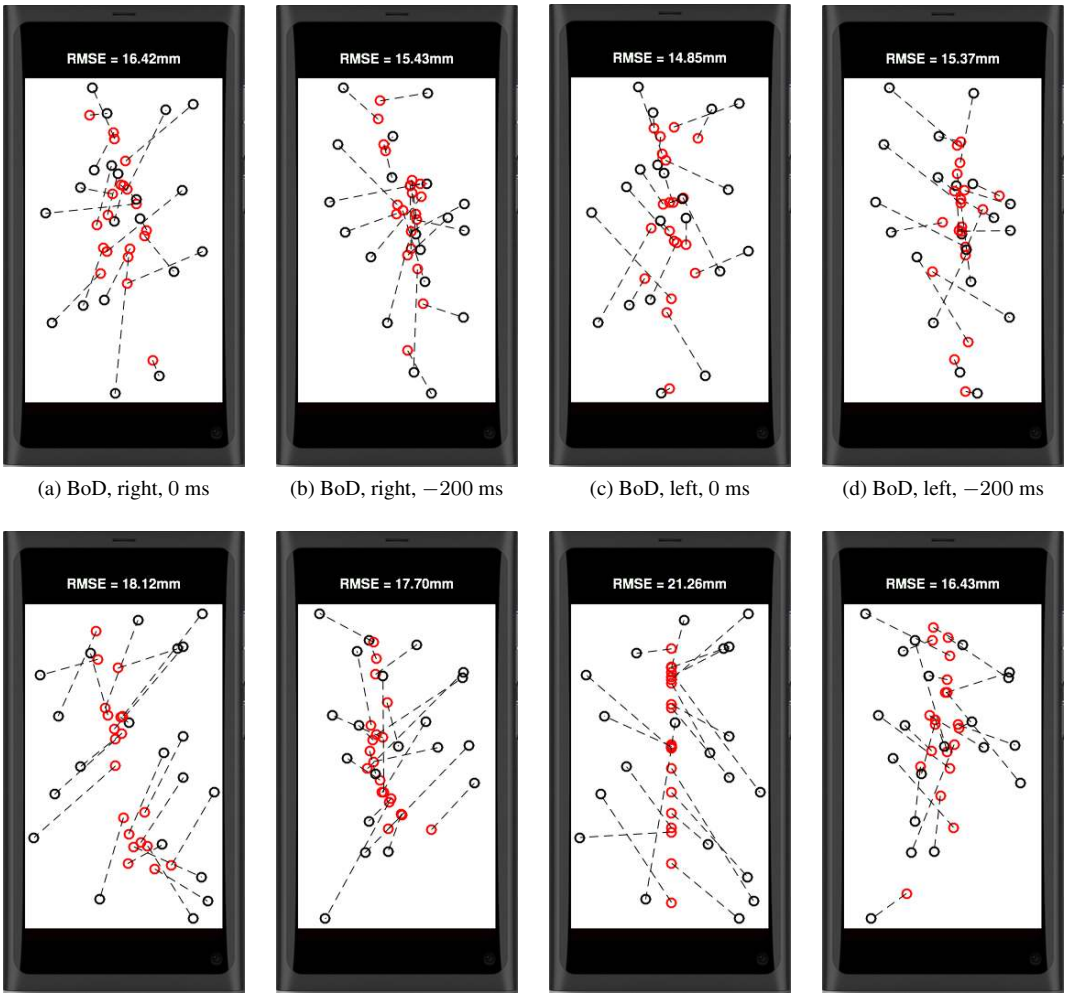
(c) Right hand (d) Left hand

Figure 5.7: Examples of the first canonical component using pooled data for each hand and for each sensor. x and y axes correspond to the first canonical variate W_s and V_o respectively. Red line corresponds to the least-squares line.

Figure 5.6 and 5.7.

The scatterplots of CCA show that there is a correlation between individual feature input and touch location (CCA determines the weighted average (canonical roots) of $\mathbf{x}$ and $\mathbf{y}$) for both hands and sensors. The statistical test (Wilks' Lambda) on the first canonical correlation coefficient shows that the correlation is significant for left hand BoD ($p < 0.001$) and right hand BoD ($p < 0.001$). The correlation also is significant for both left ($p < 0.0001$) and right ($p < 0.0001$) hand using accelerometer. Based on this test, we can reject the null hypothesis that all canonical variate pairs are uncorrelated.

The correlation is apparent when we used observations from all participants, however the correlation coefficient is lower compared to when using individual data. Given these correlations, we anticipate good touch location predictions using BoD and accelerometer sensors values.



(a) BoD, right, 0 ms (b) BoD, right, -200 ms (c) BoD, left, 0 ms (d) BoD, left, -200 ms

(e) Accelerometer, right, 0 ms (f) Accelerometer, right, -200 ms (g) Accelerometer, left, 0 ms (h) Accelerometer, left, -200 ms

Figure 5.8: Example of touch target predictions for a random participant, for both hands and sensor inputs at $t = 0$ms and $t = -200$ms. Black markers correspond to real targets and red markers correspond to the predicted targets.

5.3.2 Touch Prediction

Predicting intended touch locations from BoD and accelerometer sensor inputs is the main goal in this study (RQ2). Therefore using GP regression, the task is to find mapping f between a set of inputs: BoD $\mathbf{x}^c$, accelerometer $\mathbf{x}^a$ and touch targets $\mathbf{y}$ with the purpose to infer the relationship f between sensor input $\mathbf{x}$ and touch targets $\mathbf{y}$:

$$f(\mathbf{x}) = \mathbf{y}$$

By using GP we assume that regression to be non-linear. Despite CCA depicting linear correlation, a non-linear ML model may capture more of the information since the linear correlation is not 1. Morever, we use GP mainly due to its flexibility (i.e. choice of covari-

ance functions). Although other techniques such as Neural Network exist, these techniques however requires a lot of data which is one of the limitations of this study.

As described in Section 5.2.2, we would like to evaluate the prediction performance at different point of times – that is by varying the time where the GP is trained. This is to see how early the GP can predict the touch target. In all experiments, the data are split into independent training and test sets. For simplicity, individual GP models are trained for each dimension (co–ordinate axis). A more elaborate mode would correlate the two dimensions though (e.g. a multi-task kernel).

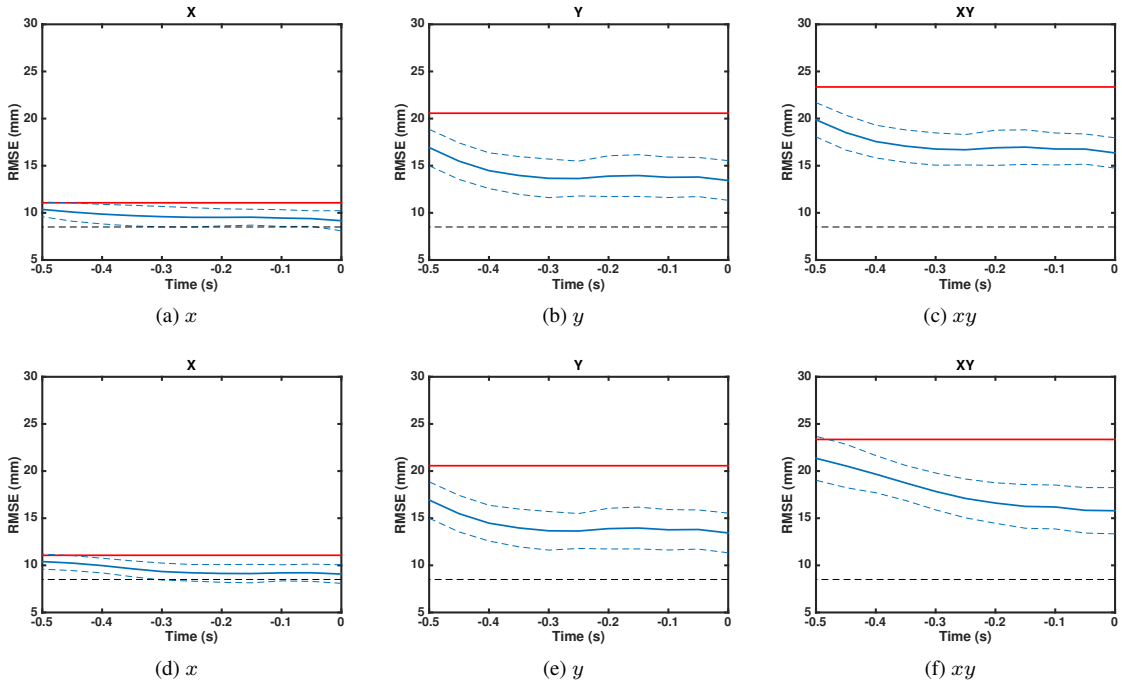
We used data from the first to fourth sessions as training set and the fifth session as a testing set. The hyper-parameters are optimised by maximising the marginal likelihood on the training set [98]. As described in Chapter 4.4, predictions using GP regression are probabilistic, however we are only interested in estimating touch locations and thus neglecting the probabilites (uncertainty) of the predictions. Nevertheless the probabilistic nature of GP is exploited in the training of the hyperparameters though the evidence optimisation.

Figure 5.8 shows predictions of 25 touch targets selected randomly from the test set of a random participant at $t = 0$ ms and $t = -200$ ms. From the plots, it shows that even if we use data captured before the touch, GP is still able to predict touch target with a relatively low root-mean square error (RMSE).

In Figure 5.9 and 5.10, we show the mean prediction RMSE from all participants when we vary the time up to 500 ms before touch. In general, using BoD data, the RMSE decreases as the thumb approaching touch target ($t = 0$), however using accelerometer data (Figure 5.10), it shows that the RMSE increases slightly at -300ms bringing the RMSE to approximately 20 mm for right hand and 19 mm for left hand at -100 ms. This indicates that participant's hand pose may not change that much compared to hand grip when targeting targets on the touch screen.

To evaluate the performance of the GP, we compare RMSE of all predictions with the baseline performance and standard Android touch button size. As mentioned in Section 5.2.2, baseline performance is the RMSE of always predicting the centre of the screen. In all cases, we would like the GP to predict better than random. Comparing with standard Android touch button size allows the performance to be interpreted in a more realistic context. In Figure 5.9 and 5.10, random performance is denoted by a solid red while dashed black line corresponds to standard touch button size. From the results it is clear that all predictions remain below the baseline RMSE for both sensor inputs. However, the RMSE is generally higher than the size of standard touch button. This indicates that both sensor inputs can be used to predict coarse touch locations (i.e. top right area, bottom left area, etc.), however may not be sufficient to predict intended exact touch target precisely.

Given the two input modalities that we have, it is interesting to see how the combination



(a) x (b) y (c) xy

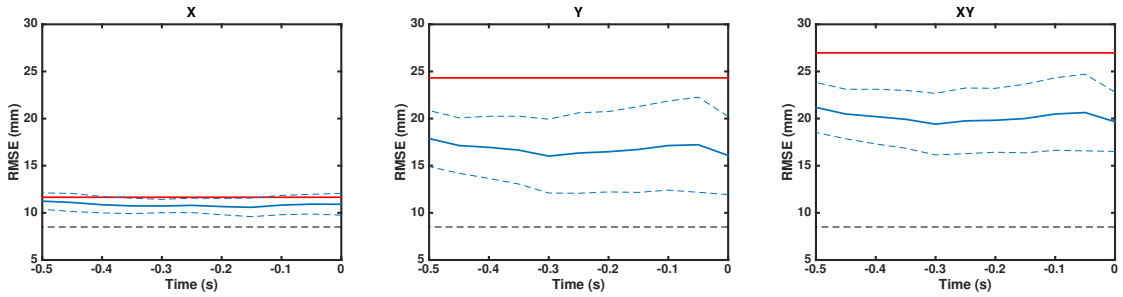
(d) x (e) y (f) xy

Figure 5.9: RMSE of touch target predictions using BoD data including $\pm$ standard errors before touch contact for x and y axis and combination of both axes for right (a, b and c) and left (d, e and f) hand averaged across 20 participants. Solid red and black dashed lines correspond to baseline RMSE and standard Android button size respectively.

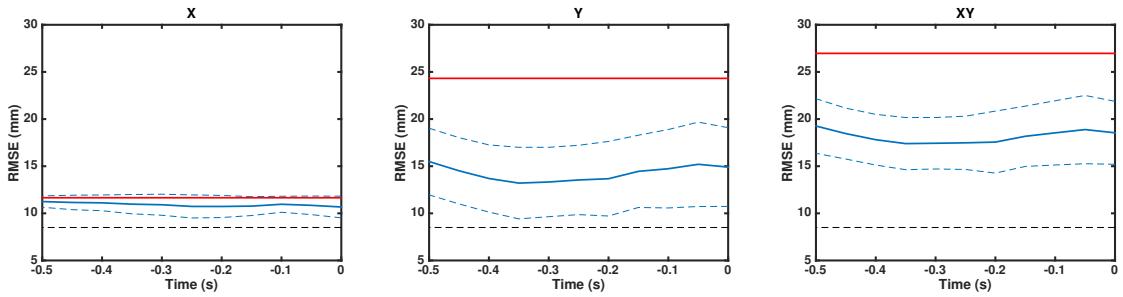
		RMSE (mm)				
	Hand	**−400ms**	**−200ms**	**0ms (touch)**	**Baseline**	**Button size**
BoD	Left	19.67	16.61	15.80	**23.36**	**8.50**
	Right	17.57	16.90	16.35		
Accel	Left	17.81	17.55	18.54	**26.97**	**8.50**
	Right	20.21	19.82	19.66		
Comb	Left	16.48	14.79	13.94	**20.88**	**8.50**
	Right	16.66	15.53	14.93		

Table 5.1: Mean RMSE of touch target predictions at 400ms, 200ms and 0ms (touch) before touch contact using BoD, accelerometer and combination of both data. Baseline is RMSE of predicting the centre of the screen. Button size RMSE follows the size of standard Android touch button.

of both modalities would influence the predictions. In order to combine both modalities, we simply concatenated both feature vectors resulting in a new 85 dimensional composite feature vector $\mathbf{x}^{ca}$. We then repeat the experiment using this new feature to predict touch location $\mathbf{y}$ by varying the time before touch contact. Prediction results using composite feature for both hands are shown in Figure 5.11. From the plots, it is clear the combination

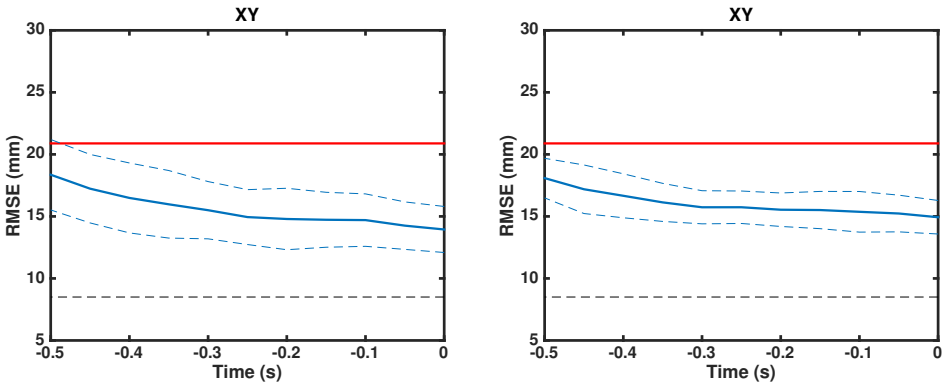


(a) x (b) y (c) xy



(d) x (e) y (f) xy

Figure 5.10: RMSE of touch target predictions using accelerometer data including $\pm$ standard errors before touch contact for x and y axis and combination of both axes for right (a, b and c) and left (d, e and f) hand averaged across 20 participants. Solid red and black dashed lines correspond to baseline RMSE and standard Android button size respectively.



(a) Left (b) Right

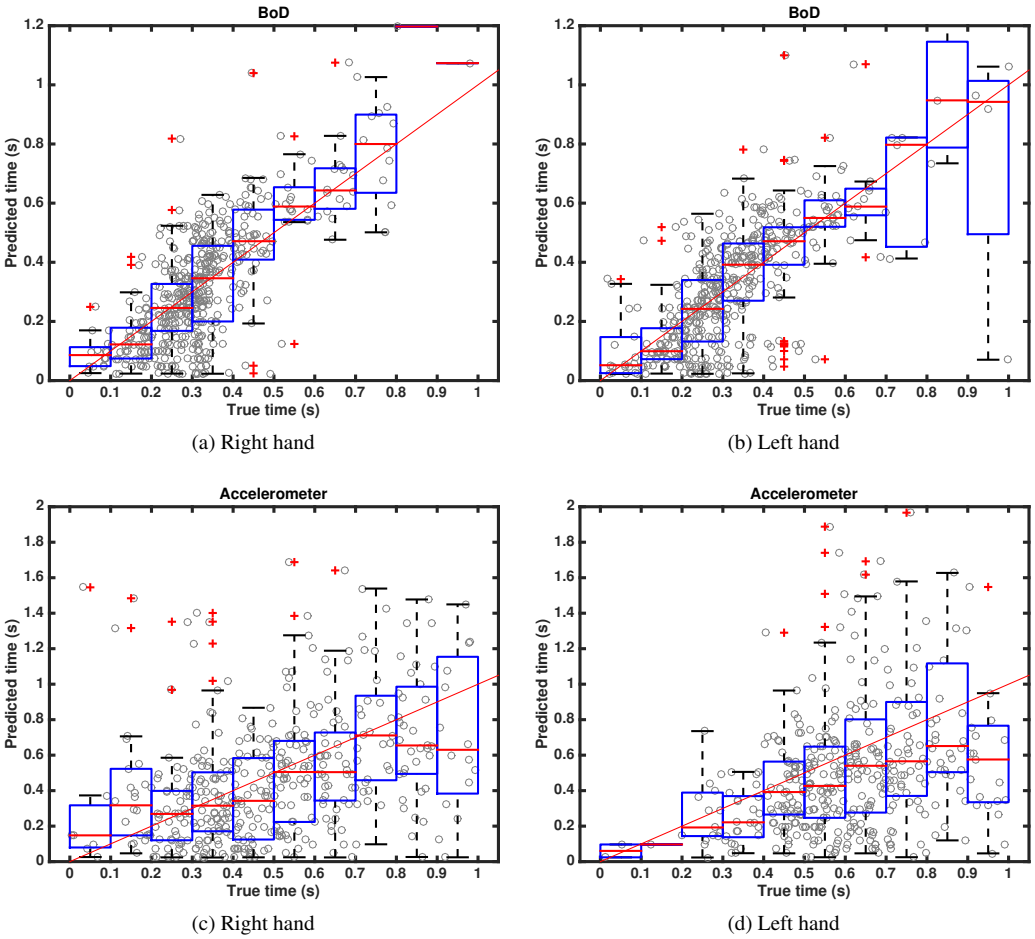
Figure 5.11: RMSE of touch target predictions using combination of BoD and accelerometer data including $\pm$ standard errors before touch contact for combination of both axes for both hands averaged across 20 participants. Solid red and black dashed lines correspond to baseline RMSE and standard Android button size respectively.

of both modalities produces lower mean RMSE for both hands compared to using BoD or accelerometer data alone.

Table 5.1 summarises the mean RMSE for all touch location predictions (xy) for both input modalities and combination of both for left and right hands. Compared to accelerometer, touch location can be predicted statistically significant better using right hand BoD data at

400ms ($p < 0.05$) and 200ms ($p < 0.05$) before touch contact. However when using left hand data, the difference is not statistically significant despite the BoD producing lower mean RMSE. Similarly, the difference in mean RMSE between right and left hands is not statistically significant for both input modalities for all time points before touch contact. Although using composite feature greatly reduced the mean RMSE of the predictions, only prediction at 200ms ($p < 0.05$) before touch contact is statistically significant.

5.3.3 Time of Touch Contact Prediction



(a) Right hand (b) Left hand

(c) Right hand (d) Left hand

Figure 5.12: Distribution of 500 time predictions from both hands using BoD (a, b) and accelerometer (c, d) examples within 0 ms – 1000 ms time range, pooled randomly from all participants, from all sessions.

Besides predicting intended touch locations, we also are interested to find if sensor inputs can be used to make time of touch contact predictions (RQ3). Therefore using GP regression, the goal here is to find a mapping f between a set of input: BoD $\mathbf{x}^c$, accelerometer $\mathbf{x}^a$ and touch target contact time t.

$$f(\mathbf{x}) = \mathbf{t}$$

In all experiments we pooled 2500 random examples within 0 ms to 1000 ms time before touch range from all participants. From this, we used 2000 examples to train the GP and the remaining 500 examples for testing. All data were standardised (mean = 0, std = 1) prior to training and testing. Here we again used squared exponential (SE) covariance function to train the GP (see Section 5.2.2). The hyperparameters were optimised by maximising the marginal likelihood on the training set.

The distributions of predicted time against actual time for both both sensor data are shown in Figure 5.12. From the boxplots, it is apparent that time of touch contact can be predicted using both input modalities. Using BoD data, the time of touch contact for both right and left hands can be predicted reasonably well up to 700 ms before touch contact. This suggests that left and right grips are similar in temporal domain.

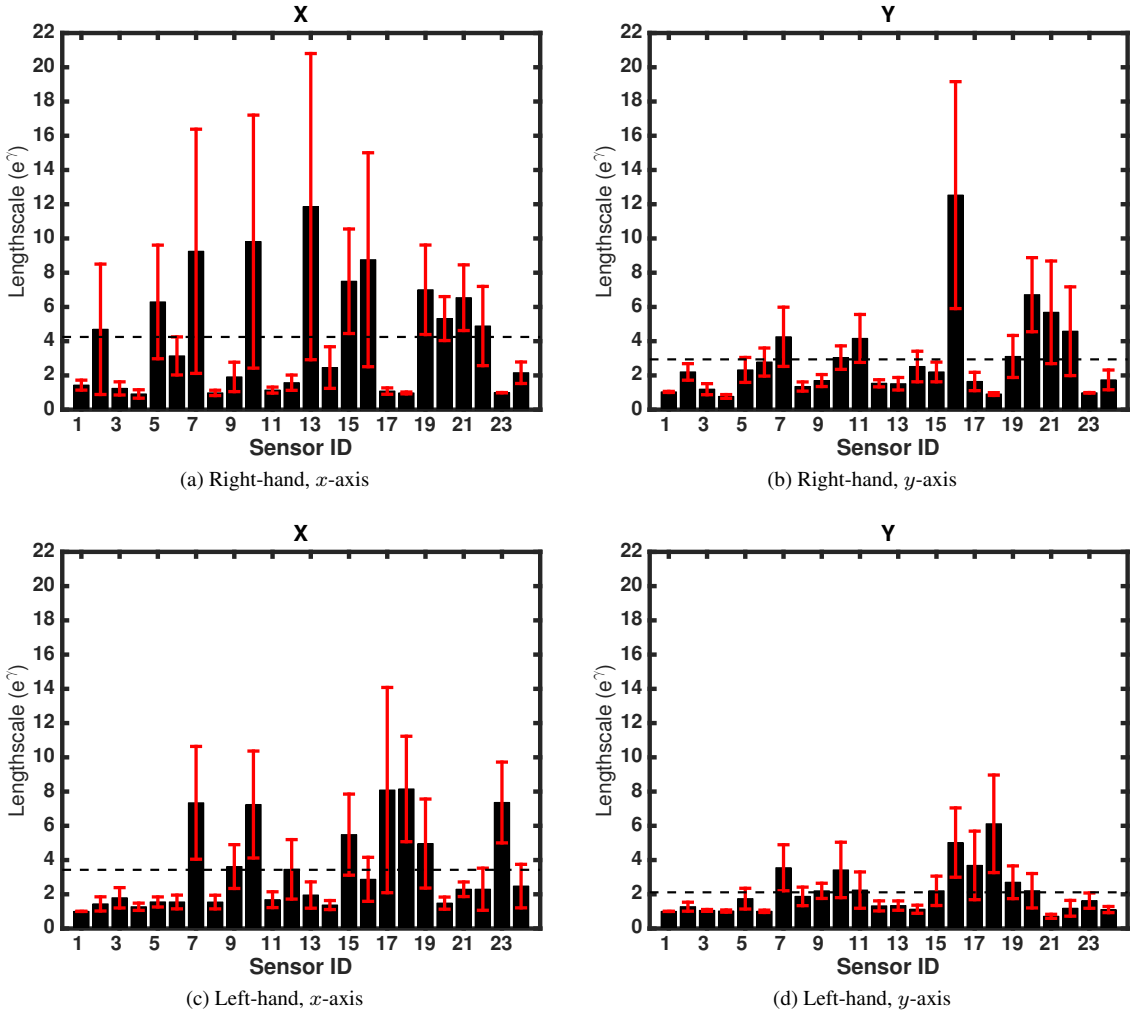
Predictions using accelerometer data also show similar performance for both hands, indicating that regression for this task may be invariant on poses from any hand. As shown in Figure 5.12 (c) and (d), the GP was able to predict time of touch contact relatively well up to 500 ms for both right and left hands. However, regression using accelerometer data shows more outliers compared to BoD, suggesting that accelerometer may not be as stable as BoD grip contact.

Apart from touch location, our results indicate that both modalities also are correlated with time of touch contact. As discussed earlier in this chapter, if time of touch contact can be determined ahead of time, it can be used as an indicator to queue auditory and tactile touch feedback so that they can be triggered them exactly at the predicted touch contact times. This in return can potentially decrease the latency of feeding in the feedback into the system.

5.3.4 Sensor Relevance

In all experiments we use 24 BoD capacitive sensors to capture hand grip contact with the purpose to predict intended touch locations. Whilst the results from Section 5.3.2 and 5.3.3 indicate that input from 24 sensors could produce low RMSE, we are curious to see the relevance of each sensor to the GP regression task. In other words, the relevancy test could determine which sensors are redundant and give an insight on possible minimum number of sensors required to make predictions (RQ4).

Using the GP, the relevancy of the sensors can be determined by finding redundant mappings between sensor input $\mathbf{x}$ and touch targets $\mathbf{y}$. A redundant mapping is characterised by an input (sensor) with a very large characteristic lengthscale hyperparameter γ [98]. As mentioned in Section 5.2.2 we use GP with squared exponential (SE) covariance function that



(a) Right-hand, x-axis

(b) Right-hand, y-axis

(c) Left-hand, x-axis

(d) Left-hand, y-axis

Figure 5.13: Mean hyperparameter characteristic lengthscale for each BoD sensor, for each axis determined by ARD for both hands from all participants at 200ms before touch. The horizontal black dashed line corresponds to the threshold used to determine sensor relevancy.

implements Automatic Relevance Determination (ARD) to learn which sensors are relevant for predictions.

To determine relevant sensors, we repeat the touch location prediction experimentsment using 24-dimensional **BoD** feature from all participants. We used data from the first to fourth sessions as training set and the fifth session as testing set. SE covariance function with ARD has a separate lengthscale for each input dimension $m = 24$, for a total 25 hyperparameters (including signal variance σ^2 hyperparameter). All lengthscale and signal variance were initialised to 1 (in log space). This is due to the fact that the input has been re-scaled to $[0, 0.5]$. The signal variance was set to 1 to reflect the geometry of the screen which is bounded. We optimised the hyperparameters by maximising the marginal likelihood on the training set.

We show the mean sensor hyperparameter lengthscales from all participants for each axis at 200ms before touch in Figure 5.13 (a) and (c). Notice that some sensors have very large

		Sensor ID																							
	Hand	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
x	Left	○	○	○	○	○	○	×	○	×	×	○	×	○	○	×	○	×	×	×	○	○	○	×	○
	Right	○	×	○	○	×	○	×	○	○	×	○	○	×	○	×	×	○	○	×	×	×	×	○	○
y	Left	○	○	○	○	○	○	×	○	×	×	×	○	○	○	×	×	×	×	×	×	○	○	○	○
	Right	○	○	○	○	○	○	×	○	○	×	×	○	○	○	○	×	○	○	×	×	×	×	○	○

Table 5.2: Tabulation of BoD sensors according to their lengthscale for each axis and hand. Circle corresponds to a sensor with lengthscale smaller than the threshold lengthscale (relevant). Cross corresponds to a sensor with lengthscale larger than the threshold lengthscale (redundant).

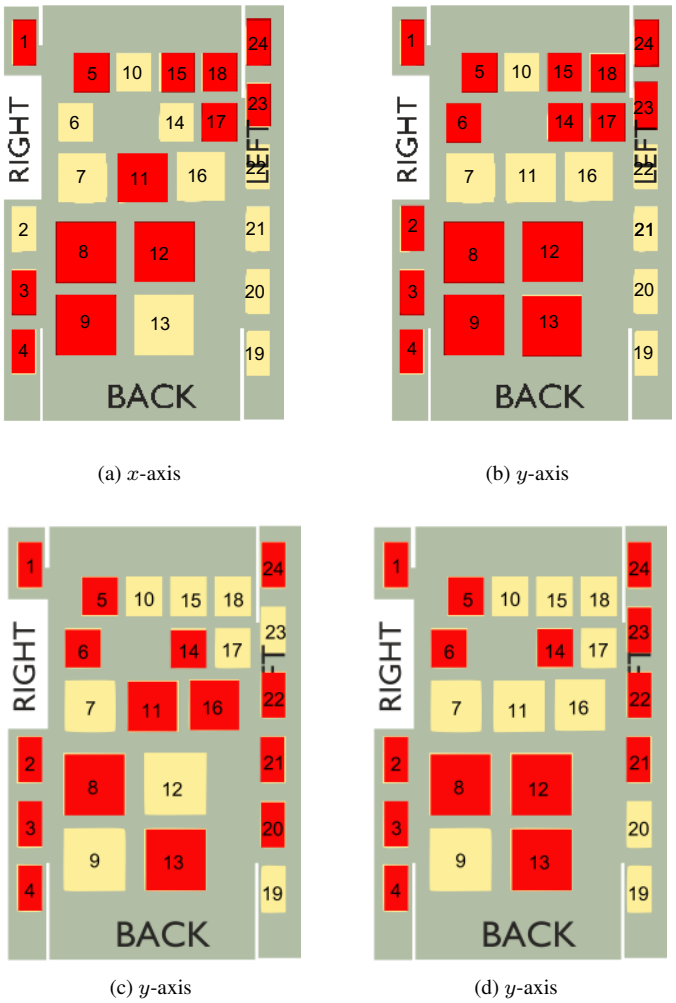
hyperparameters lengthscales, particularly for x-axis. Table 5.2 shows the tabulation of relevant and redundant sensors by comparing individual sensor's lengthscale with the threshold lengthscale (mean lengthscale from all 24 BoD sensors). For convenience, we marked (red) the position of relevant sensors in Figure 5.14 (a) and (b). From the diagram, it appears that these sensors are located around the palm and finger tips areas.

To test whether fewer sensors can be used to predict touch location, we again repeat the experiment in Section 5.2.2 using only the relevant BoD sensors for x-axis and y-axis estimated at 200ms before touch contact. Results from this experiment are shown in Figure 5.15. In comparison to the results with all sensors (Figure 5.9), the RMSE curve decays quite linearly with time before touch contact when using fewer sensors. This is expected since using fewer sensors resulted in information loss, particularly when we varied the time further back before touch. The summary of the predictions are given in Table 5.3. From the table, it shows that the lowest mean RMSE at 200ms before touch for right hand is given by sensors for x–axis while for left hand, this is given by y–axis. These are comparable and not significantly different from the mean RMSE of predictions with all sensors at similar time before touch contact.

		RMSE (mm)				
	Hand	**400ms**	**200ms**	**0ms (touch)**	**Baseline**	**Button size**
x	Left	20.58	18.73	17.40	**26.46**	**8.50**
	Right	20.01	17.67	15.24	**22.84**	
y	Left	19.53	16.74	15.80	**21.84**	**8.50**
	Right	23.75	21.62	18.31	**26.78**	

Table 5.3: Mean RMSE of touch target predictions at 400ms, 200ms and 0ms (touch) before touch contact using only the relevant BoD sensors. Baseline is RMSE of predicting the centre of the screen. Button size RMSE follows the size of standard Android touch button.

Our results indicate that it is possible to use fewer than 24 sensors to make touch location predictions without impacting the performance significantly. However the location of these



(a) x-axis (b) y-axis

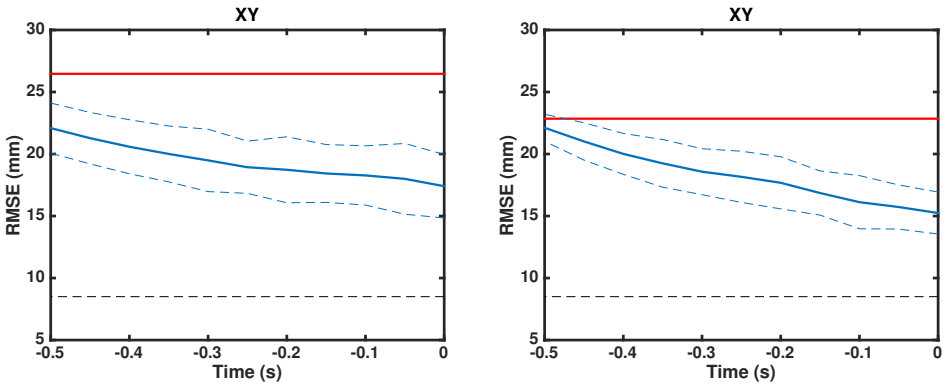
(c) y-axis (d) y-axis

Figure 5.14: Position of relevant BoD sensors (marked red) from Table 5.2 for x- and y-axis prediction for right (a, b) and left (c, d) hands determined using GP with ARD.

sensors is crucial in order to capture all plausible grips to predict touch locations. In most cases, using fewer sensors is favourable in mobile devices since it could simplify the sensing mechanisms, reduce power consumption as well as minimise the cost of manufacturing. Furthermore, sensor reduction also could help to simplify the learning algorithm, in our case the GP, to learn the mappings efficiently.

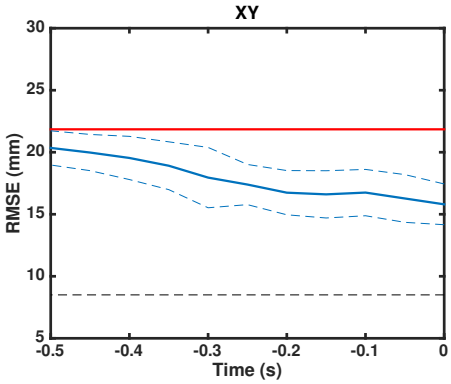
5.4 Online Touch Prediction System

To demonstrate how our technique would perform in the actual product, we implemented an online prediction system on the N9 in real-time. Due to N9's limited computational power, we wrote a simplified version of GP predictor based on Python. Similar to the pre-processing technique described in Section 5.2, we configured the N9 to sample data at 100Hz and ap-

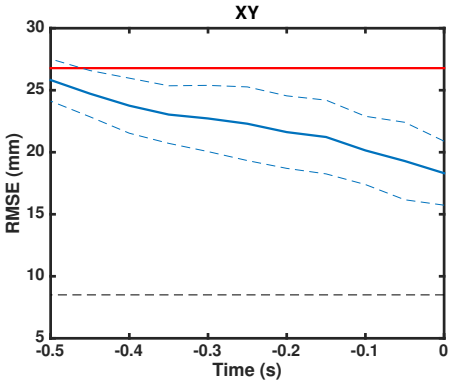


(a) Left

(b) Right



(c) Left



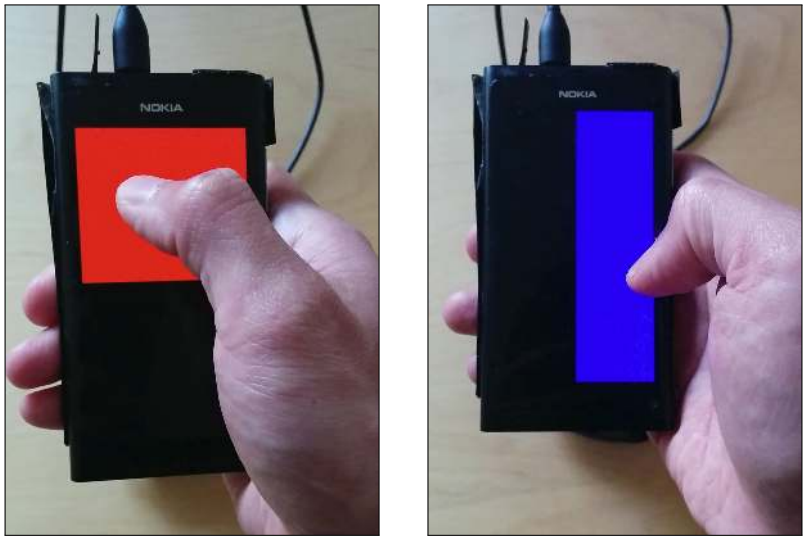
(d) Right

Figure 5.15: RMSE of touch target predictions using relevant sensors for x– (a, b) and y–axis (c, d) as tabulated in Table 5.2, including $\pm$ standard errors before touch contact for combination of both axes, for both hands averaged across 20 participants. Solid red and black dashed lines correspond to baseline RMSE and standard Android button size respectively.

plied Savitzky-Golay smoothing filter to the capacitive signals prior to predicting every touch location. The predictor was based on the BoD only GP models, which were trained offline using data pooled from all participants.

We ran the prediction in two modes: horizontal and vertical. This was done by splitting the screen equally into two halves, and predict which half the thumb was going to touch. The system will highlight the half it is predicting (blue/red) before the thumb touches it, and stop predicting when it detects touch contact. We let the participants use the system, however no formal evaluation was made on the predictions.

Our online prediction system demonstrates how implicit BoD sensing can be used to predict touch location in real-time. As shown in Figure 5.16, the blue and red bars on the the screen represent the probable area that is going to be touched, which can contain many UI components such as buttons, hyperlinks in the web browser and icons. As discussed earlier in this chapter, we can pre-load the associated content to these respective UI components as soon as we know the area to be touched, thus reducing launching/loading times.



(a) Horizontal (b) Vertical

Figure 5.16: Real-time prediction using simplified GP predictor on the N9 in (a) horizontal and (b) vertical modes.

5.5 Conclusion

In this study we presented a technique to predict screen touches from **BoD** grip and hand pose. We collected touch–grip/pose examples from 20 participants and we formulated four main research questions that we answered through scientific experiments.

In the first question, we are interested to test if there is any relationship between touch locations and sensor inputs (RQ1). To answer this we measured the correlation between multi-dimensional sensor inputs and touch location (x, y coordinates). The analysis revealed that there is a linear correlation in both individual data and data pooled from all participants.

Given the relationship shown in the correlation analysis, we are curious to see if we can make touch location predictions ahead of time from sensor data, which is our second research question (RQ2). Even tough there is a linear correlation, a non-linear ML model may capture more of the information. Thus by using machine learning, we trained a GP and made touch location predictions on a per– and pooled–participant basis. Our method achieved RMSE of 16mm using **BoD** data and 17mm using accelerometer data extracted at 200ms before touch. With this level of error, we conclude that touch location prediction can be made from sensor data.

Apart from touch locations, we are also intrigued to see if time of touch contact can be predicted from sensor data (RQ3). In order to answer this, we performed another experiment where we trained the GP to learn mappings between sensor data and time of touch contact. Results show that using **BoD** data, time of touch contact can predicted fairly well up to 500ms before touch. This shows that apart from touch locations, sensor data can also be

used to predict time of touch contact.

Finally we would like to investigate whether every BoD sensor is relevant to the prediction task (RQ4). This is answered through an experiment using GP with ARD covariance function. From this experiment we show that not every sensor is relevant and therefore can be effectively removed from the input. Moreover, we also show the location of the relevant sensors on the BoD PCB layout to illustrate the effective sensor location to capture hand grip.

Besides achieving interesting result in touch prediction, our technique could be applied to extend user interfaces (e.g. with mid-air *taps*) or to improve error correction (e.g. as a more robust measure of finger *slip*), and it has immediate and compelling application to reduce latency in mobile applications.

Chapter 6

Detecting Swipe Errors on Touchscreens Using Hand Grip and Motion

Part of the work presented in this chapter has been published at the Conference on Human Factors in Computing Systems (CHI) 2016.

Summary

We show that when users make errors on mobile devices they have immediate and distinct physical responses that can be observed with standard sensors. We used three standard cognitive tasks (Flanker, Stroop and SART) to induce errors from 20 participants. Using simple low-resolution capacitive touch sensors placed around a standard mobile device and the built-in accelerometer, we demonstrate that errors can be predicted at low error rates from micro-adjustments to hand grip and movement in the period shortly after swiping the touchscreen. Specifically, when combining features derived from hand grip and movement we obtain a mean AUC of 0.96 (with false accept and reject rates both below 10%). Our results demonstrate that hand grip and movement provide strong and low latency evidence for mistakes. The ability to detect user errors in this way could be a valuable component in future interaction systems, allowing interfaces to make it easier for users to correct erroneous inputs.

6.1 Introduction

Touch gestures are a common way of interacting with smartphones. Common examples are scrolling through a phonebook by flicking upwards, or accepting or rejecting a call by swiping left or right whereby zooming in and out a photo is achieved by pinching and spreading

using two fingers on the touchscreen. Even though interactions can be initiated using other methods such as voice commands, touch gestures are likely to remain as a preferred method since they are analogous to the controls used in desktop environments, making them easier to understand [56].

Despite the simplicity of touch gestures, users often make mistakes when interacting with mobile devices [84], particularly when in attention-demanding situations in mobile contexts. With simple swipes, these mistakes can be both errors in gesture performance and recognition, and higher-level cognitive errors. Detection of these mistakes could be used to support recovery from error. For example, an error-detecting system could provide additional verification to check if the selection was as intended. This process requires timely detection of an *error signal* – a sensor measurement which correlates strongly with user error.

To date, most techniques to identify interaction errors in HCI have been done through EEG-based brain-computer interface (BCI). This is achieved by observing the presence of error-related negativity (ErrN), a sub-component of error-related potential (ErrP) [39, 34, 20]. This is a distinct electrical variation in the EEG occurring 100-300ms after an error has been made [89]. Over the years, BCI equipment has been simplified and made available commercially, allowing this technology to be more approachable by consumers for everyday use. For instance, Vi and Subramanian have used an off-the-shelf EEG headset to detect error-related negativity (ErrN) when a user makes a mistake [136]. Using a similar headset, the ErrN signals present when observing someone else making a mistake are described in [137].

Whilst using brain waves to detect error is well established, it is not a practical approach for most interactions. There is less work on physiological signals and body channels which might produce similar signals when a mistake is made (other than measurements such as GSR or heart rate modulations, which tend to be too slow to identify individual errors in an interaction [122]).

6.1.1 Swipe Errors Detection

In this study, we investigate whether subtle fluctuations in hand grip and movement can be used to detect cognitive errors in gesture based interfaces on a mobile phone.

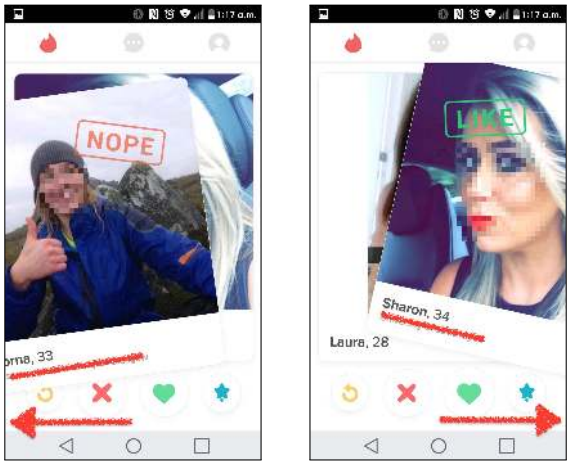
In particular, we are interested to see if hand grip and movement contain sufficient information to reliably detect errors from swipe gestures. In order to do so, we collect swipe gestures from three swipe-based cognitive tasks using a prototype mobile phone and train classifiers to distinguish be tween error-related and correct swipes. We evaluate the performance of both input modalities, and the combination of the two over different period after the swipe is made.

We hypothesise that when an error is made on a touchscreen interface, there are physical changes in the grip and pose of the device, such as gripping the phone more tightly or tilting it. Distinctive modulation of grip as a function of emotional state was detected by Coombes et al. [31] and Noteboom et al. [92] found that pinch grasps were modulated by the potential unpleasant stimuli. Our hypothesis is that when users realise they have made such errors they induce similar modulations of their grasp.

These post-error 'flinches' might be detectable with contact and inertial sensors on the device. We set out to experimentally demonstrate the existence and character of these signals, by augmenting a standard device with wraparound capacitive sensing (and built-in inertial sensing), and designing a series of onscreen tasks to induce cognitive errors.

We use sensors that can be practically integrated into mobile devices: ultra-thin, lightweight, low-cost and low-power capacitive sensors alongside standard inertial sensors. This makes our approach suitable for devices such as tablets and smartphones which could serve as a foundation for sensor-based gestural error detection technique for mobile device.

6.1.2 Example Use Case: Mobile Application



(a) (b)

Figure 6.1: Screenshots of Android Tinder application. Tinder relies heavily on swipes – (a) swipe right to accept or (b) swipe left to pass on. *Undo* features are only available for premium users.

Tinder[1] is a popular dating application that is based entirely on swipes; left to recommend/pass the person to someone else or right to like the person. In the event of a mistake, a user may swipe to the wrong direction, resulting in liking or disliking someone unintentionally, a potentially sensitive and embarrassing mistake. This occurs so frequently that Tinder

[1] `https://gotinder.com`

has added a (premium) undo feature to the application[2]. Prediction of error could be used to identify if the swipe that the user just made was intentional or unintentional, allowing users to retract the swipe without explicit undo. A more sophisticated input could have the user simply reverse the direction of a swipe to cancel a detected possible error.

A similar approach could be applied to common scrolling–based applications such as contact lists. While flicking through a list of names, people often miss the name they were going for. If this 'overshoot' error can be detected, the scrolling can be slowed and gently reversed to transparently recover from the slip. The detection of errors also has applications in instrumented usability and analytics, for logging likely mistakes or confused actions on the part of the user. Developers could use this to optimise error-prone elements of their applications.

6.1.3 Research Goals

In this chapter we investigate the subtle hand movements that occur following mobile device gestures when an error is made. We examine **BoD** capacitive and device motion to infer mistakes in interaction. The underlying hypothesis is that after making a mistake, a user will modulate their grip, affecting the contact with the phone and its orientation. We investigate whether these subtle hand-micro movements captured by **BoD** capacitive sensors and accelerometer can be used to reliably recognise user error. In order to establish our findings, we organise this study around the following research questions in which we attempt to answer through a number of experiments:

RQ1 ***Can we use hand grip and hand movement to detect swipe errors?***
As mentioned earlier in this chapter, we are interested to see if swipe errors can be detected from two input modalities 1) **BoD** hand grip and 2) hand movement. In particular, we would like to measure the performance of this method in recognising swipe errors during the course of interaction. Section 6.2 describes the experiment to answer this question and we present the results in Section 6.4.

RQ3 ***Can we generalise this method across users and tasks?***
Our investigation involves a user study with different swipe-based cognitive tasks. Based on this, we are interested to see if a generic detection model can be established to detect swipe errors across all users and across tasks. In order to answer this question, we run an experiment in Section 6.2, whilst the results are shown in Section 6.4.

RQ4 ***Can we improve the detection by combining both input modalities?***
Apart from evaluating detection performance for each input modality, we are curious to know if combining both inputs could lead to performance gain. We demonstrate

[2] https://www.gotinder.com/faq

our multimodal detection approach in Section 6.2 and we compare our results with individual input approach in Section 6.4.

RQ5 ***How much of data is required to detect swipe errors accurately?***
Our approach detects swipe errors shortly after they happened. To quantify this, we measure detection performance as a function of time. This also indirectly quantifies the amount of data that needs to be observed in order to detect swipe errors accurately. We show the detection performance of by varying the time in Section 6.4.

There are two main factors that motivate using BoD and accelerometer. First, by leveraging built-in sensors, we are able to avoid using additional peripherals (i.e. headsets) to capture error-related signals, making it more practical for real devices. Second, since both **BoD** and accelerometer sensors can be built into a device at a low power cost, the sensors can give an always available channel for detection.

In Section 6.2 where we describe our study in detail, including the experimental protocol used to induce users into making swipe mistakes. In Section 6.3, we explain our swipe error detection method. In Section 6.4, we present the main findings based on the research questions outlined previously. We discuss and give insights about our technique in Section 6.5 before we finally conclude this chapter in Section 6.6.

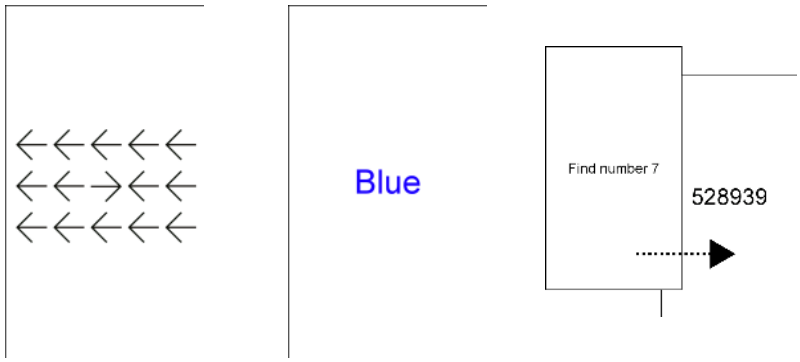
6.2 Experiment

This section discusses about experiments that we designed to answer research questions we established earlier. In general, we investigate if swipe errors can be detected using **BoD** grip and hand movement sensor data. We answer this question through a user study that involves acquisition of swipe response samples from cognitive tasks using the N9 prototype.

6.2.1 Hardware

As mentioned in previous chapters, we used a custom **BoD** capacitive sensors wrapped around a Nokia N9 (see Section 3.3 for details) to capture hand grip. In contrast to tactile sensing used in [49], we opted for capacitive sensing technology because it is a well proven touch sensing technology which is practically implementable on mobile devices. The prototype was configured to sample data from all 24 **BoD** and built-in accelerometer sensors at 150Hz, with 16 bit resolution. A Python client application was developed on a PC to coordinate the data acquisition from the prototype.

6.2.2 Cognitive Tests



(a) Flanker (b) Stroop (c) SART

Figure 6.2: Example of incongruent stimulus used in Flanker and congruent stimuli in Stroop and SART tasks. In (a) and (b) the correct swipe direction would be right, whereby in (c) it would be left.

To stimulate error-related responses, we designed three experiments that require significant cognitive effort, based on well-established protocols in the psychology literature. The first task was based on the Flanker task [38] where the goal is to specify the direction of a central arrow that is bordered by flanking arrows. There were two types of arrows, with each type had two stimuli: congruent stimuli (<<<<< and >>>>>) and the incongruent stimuli (<<><< and >><>>). We also used no-go stimuli (◊◊ <> ◊◊) where no specification of arrow direction is required.

Our second experiment is based on Stroop task [129]. The stimuli used in this task were texts of colours (i.e. *Blue*, *Yellow*, *Green*) where the objective is to determine whether the word matches the colour used by the text. Congruent stimuli are matching colour/word and incongruent stimuli otherwise. For example, the word blue written in blue is a congruent stimuli whilst the word green written in blue is incongruent. For no-go stimuli we used unrelated words within the context of colour (i.e. *Book*, *Grin*, *Back*).

The final task used Sustained Attention to Response Task (SART) [105]. There were two conditions used in this task: Non-Zero and Specific. In the former, a series of 6 digits is presented containing mostly zeros. The objective is to determine if the series contain a non-zero number. In the Specific condition, a series of 6 digits is presented containing different non-zero digits. Here the goal is to determine if a particular digit appears in the series. In contrast to Flanker and Stroop, there is no no-go stimulus used in SART.

We developed a Python application on the N9 to run the experimental software. Unlike classic computer-based cognitive tests which typically use a physical keyboard or mouse, we adapted the tests to use standard swipe-gestures on the N9 touchscreen. Participants responded to the stimulus by swiping the N9 touchscreen horizontally. We picked swiping

over tapping since swiping does not require us to place occluding buttons on the stimulus screen and swiping is a very common task on modern touchscreen devices. We mapped the rightwards swipe for congruent stimuli and left swipe for incongruent stimuli for the SART and Stroop tests. For Flanker, however, the direction of swipe is based on the direction of the central arrow regardless of the stimulus type. Participants were required to withhold their responses (i.e. not touch the screen) when no-go stimuli are presented.

6.2.3 Swipe as a Response

Since our experiments used swipes rather than key presses from dedicated buttons, it is important to clearly define valid swipes. For this purpose, we defined a swipe as a touch stroke with length of at least 30 pixels (3.04 mm). The direction of the swipe is determined by measuring the angle of the straight line connecting the start and end points of the swipe. Duration of swipes was required to be within $50 - 150$ms. The lower limit is to ensure that the swipe was not accidental (click/tap) and the upper limit is to prevent participants from leaving their finger on the touchscreen.

6.2.4 Participants and Data Acquisition

$N = 20$ participants, 18 right-handed and 2 left-handed were recruited locally after thhe formal ethics has been granted. This included 16 males and 4 females, aged between $25 - 35$ (mean = 29.4, sd = 3.15) each with at least one year of experience in using a smartphone. They performed Flanker, Stroop and SART tests whilst seated in a lab, with the phone in a single handed grip, swiping with the thumb in a portrait orientation. All participants were briefed and given a practice session before the main experiment.

For each touch event, we recorded timestamps, accelerometer and BoD capacitive readings for subsequent offline analysis. Each recording was performed in three sessions, separated by an approximately 5 minute break. This was done by asking participants to put down the phone on the table at the end of every session. This ensures that we are not observing temporary grip patterns, but a range of plausible grips for each user.

Each session consisted of 30 trials (swipe responses), resulting in 90 trials in total for each of the three tasks and a total of 270 trials per participant. Each trial begins with a start screen. Prior to showing the stimulus, a 2000ms waiting time is given to allow participants to prepare themselves. We used a red circle at the centre of the screen that would reduce its radius relative to the remaining waiting time. The stimulus is shown immediately after the waiting time. Participants were required to respond to the stimulus within a designated time. We fixed the response time to 1000ms and reduced/increased the time based on the error rate

after the 10th trial. We reduce the time by 100ms if the error rate stays below 25% or increase it by the same amount if the error rate is higher. The minimum and maximum response time were capped at 100ms and 1000ms respectively. This adaptive procedure induces errors at a consistent level across participants of varying skill and performance. A notification of correct or incorrect responses was given immediately at the end of each trial. We used plain black 'Incorrect' and 'Correct' texts on white background as feedback stimuli. This stimuli was shown for 2000ms before returning back to the start screen for next trial.

Overall experiment time ranged from 25 to 43 minutes per participant (mean = 33.74, sd = 5.89). A single Flanker session (30 trials) ranged between $1.9 - 2.4$ minutes (mean = 2.14, sd = 0.12), Stroop between $1.88 - 2.35$ minutes (mean = 2.11, sd = 0.14) and SART between $2.5 - 2.8$ minutes (mean = 2.64, sd = 0.08).

Table 6.1 breaks down the complete set of trials over all participants. On average, performance across the three trials are broadly equivalent with many more correct swipes than incorrect. The number of mistakes per user varies quite dramatically. For the SART test, users have between 0 and 28 errors. Such an imbalance between the two classes poses a challenge to classification. We discuss the approach we took to overcome class imbalance in Section 6.4.

6.3 Analysis

The following section explains the method we used to analyse swipe responses recorded during the user study. The main objective of this investigation is to *detect* errors in swipe responses. We use standard machine learning classification approach where the goal is to solve a binary classification problem: that is to distinguish between mistake (positive class) and non-mistake (negative) swipe responses.

6.3.1 Feature Extraction

The prototype produces 24 capacitive and 3 acceleration time series for each swipe action (one time series per sensor). The BoD sensors in our prototype are occasionally affected by noise spikes which we first remove and interpolating the gaps left by the spikes with a linear fit. This method is sufficient since all spikes in the signal were short (mean = 4 samples).

For each trial we extract a time segment consisting of 1000ms before and after the feedback (notification to the user of whether or not they have made a mistake). In order to create a fixed length representation from the time series (the sampling rate from the sensors exhibited small variability) sampling rate was down sampled linearly to 100Hz. Thus we are left with an equal length of 200 x 24 BoD and 200 x 3 accelerometer time series vectors (100 samples

	Flanker				Stroop				SART			
	Correct	**Incorrect**	**Invalid**	**Late**	**Correct**	**Incorrect**	**Invalid**	**Late**	**Correct**	**Incorrect**	**Invalid**	**Late**
$\bar{x}$	49.65	5.00	4.95	3.30	37.85	10.40	6.85	8.45	64.05	10.95	6.80	6.75
sd	4.60	4.63	2.33	2.68	4.51	7.88	3.76	3.32	8.91	7.55	3.04	3.08
min	40	0	0	2	29	1	4	2	42	0	3	2
max	57	15	9	10	46	26	15	14	77	28	11	11

Table 6.1: Sample statistics (mean, standard error, min (per user) and max (per user)) of correct, incorrect, invalid and late responses made by participants in the cognitive experiments.

before and 100 samples after the feedback (notification)). Both BoD and accelerometer signals were smoothed using an 3rd order Savitzky-Golay filter with frame size of 5. This applies a very mild low-pass filtering.

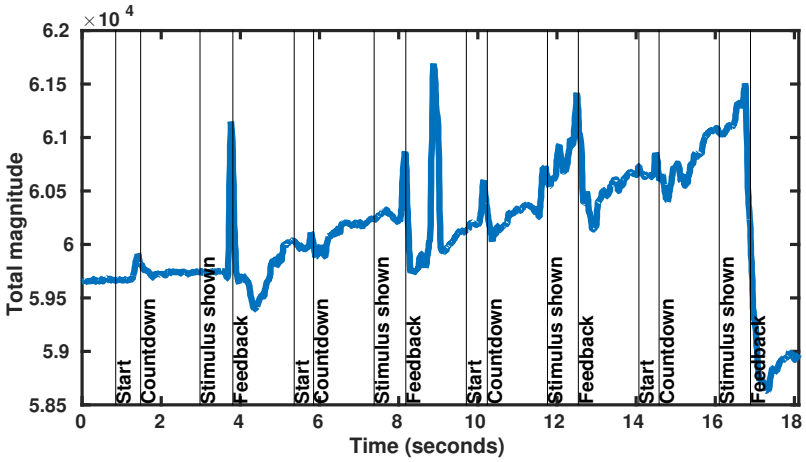
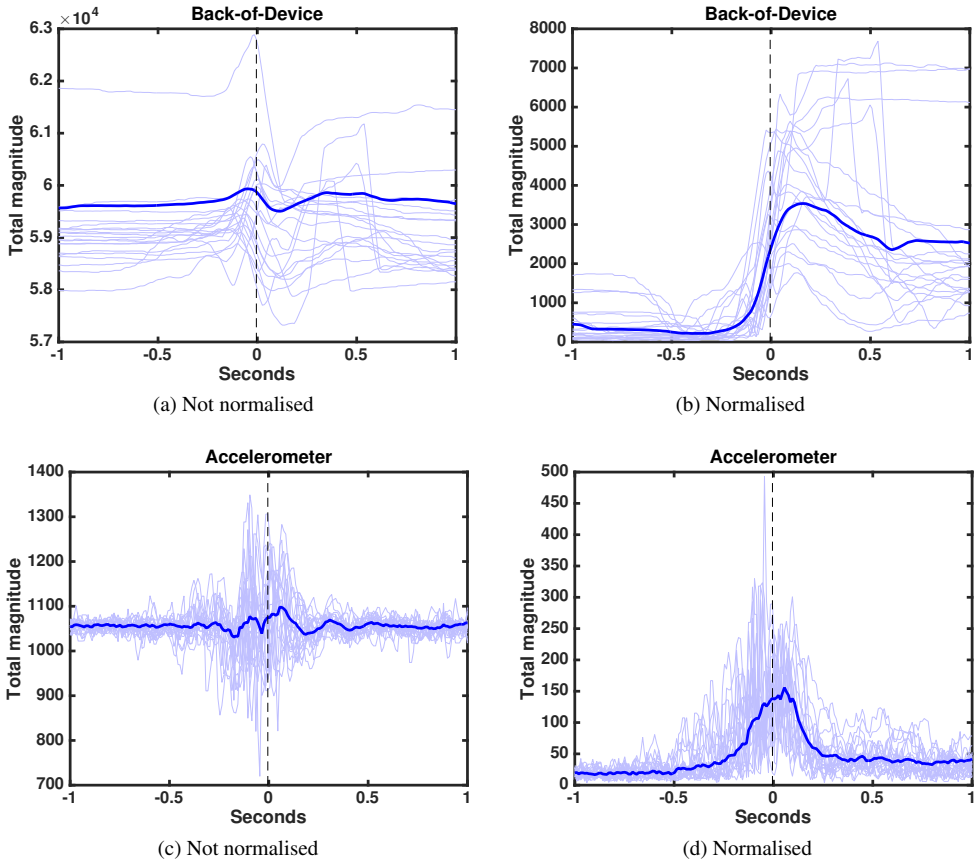


Figure 6.3: Example of the BoD total magnitude signal showing four consecutive trials for one participant in the Stroop task. The vertical lines denote the key moments in the trials.



(a) Not normalised (b) Normalised

(c) Not normalised (d) Normalised

Figure 6.4: 20 swipe trials overlayed from one participant, from the Stroop task before (a, c) and after normalisation (c, d). Vertical dashed line correspond to the moment when the feedback was shown to the user.

The time series typically exhibited long term trends (e.g. gradually increasing grip strength across a session). An example is shown in Figure 6.3. To remove this, data was normalised for each trial by subtracting the mean of the values before feedback from the vector (i.e. for each sensor, the values were normalised to that the time series had zero mean for the first 100 values). Figure 6.4 shows the difference before and after data normalisation.

In all experiments we compared keeping all of the distinct sensor values as well as using the total magnitude (the square root of the sum of squared values). The total magnitude overcomes the problem of handedness (for both $\mathbf{BoD}$ and accelerometer) and gives, for $\mathbf{BoD}$, a value that can be considered a proxy for the grip strength. Comparison of the individual sensors and the total magnitude can therefore help us to decide whether the classification signal is based on grip strength (total magnitude) or also requires grip shape (individual sensors).

6.3.2 Classification

The primary goal of this study is to measure the extent to which hand grip and hand movement can distinguish swipes that are related to mistake. This is as a binary classification task where the problem is to determine whether the swipe belongs to either the positive (mistake) or negative class (non-mistake/normal). As well as investigating if classification is possible, we are interested in investigating classification performance when we vary the time. To this end, in all experiments we investigate features derived from 6 different time segments. Each segment starts at -100ms (100ms before feedback) but has a different end point (0s, 200ms, 400ms, 600ms, 800ms and 1000ms). In all experiments, the data is split into independent training (70%) and testing (30%) sets. Due to very small incorrect response examples in every session, we randomly partitioned incorrect response data instead of partitioning them by session. To perform classification, we used Support Vector Machine (SVM) and Random Forests (RF) classifiers implemented in Matlab (for SVM we used MATLAB's libSVM toolbox [18] whereas for RF, we used Matlab's `TreeBagger` function). The same train/test splits were used for the different classifiers to facilitate paired statistical testing.

RF is an ensemble approach for approximating the optimal decision boundary function between classes. Since it is effectively non-parametric, RF can be a sensible choice if no prior knowledge is available about the underlying model which is the case here. To classify data using RF, we used an ensemble of 100 trees.

The Support Vector Machine (SVM) classifier is a discriminative classifier that creates a separating hyperplane between the classes in a feature space defined by a kernel function. An in-depth description of SVM is given in Section 4.3.1. We use SVM because it has shown to be able to identify ErrNs [136, 137] and also exhibits general state-of-the-art performance in many data classification problems (in particular for large numbers of features and imbalanced

class, as is the case here). For classification using SVM, we used Gaussian (RBF) kernels throughout. The form of the kernel function is given in Section 4.3.1.

As well as being interested in the relative performance of BoD and accelerometer features, we also investigated combining the data sources (BoD and accelerometer) through kernel combination with the SVM and feature concatenation for the RF. In the kernel combination approach, a weighted sum of the two kernel matrices (BoD and accelerometer) is used, with an additional parameter a controlling the influence of each kernel. We explain more about this procedure in Chapter 4.3.1.

When combining Touchstroke and $\mathbf{BoD}$ features into the same classifier, we used a linear combination of kernels with an additional parameter a that controls the influence of each kernel where $0 \leq a \leq 1$ with 0 represents only BoD kernel and 1 representing only Touchstroke kernel. Optimising a allows us to measure the relative contribution of each data modality. We explain more about this procedure in Chapter 4.3.1.

Optimal parameters γ and C were selected via 3-fold cross-validation method on the training set with AUC as selection criteria. When optimising a, we fixed the γ parameter for each kernel to their respective optimal values and searched for the best a and C combination.

6.4 Results

The following results are organised according to the research questions (see Section 6.1.3). Prior to reporting our main findings, we first show the differences between swipes. This is to give an insight into what distinguishable features are contained in the signals. In the first experiment we would like to answer whether hand grip/motion from swipe responses can be used to detect swipe mistake (RQ1). The second experiment answers whether our approach can be generalised (RQ3) whereby in the third experiment we show whether combination of both inputs could lead to an improved error detection (RQ4). To evaluate classification performance, we use Area Under the ROC Curve (AUC), False Accept Rate (FAR) and False Reject Rate (FRR). We use AUC of random classifier as baseline to all classification results, computed by permuting training data labels. See Section 4.6 for a review on these performance evaluation metrics.

6.4.1 Preliminary Analysis

Before performing any classification, we visualised the mean trial of normalised time series across all participants. The time series can be seen in Figure 6.5 for BoD sensors (top three plots) and accelerometer (bottom three). In all plots, the blue lines show the mean (plus and minus standard error) for all correct swipes and the red for all incorrect swipes. Various

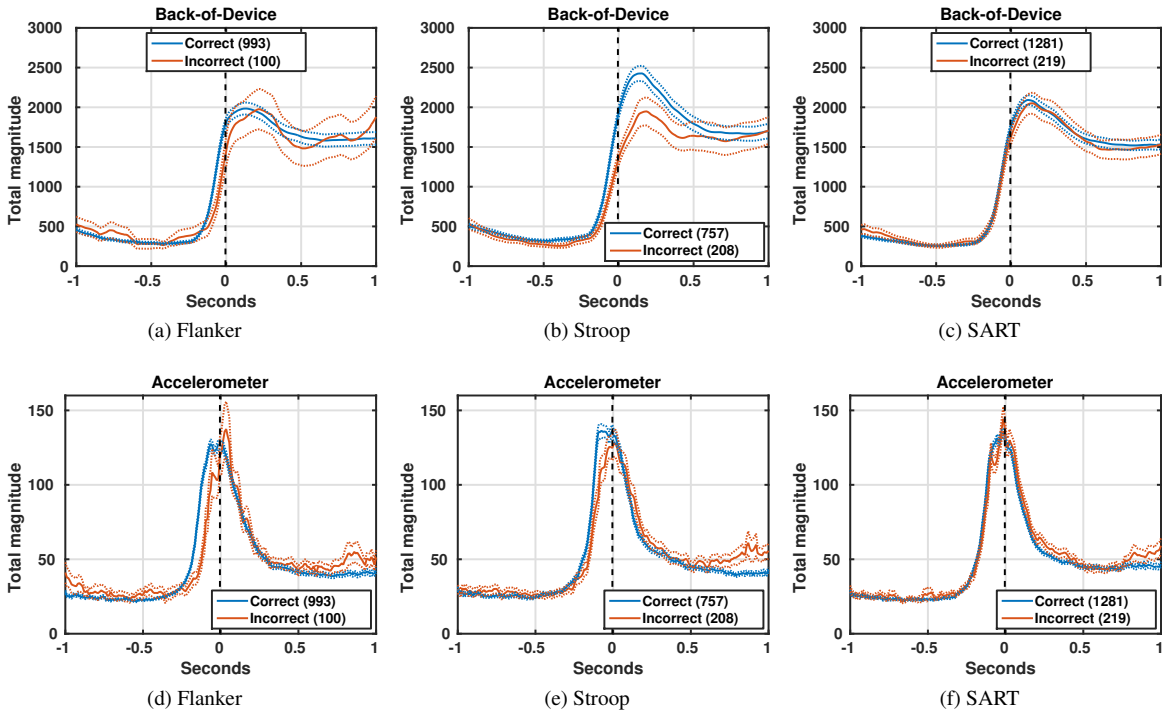


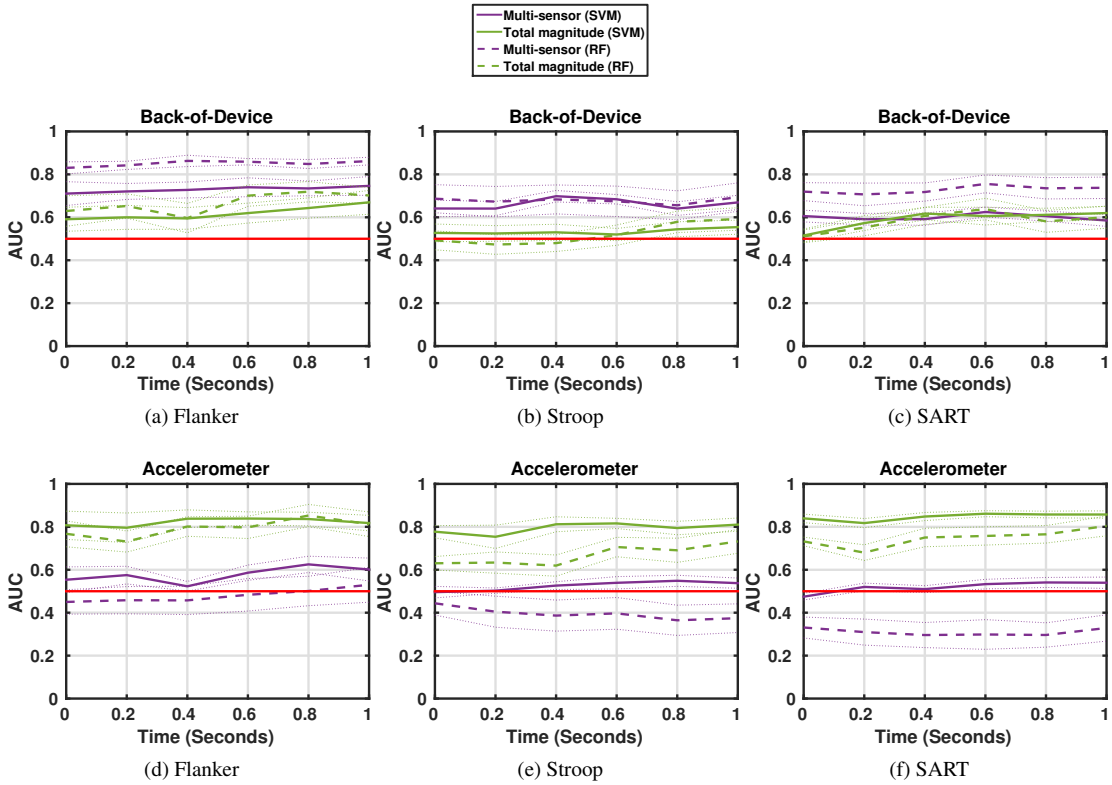
Figure 6.5: Mean total magnitude of signals from the 24 BoD capacitive sensors (a, b, c) and 3-axis built-in accelerometer sensor (d, e, f) from all participants. Blue and red solid lines correspond to correct and incorrect responses respectively. The x axis is the time before and after the notification moment. The y axis is total magnitude of the sensors. The blue and red dashed lines are $\pm$ standard error of the mean total magnitude. The black dashed line is the feedback moment.

interesting features are visible in these plots. Firstly, there appears to be a clear difference in the lines for the Stroop test, suggesting that there are signals that could be classified to detect errors. Secondly, the differences appear to start before feedback (dashed vertical line), suggesting that users know when they have made a mistake. Such changes are not so visible in the other tests (there is a small difference in the accelerometer plot for Flanker but nothing for SART) suggesting that the different tests reveal different physical responses from the participants. Note also that the BoD features for Flanker errors have quite large standard error. This is due to the relatively small number of errors in this test (see Table 6.1).

6.4.2 Experiment 1: Can We Classify Swipe Errors Automatically?

The grand average plots of Figure 6.5 suggest that there are systematic variations in the measured grip and pose when errors are made. To determine whether this could be reliably distinguished automatically, we trained classifiers to recognise error swipes.

Our first approach trained user-specific classifiers. The classifiers were trained using both



(a) Flanker (b) Stroop (c) SART

(d) Flanker (e) Stroop (f) SART

Figure 6.6: Classification results using individual model averaged across all participants for BoD (a, b, c) and accelerometer (d, e, f). Solid and dashed lines correspond to mean AUC from all participants (with ±standard errors) for SVM (green) and random forest (purple) classifiers respectively. Red solid line is the reference baseline.

the total magnitude and multi-sensor features. The results of this experiment are shown in Figure 6.6. It shows that error swipes *can* be classified automatically from the sensors we have, with AUCs well above baseline performance. Here the baseline remains 0.5 because AUC under the ROC is insensitive to class imbalance. There is also significant user variation in classification performance, possibly due to the large variance in numbers of errors made by users.

In all cases, observing longer periods of sensor data did not improve classification, and performance 200ms after feedback was comparable with feedback including subsequent time points. In particular, we chose 200ms to denote time in the actual use-case (e.g. Tinder), where a user is unlikely to be able to correct the swipe manually.

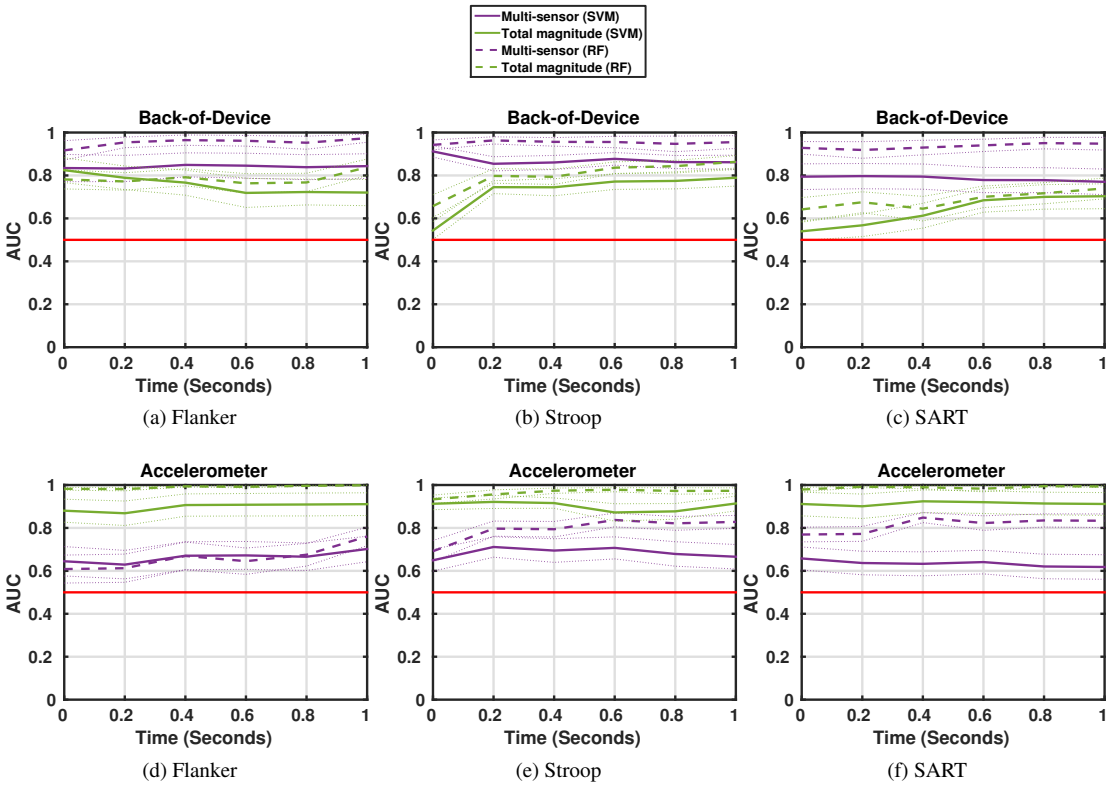
We found that the multi-sensor features (time series of 24 pad values) gave better performance than total magnitude for the BoD sensors. The RF performed better than the SVM for Flanker and SART, giving AUC of 0.80 and 0.70 respectively after 200ms of data. In Stroop however, the performance of RF and SVM is similar, producing classification AUC of about 0.65 after 200ms observing BoD data.

For the accelerometer features, total magnitude performed much better than individual axis

classification, with mean AUC of 0.80 at 200ms in all tests, compared to 0.60 at 200ms for individual axes. This may be due to the device being held at slightly different orientations, leading to big offsets in the individual axis data, but having no effect on the total magnitude of the accelerometer vector. The SVM performs better than RF for accelerometer classification, particularly in Stroop and SART tests.

The RF performance is very poor for Stroop and SART accelerometer classification, possibly due to the very small number of swipe error examples available for these tasks. We required at least 5 training examples of both correct and incorrect responses to train the classifiers. Due to insufficient numbers of error swipe examples, we could not train classifiers for some participants. We used 8 users for Flanker, 15 for Stroop and 18 for SART.

6.4.3 Experiment 2: Can We Build Classifiers That Do Not Require Individually Trained Models?



(a) Flanker (b) Stroop (c) SART

(d) Flanker (e) Stroop (f) SART

Figure 6.7: Classification results using pooled model for BoD (a, b, c) and accelerometer (d, e, f). Solid and dashed lines correspond to AUC (with standard errors) for SVM and random forest classifiers respectively. Solid red line is the reference AUC baseline.

Having to train a specific classifier for each potential user is problematic in real-world use: it requires a potential user to explicitly enroll in the system, online training must be performed, and we must know which user is using the device at all times. A universal, user-insensitive

model would be much more useful. We set out to determine if all participants share common error-related grip or hand motion patterns.

To determine if a general model can be trained to classify error swipes, we constructed classifiers using a leave-one-user-out (LOUO) strategy. Through this strategy, we used a particular participant swipe data as test data and trained the classifiers using swipe data sampled from the other 19 participants. Our results are summarised in Figure 6.7. Surprisingly, the general classification model performs much *better* than the individual model, with AUCs of > 0.9 for RF classification of BoD features 200ms after feedback. This improvement is likely to be due to the increase in quantity of training data available.

Again, we found that the best classification performance for BoD used the multi-sensor features while total magnitude worked best for the accelerometer. We also again found that additional time periods after 200ms did not significantly improve classification performance. For BoD the RF has an AUC > 0.90 at every point of time after the feedback in all tests. This suggests that there is highly discriminative hand grip pattern that happens early during error swipes. For accelerometer, the SVM matches the AUC of RF 100ms after the feedback but drops slightly when more data points is observed by the classifiers particularly at 200ms after the feedback onwards in Stroop and SART tests. In contrast, the AUC using RF increases when more data point is used to build the classifiers.

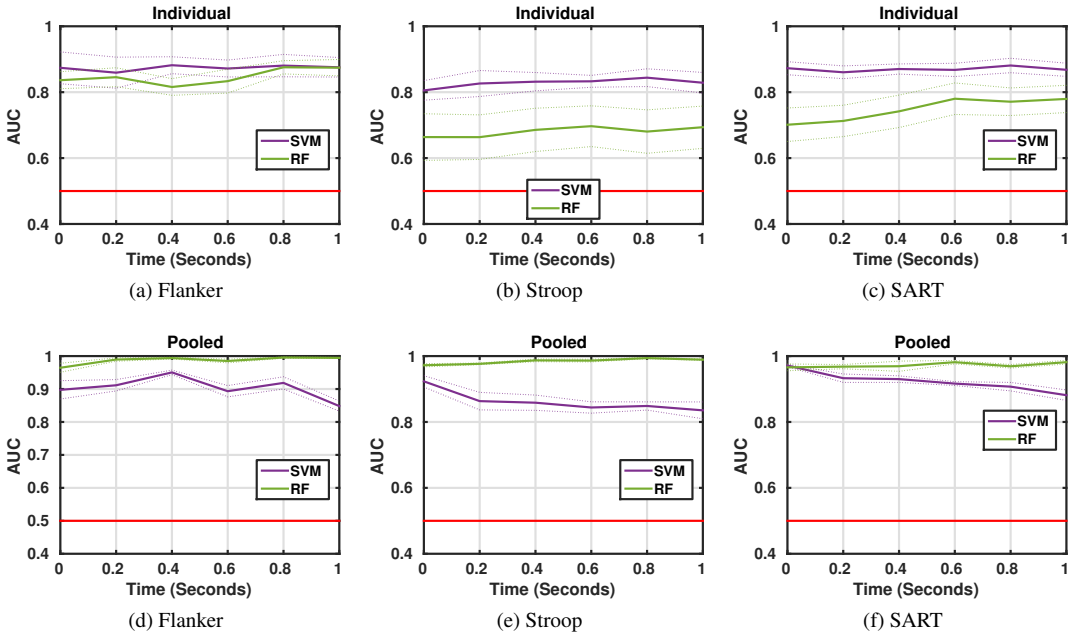
6.4.4 Experiment 3: Can We Improve Performance by Fusing The Accelerometer and BoD Sensors?

Given that we have both the accelerometer and BoD sensors, we wanted to test whether they were providing the same information or if they had separate elements; and if there was separate information, could this be used to improve classification performance. We built composite classifiers using the best performing feature of each input modality: multi-sensor for BoD and total magnitude for accelerometer.

For the SVM, we can use a weighted linear combination of kernels, parameterised by a the kernel weight a (0=accelerometer only to 1=BoD only). For the RF we simply concatenate the feature vectors. The relative influence of each input cannot be assessed with this approach.

Table 6.2 shows the overall performance of the classifiers with the composite features. The random forest has better performance overall, and suffers much less from the class imbalance problems that cause the SVM to have extremely high FRR rates and very low FAR rates.

The results are summarised in Figure 6.8. Composite classifiers give better AUCs than either accelerometer or BoD for both individual and pooled models. The best AUC for individual



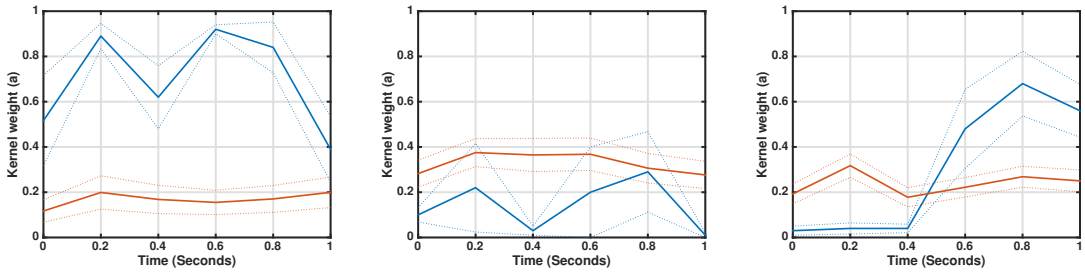
(a) Flanker (b) Stroop (c) SART

(d) Flanker (e) Stroop (f) SART

Figure 6.8: Classification results using combination of BoD and accelerometer kernels for individual (a, b, c) and pooled (d, e, f) models. Solid green and purple lines correspond to AUC for RF and SVM classifier respectively. Dashed lines are the standard errors. Reference AUC baseline is represented by the solid red lines.

	Task	Accuracy (%)	FAR (%)	FRR (%)
SVM	Flanker	78.9	0	2.1
	Stroop	80.7	15.2	23.2
	SART	89.3	0.4	20.9
RF	Flanker	95.8	5.3	3.2
	Stroop	91.8	8.6	7.7
	SART	89.8	10.7	9.8

Table 6.2: The overall performance of the classifiers for the pooled model in terms of accuracy, FAR and FRR at $t = 200$ms with the composite features.



(a) Flanker (b) Stroop (c) SART

Figure 6.9: Optimal mean kernel weight a for composite SVM classifier as a function of time (second) for individual (red) and composite (blue) models.

models uses the SVM classifiers, producing AUC > 0.80 in all tests. For pooled model, the RF classifiers outperform the SVM in all tests, producing AUCs > 0.95 for all time periods.

For the SVM model, we found that the best composite kernel weight a varied with the task: the Flanker test performed better with a high BoD weighting, while SART and Stroop performed better with a lower BoD weighting. As shown in Figure 6.4.4, optimal a was neither 0 nor 1.0, indicating that the classifier found independent information from both sensors.

In order to investigate if the optimal a is statistically significant, we ran a Wilcoxon paired signed-rank test (p–value threshold $= 0.05$) between the balanced composite features ($a = 0.5$) and the individual features for each BoD ($a = 1.0$) and accelerometer ($a = 0$) features at $t = 200$ms following feedback. The null hypothesis is that there is no improvement in performance using composite features.

From the test we found out that there were no significant improvements in all tests for composite features versus accelerometer. However for BoD, the improvement was significant in SART ($p < 0.01$) but not in Flanker and Stroop.

For the individual user models, we found statistically significant improvements for SART and Stroop tests for BoD versus composite ($p < 0.001$ and $p < 0.001$ respectively). Improvement over accelerometer alone was not statistically signficant for any task with the individual models. For the pooled model we found a statistically significant improvement over accelerometer alone for Flanker and Stroop ($p < 0.01$ and $p < 0.05$ respectively).

For the RF model, statistically significant improvements with the composite model were not observed for the individual or pooled models over either BoD or accelerometer. We also compared performance of the SVM and RF classifiers on the composite features, again with a paired Wilcoxon signed-rank test (p–value threshold $= 0.05$). For the pooled models, the RF has significantly better AUC for Flanker, Stroop and SART ($p < 0.05$, $p < 0.01$, $p < 0.01$ respectively). For individual models, the SVM has significantly better AUC for SART ($p < 0.05$), but there is no significant difference for Flanker and Stroop.

6.5 Discussion

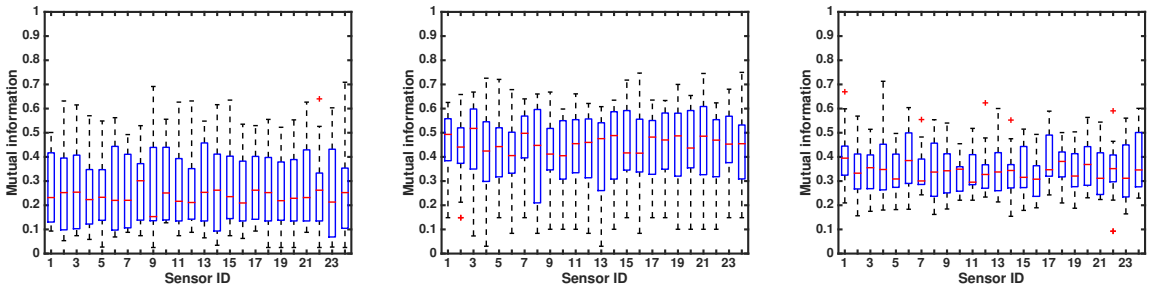
In this work, we show that swipe errors from cognitive tasks performed in a controlled lab setting can be classified automatically using hand grip and movement data. In particular we show that our composite classifiers can classify swipe errors with AUC ≥ 0.80 at 200ms after feedback (Figure 6.8). In terms of reaction time, it is highly unlikely for a user to be able to react to any stimuli within 200ms after they were shown [64]. In fact we show that our technique can achieve similar performance earlier than 200ms, that is at $t = 0$ms and $t = 100$ms. Therefore based on this level of performance, we believe that our technique can

be useful for a system/application that requires instantaneous swipe errors detection (e.g. Tinder), before the user be able to correct themselves. By detecting the error early, it is possible to verify them with the user later.

		Test					
		Back-of-Device			*Accelerometer*		
		Flanker	Stroop	SART	Flanker	Stroop	SART
Train	Flanker	0.95	0.57	0.48	0.98	0.86	0.89
	Stroop	0.60	0.96	0.69	0.97	0.96	0.98
	SART	0.49	0.68	0.92	0.80	0.84	0.99
	Flanker + Stroop + SART	0.92	0.99	0.99	0.99	0.99	0.99

Table 6.3: Cross-task classification AUC using RF classifiers constructed using back-of-device (multi-sensor) and accelerometer (total magnitude) features at $t = 200$ms after the feedback.

To this end, we evaluated the classification performance based on swiping task. It is useful to see if the error signals are indeed task-specific or generic across tasks. To investigate this we ran another experiment by evaluating the classification using mismatched models. As shown in Table 6.3 training on specific *specific* models and testing on mismatched tasks (e.g. training on Flanker and testing on SART) has very good performance for the accelerometer (AUC $0.80 - 0.99$) however weaker for the BOD, possibly due to the high dimensional features with relatively small training sets. Training with a multi-task model significantly improves test performance over training on just the matching task ($p < 0.05$, $H_0 =$ test performance using mismatched task does not lead to improvement). This indicates that it is possible to establish a generic model that can be adopted by various swipe-based interaction that requires swipe error detection.



(a) Flanker (b) Stroop (c) SART

Figure 6.10: Distribution of mutual information for each BoD sensor on per-user basis.

Our prototype phone has 24 back of device sensors. Our study has suggested that the spatial information encoded in these sensors is useful in detecting errors. It would be useful to be able to extract from such data where the sensors should be best positioned to make error predictions. The distinct differences between grip across users makes this impossible with

the current data (we have too few errors per user). Indeed, a preliminary mutual information analysis (Figure 6.5) with our data reveals no clear patterns in the information content of the different sensors. In our studies, we investigated the effect of time on the classification performance. On the whole, we found classification performance to remain fairly constant over time, suggesting that the error information is present very early in the time series.

6.6 Conclusion

In this chapter, we addressed automatic detection of errors (swiping the wrong way) in swipe gesture based interfaces by detecting subtle fluctuations in hand grip and movement. We used BoD capacitive sensors and accelerometer to detect these changes. Three swipe-based cognitive tests based on the classic Flanker, Stroop and and SART tasks were used to induce errors. From the recorded sensor data, we trained classifiers Support Vector Machines and Random Forests which could reliably detect errors shortly after their performance. It should note that, these conclusions are based on this particular small-scale experiment. In summary, we have shown that:

Swipe errors can be detected.
In our controlled lab setting and specific tasks – swipe errors have a distinctive signature that can be automatically classified. AUCs well above baseline were achieved for individual models (Figure 6.6) (e.g. AUC > 0.80 using the SVM on accelerometer data for all tasks).

Swipe errors patterns generalise over users
When we pooled data from multiple users and did not use user-specific models, we were still able to classify swipe errors reliably. In fact, AUC increased from around $0.62 - 0.83$ to $0.91 - 0.99$ with the pooled model. This suggests that classification performance may be limited by the available dataset size rather than inter-user variability.

Specific sensors work best with different transforms.
BoD performance was best with the full 24 sensor time series compared to total magnitude (SVM AUC increased from 0.85 to 0.95 at 200ms). This suggest that there are relevant changes in the pose of the hand, and not just overall grip strength. The accelerometer, in contrast, had better performance with total magnitude than individual axis features. This may be because of variations in orientation which are removed by the magnitude transform.

Composite features are the best.
We found the best classification when combining the accelerometer and the BoD sensors, both with simple concatenation of the feature vectors (RF) and with composite kernels (SVM). This suggests that the accelerometer and BoD sensors contribute independent information about the error signal.

There is little change in classification performance over time.
In all of our results, classification performance reached a peak by 200ms after feedback and did not vary significantly after that. Even at feedback time ($t = 0$), before the user has had a chance to react to the feedback we are able to classify the error signal. This suggests the error-related movements happen early, and may anticipate the feedback (users are responding to internal knowledge of the error they have made). This also indicates that observing longer periods of sensor data will not change detection performance. Despite of this, it should note that this is purely based on a small scale experiment that we conducted, thus further experiments may be needed to conclude this finding.

Random forests generally perform better than the SVM.
The random forest classifier was more effective for all of the pooled models (although performed very poorly in the individual models where training data was sparse). As can be seen in Table 6.2 the SVM did not deal well with the imbalanced classes and led to very high FRR rates. The random forest had better overall performance for the pooled models, and a much more balanced FAR/FRR result.

Performance was similar for SART, Stroop and Flanker tasks.
We were able to classify errors with a similar accuracy (Table 6.2) for errors made in SART, Stroop and Flanker tasks, suggesting that the results are indeed capturing a general error signal, and not simply a task-specific anomaly.

In summary, our results suggest that hand grip and hand movement from BoD sensors and accelerometer can be used to reliably and rapidly detect cognitive errors in swipe-based interfaces. While back-of-device sensing is not yet mainstream, some commercial devices (e.g. the Doogee DG800 or the Oppo N1) already provide BoD touch sensors that may have sufficient resolution to capture grip modulations. Although our experimental models are specific to the cognitive tests that we used, many mobile tasks have very similar components: a mentally demanding task with simple swipe gestures as input. Our results are both practically relevant in developing forgiving interfaces that can recover from errors gracefully, and suggest that there is much more to be explored in error-related micro-movements in touch screen gesture interfaces.

These results form a concrete contribution to building a rich, multi-sensor mobile phone interaction error detection system. While hand grip and hand movement alone may be insufficient to perform precise detection, it can form part of an array of contributing virtual sensors in a hybrid in recognising a users true intention.

Chapter 7

Implicit User Identity Prediction

The work presented in this chapter will be published in TOCHI journal. The co-authors are John Williamson[1] *and Simon Rogers*[2]

Summary
This chapter discusses the use of back-of-device (BoD) and accelerometer sensors as an input to predict users' identity. We present two methods that perform prediction during: 1) phone pick-up action and 2) touchscreen interaction by replicating previous accelerometer [30] and touchscreen [40] techniques and compare the results with our technique that uses hand grip. We demonstrate that the identity of users can be predicted significantly better than the previous techniques using BoD hand grip sensing in different phone's usage scenarios. Furthermore, we also show that predictions can be improved when both input modalities are combined. We also show the optimal location of the BoD sensors to sense salient hand grip information for identification purposes.

7.1 Introduction

Implicit authentication (IA) methods on mobile devices have been developing rapidly in recent years. The methods which use observed user behaviour such as location, phone activity and body motion offer a spectrum of usability enhancements, from subtle adaptations of the user interface to specific users [107] to complementing existing PIN/password security [58]. Their application has shown to improve both usability and security of devices when compared to explicit method [58]. However, despite being advocated as more usable than explicit authentication schemes, false rejects and significant delay in detection with implicit authentication can be a source of annoyance and increased security concerns [119, 69].

[1]School of Computing Science, University of Glasgow, UK (JohnH.Williamson@glasgow.ac.uk)
[2]School of Computing Science, University of Glasgow, UK (Simon.Rogers@glasgow.ac.uk)

In contrast to the obtrusive nature of standard biometric systems, the majority of IA techniques employ 'soft' or behavioural biometrics to continuously accumulate evidence about the user over time and transparently recognise and validate their identity [32]. This is typically done by leveraging standard smartphone sensors to acquire users' biometrics information. For instance, keystroke analysis uses touchscreen keyboard input to distinguish user typing behaviour [35, 109], while user gait pattern [36] and phone answering pick-up motion [30] can be recognised using the built-in inertial sensors. A recent paper demonstrated the utility of standard touch interactions to identify users [40], whilst subtle hand micro-movement and dynamics captured by inertial and touchscreen sensors can also be used to continuously verify users [125]. An interesting technique based on Transient Evoked Otoacoustic Emission (TEOAE) [66] uses built-in ear-phone and microphones to identify user through their ear canal acoustic response patterns [82]. Beyond smartphones, sensors are also found in most smartwatches and activity trackers, suggesting that IA could be the future authentication method for wearables [50].

The use of hand grip as an input for identification purpose is not new and has been well explored. For instance, Kauffman et al. designed and evaluated a user-verification system for a smart gun which is based on grip-pattern recognition [120]. Their technique uses likelihood-ratio (LRC) and support vector machine (SVM) classifiers to verify user grip pressure images acquired from the piezoresistive sensors in the grip of the gun. The reported error equal rate (EER) – the rate at which false acceptance and false rejection rates are equal, are approximated at 15.2% (LRC) and 10.6% (SVM). These rates are still above the minimum requirement (expected gun failure rate is 1 in 10000 or 10^{-4} [46]), making its implementation in the real gun a controversial.

Besides the smart gun, researchers also used hand grip on steering wheels in order to recognise the identity of the driver [22]. To sense users' hand grip pressure, they mounted a pressure-sensitive pad on the steering wheel. Whilst their technique achieved mean acceptance rates of 85.4%, the mean false acceptance rates on the other hand was not reported. In the context of identification, false acceptance rate corresponds to the likelihood that the system will accept an access attempt by an unauthorised user while false reject rate corresponds to the likelihood that the system will reject an access attempt by an authorised user. Their technique is a promising biometric technology which can be adapted in smart car design such as personalised driving.

The results from smart gun and driver recognition studies indicate that handgrip contains valuable information for identity verification. Therefore in this chapter, we investigate the utility of phone grip as an input modality to predict user identity. In contrast to previous work, we investigate whether back-of-device (BoD) capacitive sensors can be reliably used to distinguish user handgrip patterns implicitly during interaction. As a basis for comparison, we compare our results to the existing implicit methods based on inertial and touch

capacitive sensors. To do so, we replicated the work by Frank et al. [40] and Conti et al. [30] and augmented their methods with BoD sensing. Conti et al. used hand movement when answering and placing a phone call to identify users whereas Frank et al. predict user identity by observing their unique touch behaviour on the touchscreen.

Apart from previous work, there are a number of factors that motivate us to use mobile device grip via BoD sensing to identify users. First, BoD sensing does not require any explicit interactions with the mobile device, making it less intrusive than other implicit front-of-device (FoD) techniques. Second, the sensors are in continuous contact whenever a user uses a mobile device, giving a reliable, always available channel for identification. Third, BoD interaction reduces the chance of *shoulder-surfing* [35, 114] attacks, where the dynamics of FoD screen touches are emulated by an impostor. In comparison to FoD interaction which can be susceptible to forgery [3, 152], the BoD is less accessible and depends more on specific user physiology.

7.2 Research Questions

In order to investigate whether hand grip can be used as an identity predictor, we compare our method with existing methods by augmenting the experiments used in previous methods with BoD input. From the experiments we attempt to answer the following research questions:

RQ1 ***Can user identity be determined from their grip patterns?***
The main goal of this chapter is to measure the extent to which user identiy can be predicted better than random (guessing) from BoD sensor inputs. Using a standard machine learning approach, we train classifiers to predict identity from user hand grip.

RQ2 ***Does the BoD method perform better than the existing methods?***
Specifically, we are interested to compare our approach to existing approaches based on touchscreen and accelerometer inputs as identity predictor. To compare this, we replicated previous techniques and augmented them with BoD sensing.

RQ3 ***Is there any performance gain when combining sensor data?***
Apart from comparing prediction performance between sensor inputs, we also are interested to investigate whether the performance can be improved by fusing together sensor data from the touchscreen and BoD. In particular, we evaluate the fusion by finding the optimal weight from each input in order to achieve highest prediction accuracy.

RQ4 ***Can we minimise the amount of data required to predict user identity?***
We would like to test the influence of the amount of data in prediction performance.

To do this, we run predictions at different periods during the course of interaction. By doing this we can evaluate and estimate how much data the classifier needs to observe in order make accurate prediction.

RQ5 ***Do we need all* BoD *sensors to make predictions?***
Although we use 24 BoD capacitive sensors in all experiments, we investigate whether we need all sensors to make predictions on user identity. By investigating this, we are able to estimate the minimum number of sensors required as well as their locations at the back of the device.

The contribution of this chapter is a study investigating the potential use of hand grip information for IA in which we achieve using a prototype device that can sense users' hand grip. We also demonstrate the optimal sensor locations to capture salient identity information from the sensors, and propose a minimum set of sensors for effective identification. We also provide insights into the performance gain from combining both front and BoD modalities. Finally, we discuss the various tradeoffs in using BoD to identify users.

These results form a concrete contribution to building a rich, multi-sensor mobile phone identification system. While grip alone may be insufficient to perform secure authentication, it can form part of an array of contributing virtual sensors in a hybrid continuous authentication system.

7.3 Experiment Replication

The following sections describe the experiments that we used to answer the research questions established in Section 7.2. The first experiment serves as a preliminary investigation on the utility of hand grip as identity predictor where the second experiment is a more elaborate investigation to evaluate to which extent that hand grip can be used to predict user identity.

As mentioned previously, this chapter involves replicating previous work and augmenting them using BoD sensing. In other words, we repeat the experiments using our prototype devices with the purpose to compare prediction performance between their technique and ours.

7.4 Experiment 1

Mind How You Answer Me! : Transparently Authenticating the User of a Smartphone when Answering or Placing a Call [30]

In this work, Conti et al. propose an IA method that offers transparency by identifying if the user that is answering (or placing) a call is the authorised one. In particular, they investigated if a user can be authenticated just by using the hand movement that the user performs – that is the moment when the user presses "start" (to initiate a call), until the user takes the phone to the ear. They refer to this movement as a pattern which can be treated as a biometric feature. In this work they used a Google Dev 1 phone which is based on HTC Dream to capture user's hand movement and phone orientation using the internal accelerometer and orientation sensors.

They wrote an application to log accelerometer (acceleration on x, y and z-axis) and orientation (*pitch*, *roll* and *yaw*) sensor values and asked 10 participants to use this application to collect data. In particular, participants were asked to perform the phone pick-up action when using this application, which begins when the participants press "start" and ends when the phone reaches participants' ear. However, no information on the number of times the phone pick-up action is performed by each participant is given. In order to recognise users, they applied a Dynamic Time Warping (DTW) algorithm on both sets of input data (accelerometer and orientation) for each axis/orientation to compare the similarity of user's sensor timeseries. This is done by finding sum of the shortest distance of every point between user's current and stored timeseries. Zero sum denotes that both timeseries are identical. A threshold value is defined for each sensor instance to control the similarity decision.

To evaluate performance of their technique, they used two performance metrics: 1) False Accept Rate (FAR) which is the percentage of successful accesses by unauthorised users and 2) False Reject Rate (FRR) which is the percentage of rejected accesses by authorised users. The decision threshold was varied to find a balance between FAR and FRR. Using the accelerometer sensor, they achieved FAR and FRR of 20.66% and 12.88% respectively while using the orientation sensor, they achieved FAR and FRR of 9.33% and 4.44%. However, they did not mention which axis/orientation that these results correspond to.

From this study, they show that there are sufficient differences between different users, such that the movement can be effectively used for identification. In this way, as soon as a call is answered (or placed), the phone can promptly evaluate if the user is authorised to perform this action, and block the system in case of non authorised users.

We recreated Conti et al.'s experiment [30] by adding $\mathbf{BoD}$ hand grip information. In particular, we are interested in investigating whether dynamic sensing of grip contact can aid in the identification of users during the call answering process (RQ1). Furthermore, this also allows us to compare performance between hand grip and hand movement methods (RQ2) as well as to investigate if performance gain can be achieved if both inputs are used to build the identity predictor (RQ3).

In order to capture hand grip contact we developed a prototype system which consists of a

Sony Xperia Mini Pro (SK7i) smartphone connected via Bluetooth to a SHAKE SK7 sensor pack[3] that interfaces an external array of 24 capacitive sensors that detect proximity of the hand to the rear and sides of the case. Apart from **BoD** sensors, the SK7 is also equipped with various other sensors, including accelerometer, gyroscope and magnetometer. The total size of this prototype is 25 mm (H) x 150 mm (L) x 65 mm (W). For more information about this prototype, please refer to Section 3.2.

The 24 capacitive sensor values (with 8 bit dynamic range) and 3 accelerometer values (in the $\pm$2g range, with 12 bit dynamic range) were transmitted from the SK7 to the phone via Bluetooth with a sample rate of 50 Hz. An application was developed on the phone to perform the experiment and record the data for subsequent offline analysis.

7.4.1 User Study

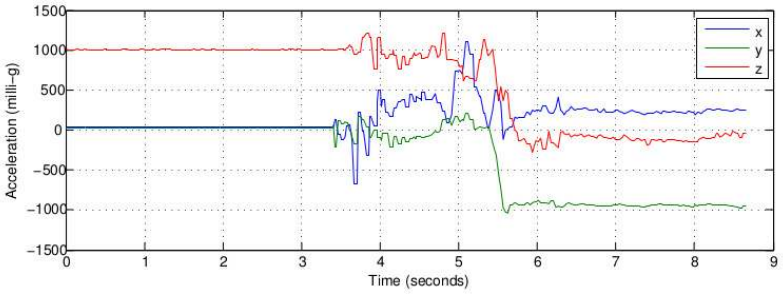
For this study, 12 participants consisted of 6 males and 6 females, aged between 25 and 35 years old (mean=26.91, sd=2.15) were recruited locally after formal ethical approval been granted on 6^{th} of May 2014. Each participant recorded data over 6 separate sessions, 3 with their left hand and 3 with their right. Each left/right session was separated by a 3-minute break to minimise repetition effects – a condition where users becoming skilled after making similar action repetitively [83]. Therefore by separating the task into sessions, we are not capturing only a single hand grip/movement pattern but a range of plausible hand grip patterns.

To acquire sensor data, we developed an Android application on the Sony Xperia Mini Pro smartphone. The application communicates with another application that we wrote on a PC to trigger a phone call on the smartphone. This allows us to coordinate the data acquisition process without the need to rely on the real GSM network. The BoD sensors were calibrated prior to every data acquisition session. This was done using simple two-point calibration of the capacitive sensors, that is by measuring untouched and touched each sensor's capacitance values.

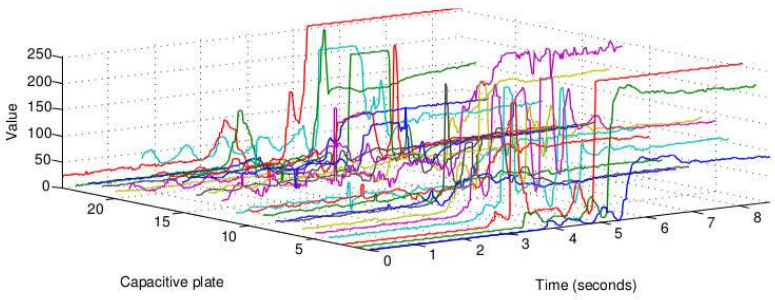
Each session required the participants to perform 10 answering actions while seated in front of a desk. Each action started with the phone ringing. On hearing this, the user picked up the phone (with whichever hand they were using in this session) and lifted the phone to the same ear (i.e. if a right-hand session, the phone was lifted to the right ear). They were then asked to hold the phone there for at least 3 seconds. Data logging started when the phone rang and finished when the phone vibrated. From each call answering action, we recorded timestamps, acceleration along x, y and z-axis and 24 BoD capacitive sensor values. All data

[3] `http://code.google.com/p/shake-drivers/`

were stored in the external storage of the phone for subsequent offline analysis. Examples of recorded signals for one call answering action can be seen in Figure 7.1.



(a) Accelerometer



(b) **BoD**

Figure 7.1: Signals from accelerometer and **BoD** for one phone answering action using right hand selected randomly from one user.

7.4.2 Analysis Methods

Similar to the approach proposed by Conti et al. [30], we used classifiers to identify to which group (authorised/non-authorised) the observation (phone answering action) belongs to. This represents a standard setting where some training data from a particular user would be compared with a pooled set of training data from other users.

We began by pre-processing our data by converting the time series into fixed length feature vectors. This was done by first removing the empty signal at the beginning (data logged between the phone ringing and the users' first touch) by deleting all data until the total output from the capacitive sensors reached 150 which is approximately 3 seconds after the trial starts (flat section of the timeseries in Figure 7.1).

The remaining data for each sensor was then split into K equal length blocks and the mean value of the sensor within this block was extracted as a feature. For example, for $K = 6$,

each sensor time series would be converted into $K = 6$ mean values, resulting in a total of $24 \times 6 = 144$ capacitive features and $3 \times 6 = 18$ accelerometer features. All features were normalised to have mean zero and standard deviation one prior to the next step of analysis.

We used a Support Vector Machine (SVM) classifiers in this work. The decision on using SVM classifier is justified by their previous use in this area [70, 40] and their general state-of-the-art performance for data classification. Furthermore, SVM is one of the best classifiers which can handle high–dimensional data as in the case here. An in-depth discussion on SVM classifier is given in Section 4.3. We used Gaussian (RBF) kernel functions throughout.

Because of the large discrepancy in number of features between the capacitive and accelerometer data, we built separate kernels for each. Putting them into the same kernel would make it difficult to assess the relative contribution to performance from each data type, and would also possibly result in the influence of the accelerometer data being drowned out by the larger number of capacitance values. When using both data types, the kernels were combined additively by applying kernel weight a to each kernel where $0 \leq a \leq 1$ with 0 representing only the $\mathbf{BoD}$ kernel and 1 representing only the accelerometer kernel. Optimising a allows us to measure the relative contribution of each data modality. We explain this technique in more detail in Section 4.3.1.

7.4.3 Results

We begin by reporting the performance of the classifier in distinguishing between a particular user and data from all other users. To identify a user, we used a One-vs-All (OVA) classification strategy where we treat data from a particular user (user of interest) as belonging to the positive class and data from the other 11 users as belonging to the negative class. To train the classifiers, we used 20 samples for the user of interest taken from the first two sessions and 80 random samples from other users also from the first two sessions. To test, we used 10 samples from the user of interest from session 3 and 80 random samples from session 3 of other users. This is to reflect the typical phone use case where calibration data is collected at some point (sessions 1 and 2) and then the user uses the phone sometime in the future (session 3).

One-vs-All

In all SVMs, the margin parameter C was set to 10^3 (making this a hard-margin SVM [123]). The kernel parameter γ was set through a Leave-One-Out Cross Validation (LOOCV) procedure on data for each user (from the first two sessions of a particular hand) and 20 random samples drawn from the other eleven users (with the same hand). We use area under the ROC curve (AUC), accuracy and false accept rate (FAR) which is the proportion of misclassifying

	Participants												
Metrics	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**Mean**
AUC													
$Right_{cap}$	0.96	1.00	0.88	1.00	0.96	0.96	1.00	1.00	0.94	1.00	1.00	1.00	**0.97**
$Left_{cap}$	1.00	1.00	1.00	1.00	1.00	0.95	1.00	0.91	0.95	1.00	1.00	1.00	**0.98**
$Right_{acc}$	0.95	1.00	0.96	1.00	1.00	0.95	1.00	0.87	0.89	1.00	0.98	0.99	**0.97**
$Left_{acc}$	0.71	1.00	1.00	0.83	1.00	0.95	0.99	0.90	0.94	0.95	1.00	1.00	**0.94**
Accuracy													
$Right_{cap}$	0.97	1.00	0.93	0.97	0.99	0.93	0.98	1.00	0.90	1.00	1.00	1.00	**0.97**
$Left_{cap}$	1.00	1.00	0.97	0.99	1.00	0.92	1.00	0.99	0.93	0.98	1.00	1.00	**0.98**
$Right_{acc}$	0.94	0.88	0.98	0.99	0.99	0.84	0.99	0.83	0.83	1.00	0.96	0.91	**0.93**
$Left_{acc}$	0.66	1.00	0.97	0.84	1.00	0.89	0.98	0.87	0.88	0.89	1.00	1.00	**0.91**
FRR													
$Right_{cap}$	0.30	0.00	0.20	0.00	0.10	0.10	0.00	0.40	0.30	0.00	0.10	0.00	**0.11**
$Left_{cap}$	0.00	0.00	0.20	0.00	0.00	0.00	0.00	0.10	0.00	0.20	0.00	0.00	**0.04**
$Right_{acc}$	0.10	0.00	0.20	0.00	0.10	0.10	0.00	0.40	0.30	0.00	0.10	0.00	**0.93**
$Left_{acc}$	0.30	0.00	0.00	0.70	0.00	0.00	0.10	0.10	0.10	0.50	0.00	0.00	**0.15**
FAR													
$Right_{cap}$	0.00	0.00	0.05	0.04	0.00	0.07	0.00	0.00	0.11	0.00	0.00	0.00	**0.02**
$Left_{cap}$	0.00	0.00	0.01	0.01	0.00	0.09	0.00	0.00	0.07	0.00	0.00	0.00	**0.02**
$Right_{acc}$	0.05	0.14	0.00	0.01	0.00	0.16	0.01	0.14	0.15	0.00	0.04	0.10	**0.07**
$Left_{acc}$	0.35	0.00	0.04	0.09	0.00	0.12	0.01	0.14	0.12	0.06	0.00	0.00	**0.08**

Table 7.1: Comparison of performance for left and right hands with capacitive sensors alone, and accelerometer alone. The performance metrics are AUC, accuracy, false reject rate (FRR), false accept rate (FAR).

a sample from another user as coming from the user of interest, and false reject rate (FRR) which is the proportion of misclassifying a sample from user of interest as coming from another user. See Section 4.6 for a review on these performance measures. To compute statistical significance of results, we use the non-parametric Wilcoxon Signed Rank Test with threshold of 0.05. In all classifications, $K = 6$ blocks were used.

Classification results for each user are shown in Table 7.1. From the table, we see that performance is typically high (AUC > 0.9). In general the capacitive models slightly outperform those trained on accelerometer data – indicated by higher accuracy and a lower number of false rejects and false accepts.

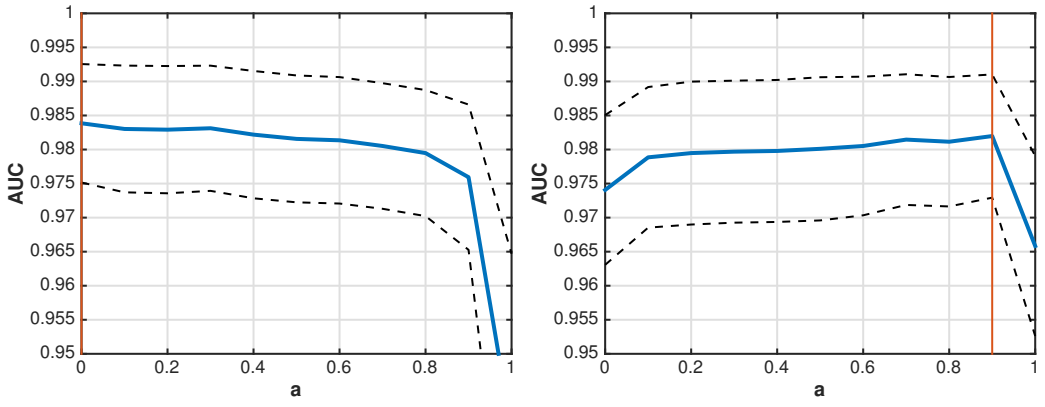
Composite Input

To investigate whether combining two sensor inputs could improve the performance of the individual sensor spaces we used a composite kernel as described previously. Figure 7.2 shows classification AUC as a function of a, the kernel weighting parameter. Note that $a = 0$ corresponds to just capacitive sensing and $a = 1$ to just accelerometer sensing. For left hand the curve decreases gradually as a increases up to $a = 0.9$ before decreasing as a approaches 1. For right hand, the AUC curve increases gradually from $a = 0$ to $a = 0.1$ and gradually increasing up to $a = 0.9$ before the AUC drops when the kernel contains only accelerometer data ($a = 1$). Whilst sensor combination for left hand shows no performance improvement (highest AUC is given by $a = 0$), the right hand however shows that the gain can be achieved through sensor combination, however the gain is not statistically significant.

From the results, it suggests that classifiers constructed using combination of capacitive and accelerometer data could give better performance, particularly when using right hand data (optimal $a = 0.9$). This indicates that the capacitive and accelerometer data contribute independent information about user identity.

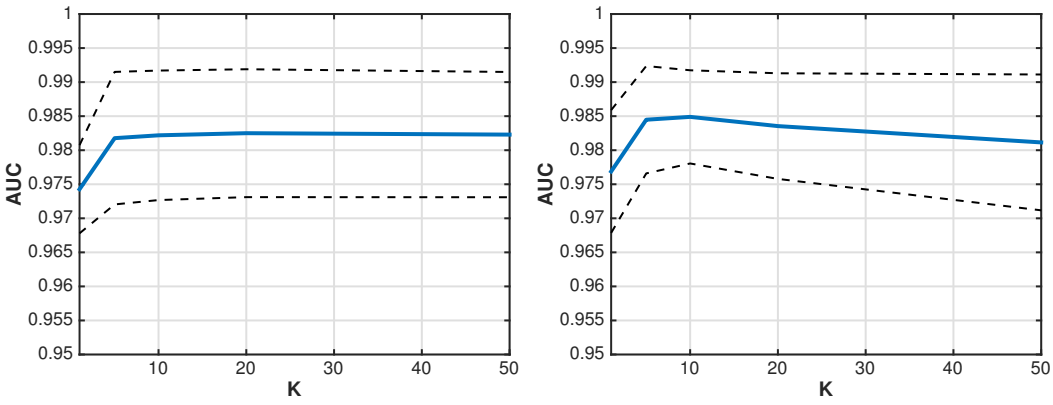
Feature Dimension

Finally, in Figure 7.3 we show how the classification performance varies as we increase the number of blocks (K) which is the fixed length segmentation of the timeseries where the feature is extracted, from 1 to 50. This is based on combining both data types, with the optimal value of a from Figure 7.2. The performance increases rapidly and then plateaus before dropping slowly as K is increased. This is to be expected – simply taking the mean for each sensor over time ($K = 1$) is unlikely to be optimal. Similarly, increasing K above some value is likely to result in different parts of the answering action ending up in different blocks (and hence different features) for examples from the same person, causing a slight decrease in performance.



(a) Left (b) Right

Figure 7.2: Classification performance (AUC) averaged across users including ± standard error as the kernel weighting parameter is varied from just capacitive data (left, $a = 0$) to just accelerometer data (right, $a = 1$). Solid vertical red lines correspond to the optimal a value.



(a) Left (b) Right

Figure 7.3: Classification performance (AUC) averaged across users including ± standard error, as the number of blocks is varied from static ($K = 1$) to highly dynamic ($K = 50$). Performance plateaus appears at $K = 6$ for both hands.

7.4.4 Discussion

Although we managed to achieve mean classification AUC and accuracy of all above 0.9, it should be noted that the sample size here (12 participants) is too small for us to be able to make comments as to how this technique might work in an actual implicit identification system. Nevertheless, it does allow us to compare classification performance between accelerometer and capacitive sensors.

In One-v-All classification setting, it shows that users can be identified with similar level of performance using hand grip and motion data. This suggests that both sensors can be used interchangeably to identify users' unique hand grip and motion patterns from the phone pick-up action. This can be useful in some phone answering scenarios such as answering

while walking/on the move where hand grip is more consistent compared to hand motion. Moreover the results also show that the performance for right and left hands are nearly similar suggesting that this technique may generalise for right and left handed users.

The phone answering tasks that we used in this study were constrained to only two pick-up motions: picking-up with the right hand and bringing it to the right ear and picking-up with the left hand and bringing it to the left ear. It is therefore interesting to explore how other possible pick-up motions (e.g. picking up the phone using left hand and bringing it to the right ear) would impact the classification performance.

Despite improving performance when using composite classifiers, the mean improvement however was marginal and not statistically significant. This indicates that the addition of hand grip or hand motion information to the kernel may not necessary lead to an improvement. Nevertheless, from Figure 7.2, it is clear that single kernel classifiers could already produce very high classification AUC using either hand grip or hand motion data and thus composite classifiers might not be needed.

7.5 Experiment 2

Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication [40]

In this study, Frank et al. lay foundational work for continuous authentication schemes that uses the touchscreen as an input. They investigated if it is possible to authenticate users while they perform basic navigation steps on a touchscreen device, without any dedicated explicit security action that requires attention from the user.

In order to investigate this authentication scheme, they designed two experiments involving touchscreen interaction. 41 participants were recruited to participate in the experiments. They consider two common interactions on the touchscreen for this study: 1) touch–scroll and 2) touch–swipe. For touch-scroll, participants were required to read three documents on an Android mobile phone. To ensure natural interaction with the touchscreen, participants were not told that their touch behaviours are being assessed. Furthermore, participants also were asked to set aside the phone to fill out a sheet with multiple-choice questions about the document. This allows them to provoke the participants to randomly pick up the phone, possibly causing the participants to hold it in another way. As a result, interaction with each document can be regarded as a new session. For touch–swipe, the participants were asked to spot differences in pairs of similar images. They stopped the participant after approximately two minutes, and repeated this procedure for the second image pair. No multiple-choice questions were given in between. One week later, a follow-up study was conducted with only

one document and one image comparison. After finishing all experiments, the participants were explained the true nature of the study.

An application was developed to record touch behaviour from the experimental reading and spot–the–differences tasks. For the first task, the application contained links to documents about *Wind*, *Tulip Mania* and *Yosemite National Park*. All of them were excerpts from *featured* Wikipedia articles. The images used to spot differences are publicly available spot-the-difference comics. The participants can see one image on one panel, and two screen–sizes away–separated by a black panel–is the second image. This means that participants need at least two strokes to get from one image to the other. They however were free to go back and forth as often as they wanted. During the experiments, the phones recorded the participants touch data sampled with a variable frequency. The touch data consists of several raw features: an event code (e.g., finger up, finger down, finger move, multitouch), the absolute event time in milliseconds, and the device orientation. For each touch they recorded its x and y–coordinates, its pressure on the screen, the area of the screen covered by the finger, the finger orientation with respect to screen and the screen orientation.

They divided the recorded touch data into individual strokes. Strokes with too small displacement were considered single clicks and therefore discarded. However they did not specifically mention the exact minimum displacement for a stroke to be considered as a single click. From each stroke 31 features (Touchalytics) were extracted. They measured the informativeness of each feature with respect to the user's touch behaviours by computing relative mutual information between the feature and user ID. Touchalytics features are listed in Table 7.2. From the table, the five most informative single features for spot-the-differences experiment are 1) area covered by fingertip, 2) 20%-percentile of the stroke velocity, 3) fingertip pressure on screen, 4) the direction of the stroke and 5) locations of the end-points of the trajectory.

Two classifiers were chosen to classify strokes: 1) k-nearest neighbours (kNN) and 2) Support Vector Machine (SVM). For kNN they used Euclidean distance with paramater k selected from all odd numbers between 1 and 7 via cross-validation on the training data. For SVM they used a Gaussian radial-basis function as the kernel. Two relevant parameters γ and C were selected via five-fold cross-validation on the training data. The classifiers were trained according to three application scenarios:

1. ***Interweek*** – In this scenario, the classifiers were trained using data collected from the first round of the experiment (enrollment) and tested against data collected the week after. (authentication).

2. ***Intersession*** – Classifiers in this scenario were trained using data from the first and second session and tested against the third session, collected on the same day (first round of the experiment).

3. ***Short-term*** – To emulate this scenario, the classifiers were trained using data drawn randomly from all available sessions.

The authors used Error-Equal rate (EER) when evaluating classification performance. This is an error rate at which both false acceptance rate (FAR) and false reject rate (FRR) are equal. Further explanation of these measures are given in Section 4.6. The lower this rate the more accurate the classification. In general the median EER across all scenarios ranges from $0\% - 4\%$ for both k–NN and SVM, for both scrolling (reading) and swiping (spot-the-differences) classifiers. For intrasession and intersession, the median EERs are approximately 0% and 4% respectively for both scrolling and swiping using both classifiers. Interweek median EER for scrolling is approximately 0% whereby for swiping median EER of approximately 2% and 3% are achieved for kNN and SVM respectively. The results show that the EER increases with increasing temporal distance to the training phase (enrolment). For instance, with exception to scrolling interweek, the lowest EER is achieved in intrasession (shortest) followed by intersession (medium) and interweek (longest). From the results also it shows that lower EER is achieved when using SVM to classify strokes from scrolling and swiping tasks.

7.5.1 Back-of-Device Sensing Augmentation

The purpose of this study is to identify whether features from capacitive $\mathbf{BoD}$ sensing can be used to identify users, and to compare performance with Touchalytics [40] methods. In particular we are interested to see if back-of-device hand grip contains sufficient information to reliably discriminate a particular user from a group of users. In practice, this reflects a typical single-user phone use-case where the phone attempts to identify whether or not the current user is the phones owner.

To do so, we replicated the Touchalytics study as a directly comparable baseline, extended their experimental method to $\mathbf{BoD}$ sensing, and developed classifiers for the $\mathbf{BoD}$ data. We evaluated the performance of both of these methods, and the combination of the two, and estimated bounds on the minimal set of sensors required, and the minimum calibration/training time.

7.5.2 Experimental Setup

As mentioned previously, we used a custom $\mathbf{BoD}$ capacitive sensor prototype. The prototype is based around a Nokia N9, which we modified to include around device sensing to sense hand grip contact. Details of this prototype is given in Chapter 3.3. In contrast to tactile sensing used by the grip-based user interface in [49], we opted to use capacitive sensing

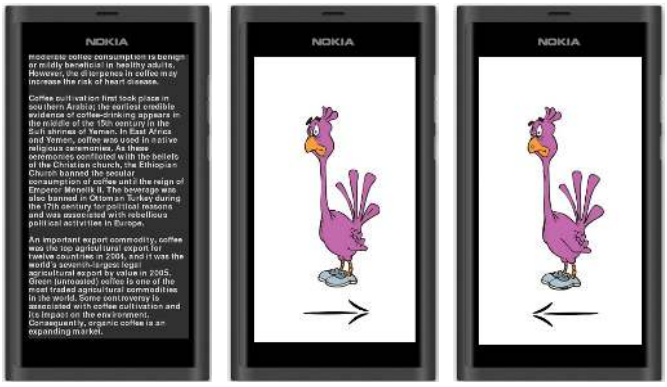
MI rank	Feature	Description
1	Mid–stroke area covered	Area covered by the fingertip.
2	20% percentile pairwise velocity	20% percentile of the stroke velocity.
3	Mid–stroke pressure	Pressure applied by the fingertip during the stroke.
4	Direction of end–to–end line	Direction of a straight line connecting the start and stop of the stroke.
5	Stop x	x coordinate at the end of the stroke.
6	Start x	x coordinate at the start of the stroke.
7	Average direction	Average direction of the stroke.
8	Start y	y coordinate at the start of the stroke.
9	Average velocity	Average velocity of the stroke.
10	Stop y	y coordinate at the end of the stroke.
11	Stroke duration (ms)	Time taken from *touch–down* to *touch–up*.
12	Direct end–to–end distance	Length of a straight line connecting the start and stop coordinates of the stroke.
13	Length of trajectory	Length of the stroke.
14	80% percentile pairwise velocity	80% percentile of the stroke velocity.
15	Median velocity at last 3 points	self-explanatory
16	50% percentile pairwise velocity	50% percentile of the stroke velocity.
17	20% percentile pairwise acceleration	20% percentile of the stroke acceleration.
18	Ratio end-to-end distance and length of trajectory	Proportion of the straight line connecting the start and stop of the stroke and stroke length.
19	Largest deviation from end-to-end line	Furthest distance of a point in the stroke trajectory from the end–to–end line.
20	80% percentile pairwise acceleration	20% percentile of the stroke acceleration.
21	Mean resultant length	Quantities of how directed the stroke is [86].
22	Median acceleration at first 5 last points	self-explanatory.
23	50% percentile deviation from end-to-end line	50% percentile stroke deviation from end-to-end line.
24	Inter-stroke time	Time between two consecutive strokes.
25	80% percentile deviation from end-to-end line	80% percentile stroke deviation from end-to-end line
26	20% percentile deviaiton from end-to-end line	20% percentile stroke deviation from end-to-end line .
27	50% percentile pairwise acceleration	50% percentile of the stroke acceleration.
28	Phone orientation	Portrait or landscape
29	Mid–stroke finger orientation	angle of finger orientation in the middle of a stroke
30	Up/down/left/right flag	Direction of the stroke in terms of screen quadrants.
31	Change of finger orientation	angle of finger orientation.

Table 7.2: Description of Touchalytics [40] features proposed by Frank et al. ranked according to their relative mutual information for spot-the-difference experiment.

technology because it is a well proven touch sensing technology which is practically implementable on mobile devices. The prototype is configured to sample data from all 24 sensors

at 50Hz, with 16 bit resolution. A Python application was developed on the prototype to coordinate the data acquisition.

7.5.3 User Study



(a) (b) (c)

Figure 7.4: Example of reading article (a) and images (b, c) used in the experiment.

In order to collect touch grip samples, 20 right-handed participants were recruited locally, 12 male and 8 female, age range 25–35 (mean = 29.4, sd = 3.15) each with at least one year of experience in using a smartphone. We want to compare Touchalytics features to BoD features, therefore we recreated the Touchalytics experiments to collect vertical (scroll) and horizontal (swipe) strokes. We followed the protocol used in Touchlytics by asking participants to read three articles, each of which required vertical scrolling, and compare three cartoon pairs via horizontal swiping while seated, with the phone in a single handed grip, portrait orientation. To encourage participants to navigate naturally, we let them perform the experiment at their own pace. We used six articles about *Coffee*, *Bicycle*, *Cat*, *Running*, *Smartphone* and *Glasgow* freely available on Wikipedia and six spot-the-difference cartoon images[4]. We randomly selected three from each for each participant. We edited all articles to have equal number of words and each pair of image to have precisely five differences. Example of the reading and spot-the-difference application is shown Figure 7.4.

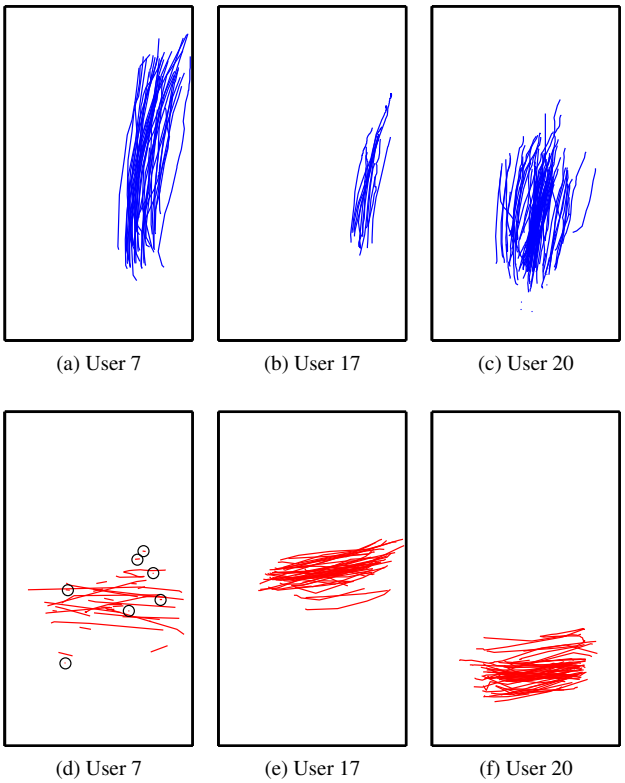
Each task begins as soon as the article/image pair is displayed on the screen. There is an exit button at the bottom of the article in the reading task to ensure participants read/scroll through the article until the end. For the image comparison, the exit button was shown at the top of the screen to the participants one minute after the task started. The participants are allowed to continue until they find all five differences. In the spot-the-difference task, we followed the technique used in Touchalytics by placing a blank panel between the images to ensure that at least two strokes are required to get from one image to another.

[4]http://easyfunpuzzles.com/category/spot-the-differences

For each task, we recorded timestamps, screen touch coordinates (x, y) in pixels and capacitive readings from the **BoD** into the prototype's internal storage for subsequent offline analysis. Each recording was performed in three sessions, and separated by an approximately five minute break. This is done by asking the participants to put down the phone on the table at the end of every session. This is to ensure that we are not observing only a single temporary grip pattern, but a range of plausible strokes and grips for each participant. Furthermore, the use of session also helps us to organise our training and test set data more fairly, by training on two sessions and testing on the third.

Overall experiment time ranged between 38–83 minutes per participant (mean = 52.6, $sd = 9.2$). A single reading task ranged between 6.1–21.7 minutes (mean = 8.4, sd = 0.5) while image comparison ranged between 1.2–2.4 minutes (mean = 1.2, sd = 0.11).

7.5.4 Feature Extraction



(a) User 7 (b) User 17 (c) User 20

(d) User 7 (e) User 17 (f) User 20

Figure 7.5: Examples of raw reading and game touch stroke from three different participants captured by the N9 prototype during reading (blue) and game (red) experiments. Differences between the participants are clearly visible. For participant 7 we highlighted the accidental touches (black circles) that were removed during pre-processing.

Touchstrokes Features

Each touchstroke is a sequence of touch coordinates (x, y) in pixels that begins with touch-

ing the screen and ends with lifting the finger. Prior to extracting the features, we filter out all accidental touches from the raw touch data to ensure that only plausible strokes are used in our study. We define an accidental touch to be a very short stroke with total displacement less than 10 pixels. Accidental touches have been highlighted in Figure 7.5 (d). To extract features from the raw touch data, we follow the techniques outlined in Touchalytics study. However, due to hardware constraints, we were unable to collect touch pressure, touch contact size and finger orientation data from our prototype. Moreover, we also omitted phone orientation from our feature set since our experiment was done in portrait mode only. Therefore, out of 31 features used in Touchalytics, we extracted 26. To avoid confusion with features from Touchalytics, we call our version of Touchalytics features *Touchstrokes*.

BoD *features*

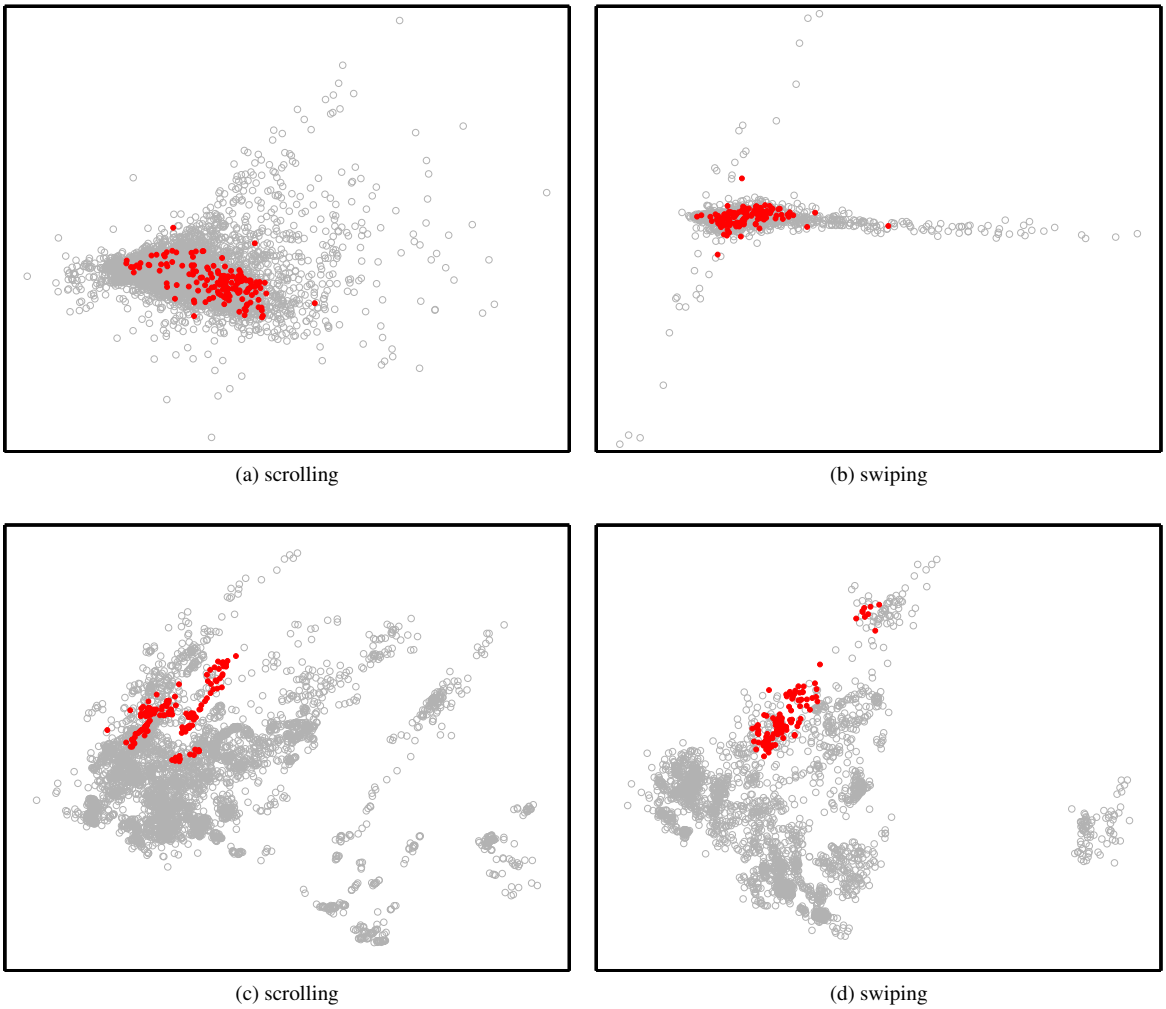
The **BoD** sensors produce 24 capacitive timeseries for each touch stroke action (one timeseries per sensor). The sensors in our prototype are occasionally affected by noise spikes which we remove using a simple length 3 median filter. These timeseries were reduced to 24 features by taking the mean value from each sensor across the stroke. Thus we only observe the static component of the grip during each stroke.

7.5.5 Analysis Methods

To visualise the overall structure of the stroke and **BoD** features we perform Principal Component Analysis (PCA). PCA enables us to see if any cluster structure exists within our feature sets and may also highlight session effects (i.e. differences within a user caused by multiple usage sessions). For further explanation, See Section 4.2.1.

As shown in Figure 7.6, distinctive user clusters can be clearly seen within the **BoD** features but, in Touchstroke features, this characteristic clustering is not visible. This suggests that participants may may be more distinct in their grip patterns than their touch characteristics. It might be possible that participants are linearly separable with the **BoD** features but not with Touchstrokes features. One experimental concern is whether user behaviour is consistent throughout the experiment. To show this behaviour, we show the data for the different experimental sessions using numbers and colour for one random user (user 17) in Figure 7.7. The plot suggests that the participant may have consistent touch behavior in all sessions, but different grip behaviour where distinct session clusters exist.

Next we perform mutual information analysis (MIA) between each feature modality, F_{touch} , F_{cap} and the user ID in our dataset. Please refer to Section 4.2.3 for a review on MIA. This measure corresponds to how well the user can be determined from the other users (combined) within the dataset. Features with 0% relative mutual information do not contain any information about the user identity whereas 100% means that the feature could be used to



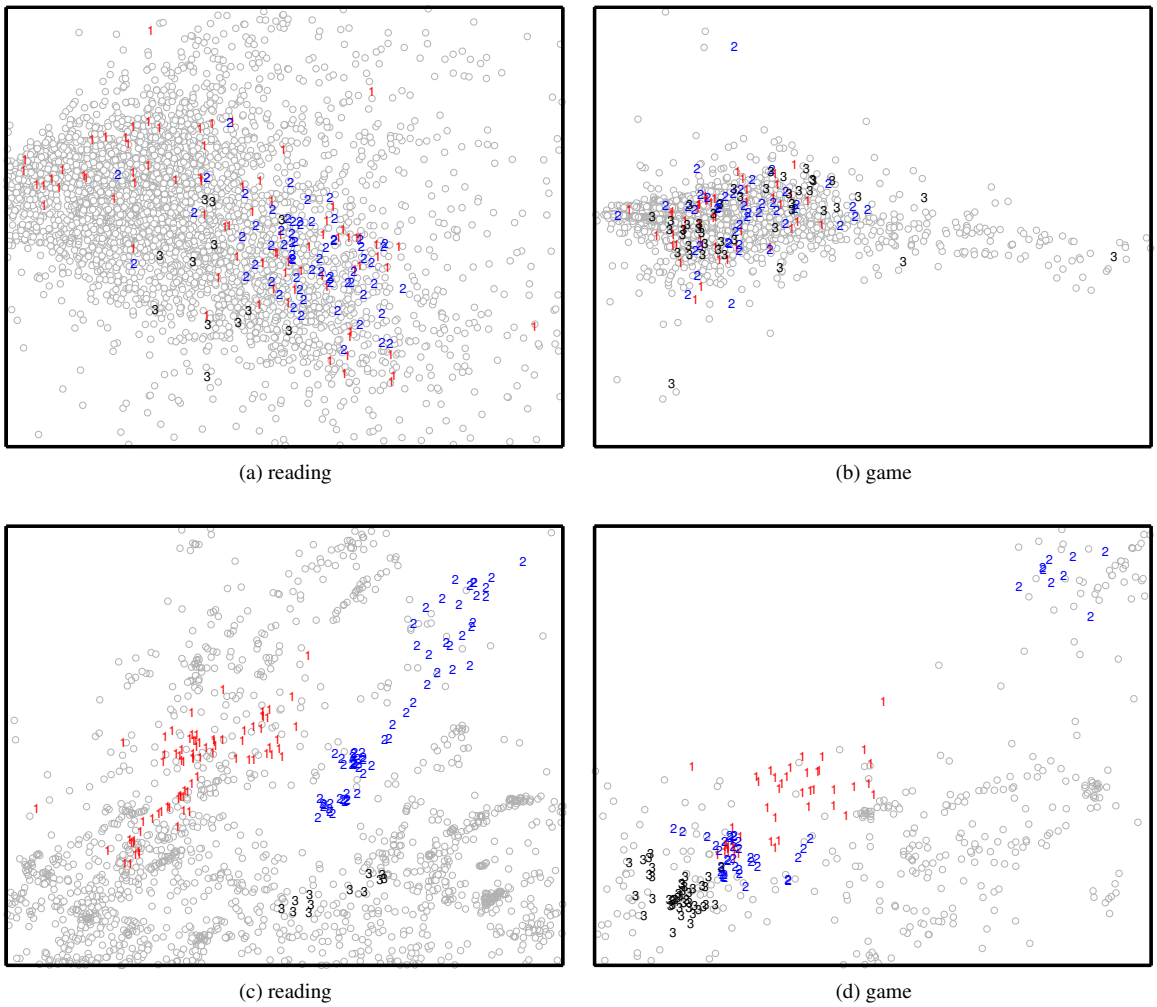
(a) scrolling (b) swiping

(c) scrolling (d) swiping

Figure 7.6: PCA plot in the global user space for (a, b) touchstroke and (c, d) **BoD** features, highlighting only user 17.

perfectly separate this user from the others. To compute relative mutual information between a feature F and user ID U, we first transform the feature to discrete variables by binning the values into 50 equally spaced bins ranging from the 10^{th} to 90^{th} quantile of the features. As shown in the Touchalytics study, this range makes our mutual information calculation more robust to outliers.

As shown in Figure 7.8 (a) and 7.8 (b), the five features with highest mutual information in Touchstrokes features are given by (1, 2) start-x, y, (3, 4) end-x, y positions of the stroke and (5) stroke duration in seconds. This differs from the original Touchalytics study [40] where the top 5 features were given by (1) *mid-stroke area covered*, (2) 20% *percentile pairwise velocity*, (3) *mid-stroke pressure*, (4) *direction of end-to-end line* and (5) x-coordinate of stroke stop position. Out of these, 2 features (*mid-stroke area covered* and *mid-stroke pressure*) were absent from our feature set due to hardware constraint. Interestingly, 20% *percentile pairwise velocity* scored 0% relative mutual information in our version but was the second



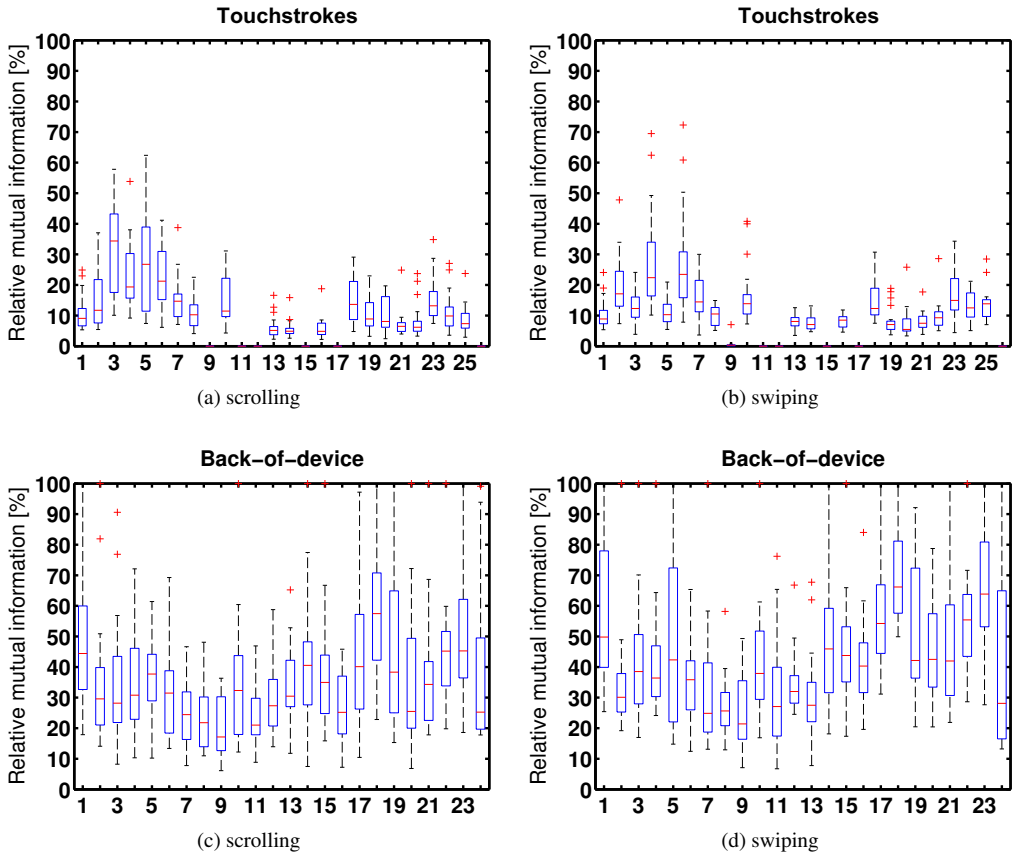
(a) reading (b) game

(c) reading (d) game

Figure 7.7: PCA in the global user space showing only user 17 for (a, b) touchstroke and (c, d) BoD features from every session. Each number/colour combination marker corresponds to session ID.

highest feature in the original Touchalytics study. This discrepancy could be due to the following reasons: (1) we used a slightly smaller sample population than Touchalytics; (2) users may possess similar touch behaviour; (3) our experimental tools (reading/game) may have provided different user experience from the one used in Touchalytics study.

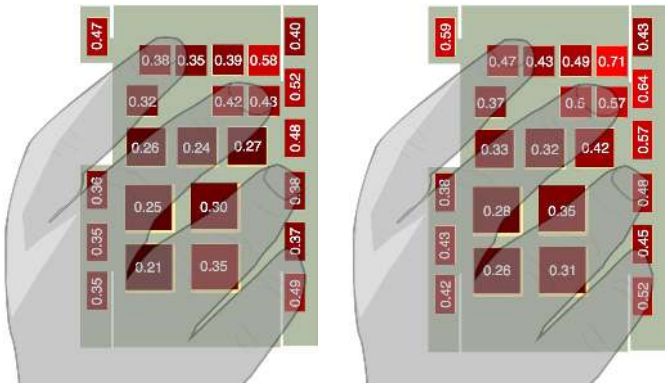
Figure 7.9 shows the mutual information of the BoD features, allowing us to get some insight into which BoD locations are useful for sensing hand grip. It is clear from the figure that sensors with high mutual information are located at the top-left and left side of the back-of-device (note that the image is reversed). This analysis also shows that sensor mutual information for the game task (Figure 7.9(b)) is much higher than that for the reading task (Figure 7.9(a)), suggesting that user hand grip is perhaps less variable in the former.



(a) scrolling (b) swiping

(c) scrolling (d) swiping

Figure 7.8: Mutual information analysis for Touchstrokes (a, b) and static back-of-device data type (c, d) for both reading (a, c) and game (b, d) tasks. From the figures, back-of-device data type clearly depict higher information content than Touchalytics in both tasks.



(a) scrolling (b) swiping

Figure 7.9: Hand overlay over back-of-device sensors shaded according to their respective relative mutual information values. Bright red shading corresponds to high whereas dark red shading corresponds to low relative mutual information.

7.5.6 Classification

The primary goal of this study is to measure the extent to which **BoD** features can be used to distinguish a specific user from a pool of other users. To measure this, we perform user classification by classifying a user of interest from the rest of users in our dataset (owner-versus-world). Within an authentication context, this represents the setting where a multi-user authentication system attempts to identify the current user from a set of users. This is a binary classification task where the problem is to determine whether the data belongs to either the positive (user of interest) or negative class (the rest).

In all experiments, the data was split into independent training and testing sets. Instead of randomly partitioning the data, we used the first and second sessions as training data and the third session as testing data. If there are session effects present, this will provide a more conservative estimate of performance than random splits.

The Support Vector Machine (SVM) classifier is a discriminative classifier that creates a separating hyperplane between the classes in a feature space defined by a kernel function. The SVM is a popular choice in solving user identification problems [40, 111, 119] and also exhibits general state-of-the-art performance in many data classification problems. Gaussian (RBF) kernel functions were used throughout this study. The form of the kernel function is given in Section 4.3.1.

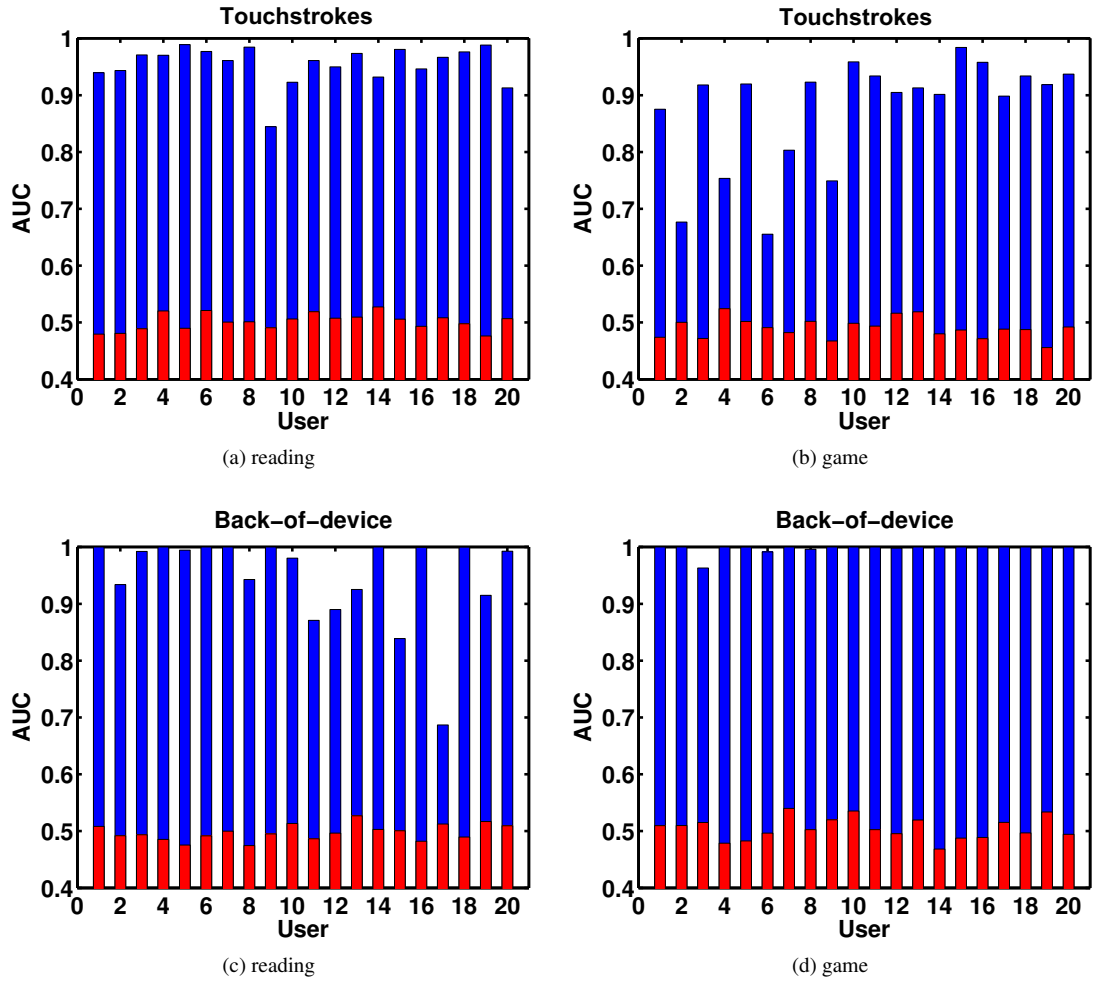
When combining Touchstrokes and **BoD** features into the same classifier, we used a linear combination of kernels with an additional parameter a that controls the influence of each kernel where $0 \leq a \leq 1$ with 0 represents only BoD kernel and 1 representing only Touchstrokes kernel. Optimising a allows us to measure the relative contribution of each data modality. We explain more about this procedure in Chapter 4.3.1.

Prior to all experiments, all features were normalised to have mean zero and standard deviation one. Data subsampling was used to address the class imbalance problem between positive (user of interest) and negative (the rest) classes. This was done by randomly sampling only as many data points in the negative class as there are in the positive class for each user. For all models, the C and γ parameters were selected via 10-fold cross-validation on the training set using a grid-search and AUC as the performance metric. For the multiple kernel models, we fixed the γ parameters for each kernel to the optimal individual values and optimised C and a with a 10-fold cross-validation procedure.

We compared the classification results using classifiers built using Touchstoke and BoD data alone, as well as the two data modalities combined. We used the following performance measures to evaluate the classifiers: the area under the ROC curve (AUC), False Accept Rate (FAR), False Reject Rate (FRR) and Equal Error (EER) rates. For further explanation, please refer to Section 4.6.

7.5.7 Results

In the following section, we report individual classification AUC (blue bars in plots) for all experiments. Our baseline is the performance of a classifer with randomised test data labels (i.e. with no predictive power). This is denoted by red bars, each of which has AUC of approximately 0.5. We use the non-parametric Wilcoxon Signed Rank Test with p-value threshold of 0.05 to test the significance level of the results.



(a) reading (b) game

(c) reading (d) game

Figure 7.10: Individual test data classification for both Touchstrokes (a, c) and back-of-device (b, d) features. Blue and red bars correspond to classification and baseline AUC respectively.

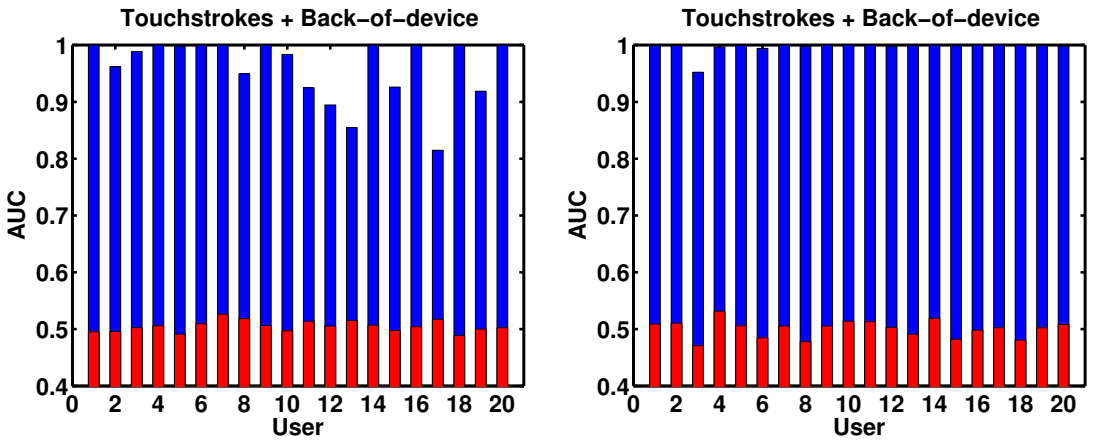
One vs All

This experiment represents the typical user authentication scenario where the classifier is trained to distinguish between the owner (user-of-interest) and other potential users. Here, we use user-specific classifiers trained using examples collected during the first and second

sessions and tested using the third session. This is to reflect the typical use case where calibration data is collected at some point (sessions 1 and 2) and then the user uses the device sometime in the future (session 3).

The results are presented in Table 7.3. In terms of overall performance, the mean AUC of BoD classifiers matches the mean AUC of Touchstrokes classifiers for the reading task. However, as shown in Figure 7.10(a) Touchstrokes classifiers depict a more stable individual performance compared to BoD (Figure 7.10(c)), particularly for user 17 where the AUC is much lower than the population mean AUC. This may be due to the session effects for user 17 highlighted in the previous PCA plots (Figure 7.7(c)). On the other hand, classification performance in the game task is dominated by the BoD classifiers. As shown in Figure 7.10(d), the BoD performance is much higher than Touchstrokes and able to classify users remarkably well most of the time.

Based on the single input result, we extended the classification task to see if any performance gain can be achieved by combination of the classifiers. To do so, we constructed a composite classifier by linearly combining Touchstrokes and BoD kernels. As shown in Table 7.3 and Figure 7.11 (a), a small improvement in AUC is achieved when both kernels is combined for the reading experiment, particularly for user 17. This improvement however is not statistically significant.



(a) reading (b) game

Figure 7.11: Individual test data classification for reading and game using composite classifiers. Blue and red bars correspond to classification and baseline AUC respectively.

As shown in Table 7.3, Touchstrokes produces higher mean FAR and EER compared to BoD in both tasks although these differences are not statistically significant. However, a relatively high mean FRR of 20.66% is given by BoD classifiers in the reading experiment. In the game task, Touchstrokes produces a significantly higher mean FAR, FRR and EER compared to BoD ($p \leq 0.0001$). The mean FAR for reading is reduced significantly from 9.08% to 2.30% ($p \leq 0.0006$) when using the composite kernel. Reduction also is achieved for FRR and

Features	FAR (%)	FRR (%)	EER (%)	AUC
Touchstrokes	9.08	10.16	9.62	0.95
BoD	3.52	20.66	7.41	0.95
Touchstrokes + BoD	2.30	20.49	6.23	0.96

(a) reading

Features	FAR (%)	FRR (%)	EER (%)	AUC
Touchstrokes	20.35	15.33	18.84	0.88
BoD	0.85	2.62	0.77	0.99
Touchstrokes + BoD	9.34	4.18	1.20	0.99

(b) game

Table 7.3: Mean classification errors (FAR, FRR, EER) and mean AUC for each feature type from all users.

EER, however these are not significant. Similar to the single kernel classification, the error rates for BoD are significantly lower than Touchstrokes in the game experiment ($p \leq 0.004$).

Training Time

In this experiment the goal is to evaluate the effect of training time on the performance of the classifiers. In particular, we are interested to see how much data the classifier needs to achieve high AUC. In practice, this represents the enrolment/calibration phase of an authentication system which should ideally be as short as possible. To simulate the calibration phase, we train the classifier using subsets of the data collected between the start and 60 seconds of the first and second sessions and evaluate performance with data from the entire third session.

As shown in Figure 7.12 (a) and (c) for reading, Touchstrokes may take more than 60 seconds from the beginning of the session to reach the highest recorded AUC value, whereas, BoD the highest AUC is already achieved at the beginning of the session and shows no discernible increase in AUC over time. In Figure 7.12 (b), we see that Touchstroke requires about 30 seconds to achieve the highest AUC in the game task. BoD (Figure 7.12 (d)) on the other hand can achieve this as soon as the session starts. Combining the back-of-device with the Touchalytics features brings the training time down to the very short time required for BoD as shown in Figure 7.12 (e). For game (Figure 7.12 (f)) similar performance is achieved as when using single kernel classifier. Out plots suggest that for both tasks, incorporating BoD data can substantially reduce the potential calibration time.

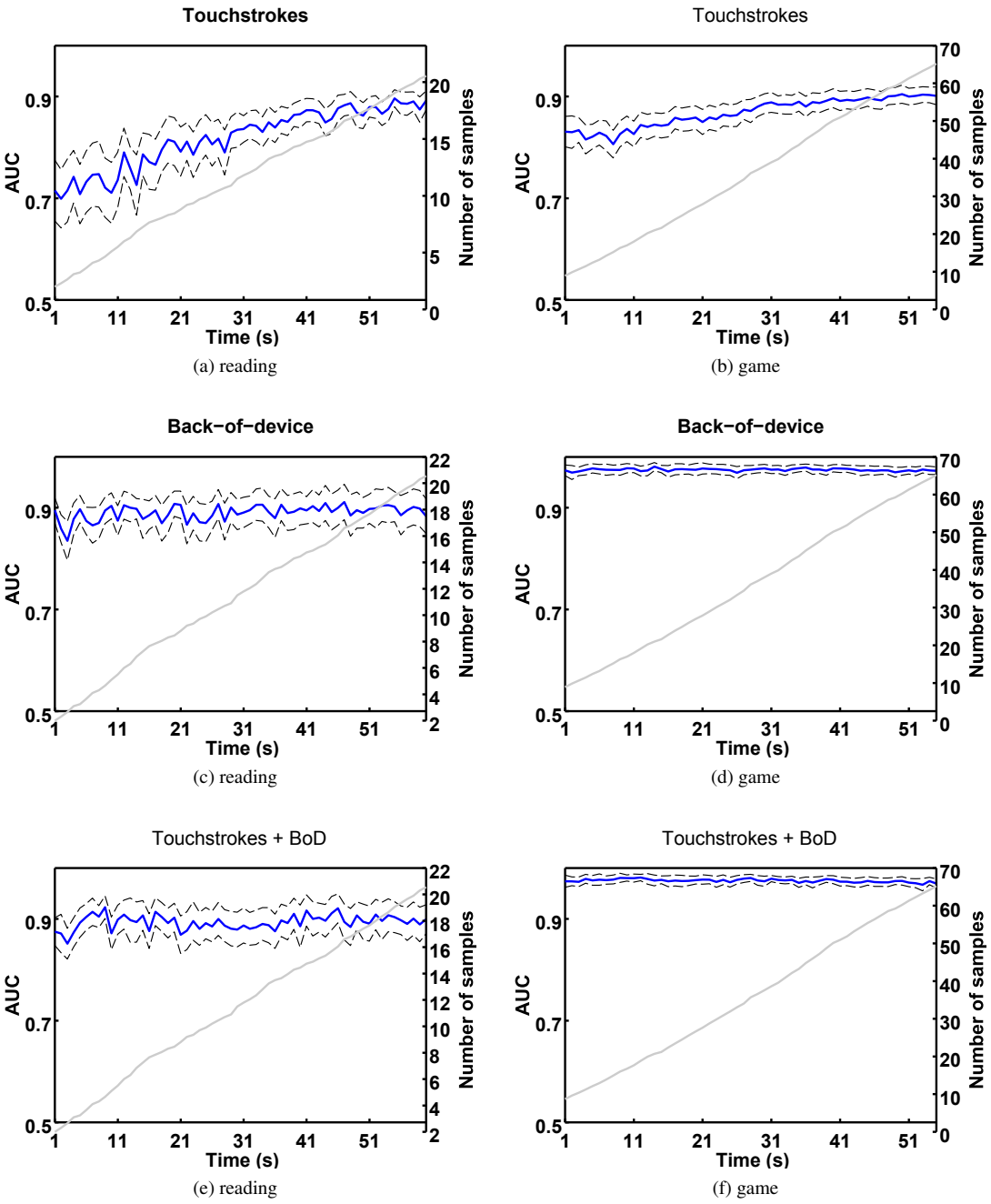
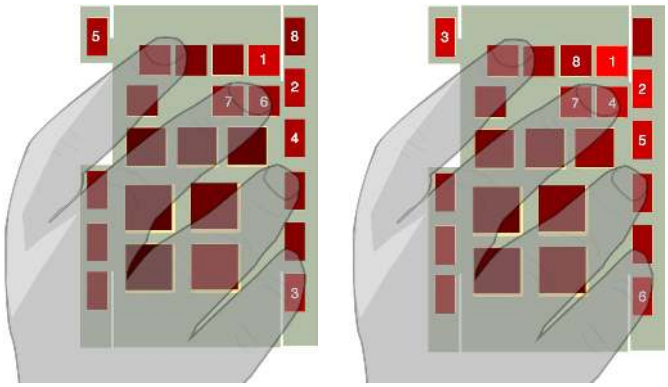


Figure 7.12: Test data classification AUC averaged across users with $\pm$ standard error (blue) and mean number of training examples (grey) using Touchstrokes, back-of-device and combination of both features as amount of time t is varied between the start and 60 seconds of the session.

Feature Selection

We now test whether a subset of features with high mutual information can be used to build a classifier that can match the performance of the classifier using all of the features. Reduction in the number of BoD features is desirable as it reduces the number of new sensors required



(a) reading (b) game

Figure 7.13: Position of the 8 most informative BoD sensors numbered according to their relative mutual information rank. Bright red shading corresponds to high relative mutual information whereas dark red shading corresponds to low relative mutual information.

on the device.

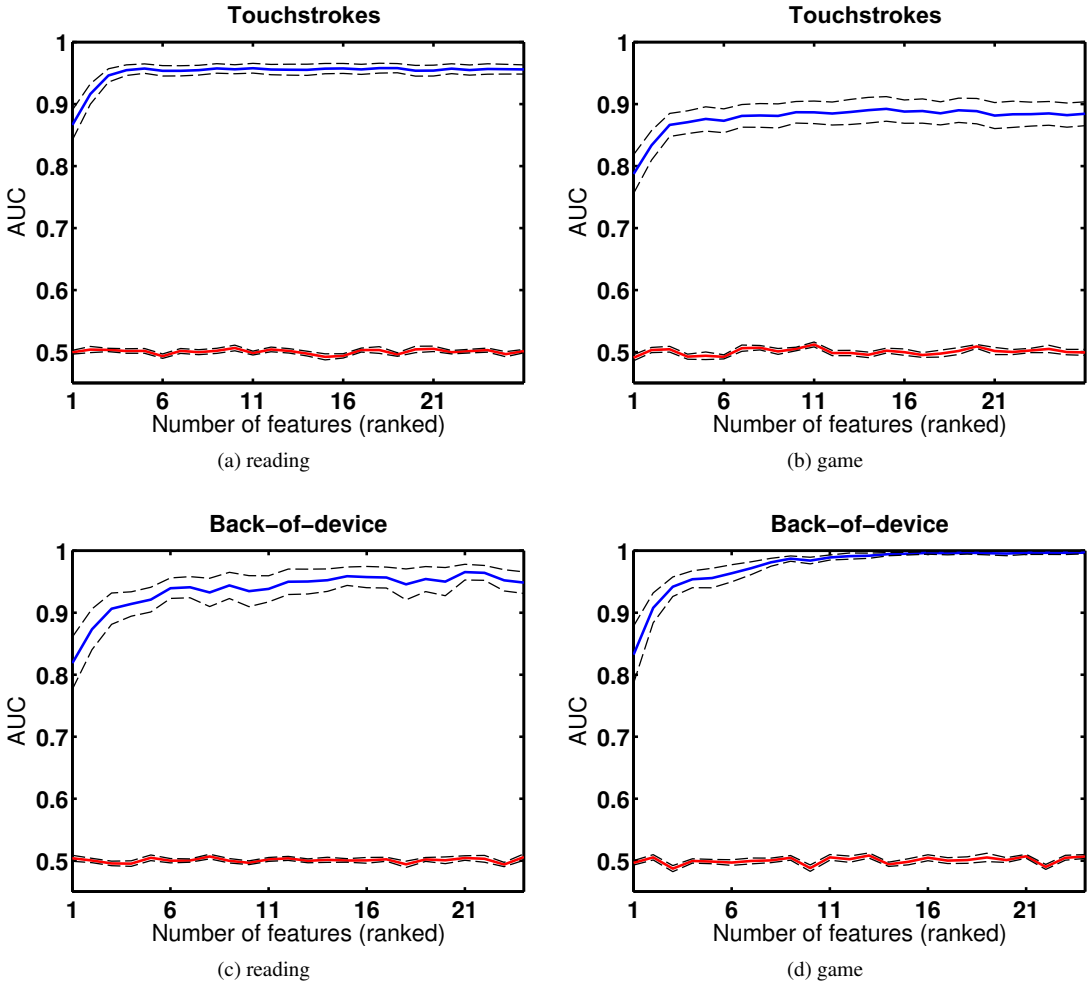
We begin by ranking the features according to the relative mutual information. Then we train the classifier using the single feature at the highest rank and then add the features one by one in order of rank. We again use data collected from the first and second sessions for training and data from the third session for testing. Figure 7.14 shows classification performance using features with the highest to lowest relative mutual information averaged across users. From the analysis, we see that Touchstrokes requires only the 5 most informative features to match the classifier trained with all features (c.f. Table 7.3). For BoD, optimal performance requires roughly 8 sensors. The position of these sensors is illustrated in Figure 7.13.

We now repeat our timing experiments for back-of-device with the smaller feature subset. We train the classifier using only the 8 highest relative mutual information sensors and test the classifier using the same procedure as in the training time experiment. Figure 7.15 shows the result for this experiment. Despite reducing the numbers of sensors, the training time is still very short for both tasks.

7.5.8 Discussion

Our results show that hand grip at the back-of-device interaction can be used to distinguish users. As shown in Table 7.3, it is evident that BoD could match the classifcation of performance of Touchsrokes in reading task. An FAR of 3.52% and FRR of 20.66% suggests that BoD would be insufficient as a primary authentication mechanism, but would contribute significant information to a continuous authentication system.

In the game task, BoD performance shows a significant improvement from 0.8758 to 0.9974 in AUC ($p \leq 0.0001$) over Touchstrokes. Moreover, the BoD also produces a significantly



(a) reading (b) game

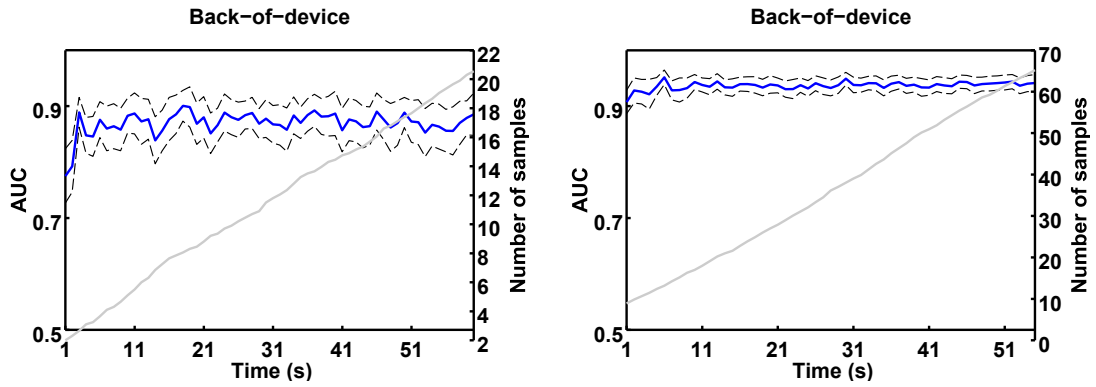
(c) reading (d) game

Figure 7.14: Test data classification AUC averaged across users ($\pm$ standard error) for Touchstrokes (a, c) and BoD (b,d) as number of features ranked from having highest to lowest relative mutual information is increased.

lower FAR of 0.85% and FRR of 2.62% suggesting that BoD in this type of horizontal swipe task might be very effective. However, we anticipate different levels of performance if the touch pressure feature as used in the original Touchalytics [10] were to be included.

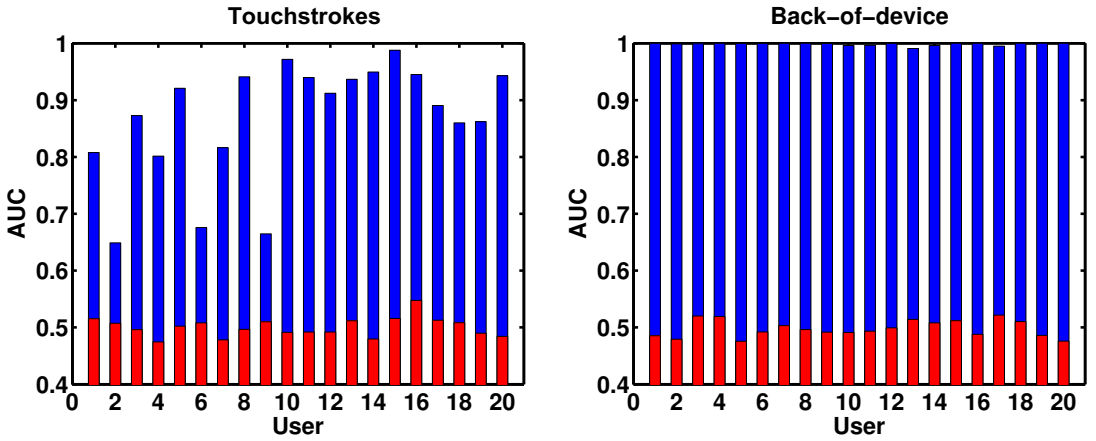
During the experiment we noticed that some users changed their grip every time they put down the device. We call this a *session effect*. For instance, such an effect is depicted by user 17 as shown in Figure 7.7(c), where the session examples are clearly separated into groups. Since our method relies on a model constructed from session examples, drastic change in grip may lead to poor classification, which is the case with user 17. This effect appears to be more obvious in the reading than game experiment. We hypothesise that after spending 5 or more minutes reading in the first session, a user may feel uncomfortable with the current grip position and alter it for comfort.

To validate the cause of this effect we repeated our analysis using randomised session la-



(a) reading (b) game

Figure 7.15: Test data classification AUC averaged across users with ± standard error (blue) and mean number of training examples (grey) using 8 most informative back-of-device sensors (relative mutual information) classifiers as amount of time t is varied between the start and 60 seconds of the session.



(a) game (b) reading

Figure 7.16: Individual test data classification for game (Touchstrokes) and reading (BoD) using classifiers trained using data assigned with random session ID. Blue and red bars correspond to classification AUC and baseline AUC respectively.

bels (i.e. mixing up all the sessions). Here the classifiers were trained in similar fashion using new surrogate sessions 1 and 2 and tested against session 3. The result is shown in Figure 7.16. The surrogate labels decrease individual AUC in Touchstrokes but gives perfect classification for all users in BoD. This suggests that *session effect* may be more significant in BoD recognition than on front-of-screen swiping.

Classification AUC can be improved by combining both Touchstrokes and BoD kernels together. This is particularly useful when building classifiers for users with *session effect* in one of the inputs. As in the case of user 17, the composite kernel increases individual AUC for reading from 0.63 to 0.78. Apart from *session effect*, the composite kernel generally also improves AUC and reduces both FAR and FRR. For instance, in the game experiment where

Touchstrokes performs much worse than BoD, the composite kernel significantly improves the mean AUC from 0.8758 to 0.9965 ($p \leq 0.0001$), from 15.33% to 4.18% in mean FAR ($p \leq 0.004$) and from 20.35% to 9.34% in mean FRR ($p \leq 0.0015$) over Touchstrokes.

Training time is one of the important factors in any authentication or identification system. Reading using Touchstrokes requires around 60 seconds (approximately 18 strokes) to achieve ≥ 0.85 AUC, and for the game task, similar performance can be achieved slightly earlier at about 30 seconds (approximately 40 strokes). For BoD, the classifier can distinguish the correct user with ≥ 0.85 AUC as soon after the first stroke is observed in both tasks.

Feature selection presents another interesting question in this study. Although this study uses classifiers built from all the computed features, it is evident that not every feature from the feature set is useful. This is particularly true for Touchstrokes as 5 out of 26 features show no predictive power at all. We demonstrate that by reducing the dimension of the features and number of original sensors, it is still possible to match the AUC of the full-featured classifiers.

7.6 Conclusion

In this chapter, we investigated whether hand grip sensed via **BoD** capacitive sensors can be used to distinguish users (RQ1). We chose two user distinguishing methods based on 1) hand movement [30] and 2) touchscreen touch behaviour [40] as direct comparison. We recreated the experiments used by both methods and augmenting them with hand grip input. Our results show that hand grip contains valuable information that can be used to distinguish users.

In the first study, we collected phone pickup action from 12 participants using a prototype mobile phone. From this, we extracted 3 features from the accelerometer input and 24 features from the **BoD** input. We then trained user classifiers using Gaussian RBF kernel SVM for each input and combination of both.

However a particular limitation of the study is that, it is based on a very small scale experiment. In the case of identification, the results are normally highly dependent on how many subjects/users are to be distinguished. Moreoever, the way the phone is picked up is constrained to only one particular setting (while seated) thus it is not known how this will generalise to actual real-world scenario.

In the second study, we ran similar experiments as described in [40] to record touch scrolling and touch swiping data using the N9 prototype from 20 participants. We then extracted 26 features from the touchscreen data and 24 features from the **BoD** and trained user classifiers

using Gaussian RBF kernel SVM. We obtained similar classification performance as the original study [40] method despite using different hardware, classification and pre-preprocessing techniques.

In both studies, we clearly show that a combination of **BoD** grip sensing and accelerometer motion signatures can be used to identify users in the few seconds between making contact with a device and lifting it to interact. Although BoD input dominate the classification performance in the second study (RQ2), the inclusion of touchscreen input does produce significant performance gains in some tasks (RQ3).

In the second study also, we show that the number of **BoD** sensors can be reduced to just six without impacting performance (RQ5). We also show that **BoD** input does not require lengthy training process to achieve high classification performance (RQ4). Although we used user–specific interaction classifiers, in practice, a system using a pooled model combined with a small individual training sample may provide adequate performance without requiring a lengthy enrollment process.

BoD sensors may not yet mainstream presently, however some commercial devices (e.g. the Doogee DG800 or the Oppo N1) already provide **BoD** touch sensors to enhance touchscreen interaction. These sensors may have sufficient resolution to capture hand grip for identification purposes. We anticipate BoD sensors with wider sensing surface to be included in commercial devices in the near future to allow grip-based interaction such in [49] which can be effectively used for grip-based implicit authentication.

In summary, these results form a concrete contribution to building a rich, multi-sensor mobile phone identification system. While hand grip alone may be insufficient to perform secure authentication, it can form part of an array of contributing virtual sensors in a hybrid continuous authentication system or as a lightweight identification model. The performance obtained in our experiments also suggest that these techniques could be readily applied to non-critical identification tasks (such as user identification with communal devices like remote control) or could feed into a larger authentication system as evidence for user identity.

Chapter 8

Conclusions

Implicit sensing of standard interactions allows the interaction to be understood from a different perspective. In this thesis we have shown that hand grip contact during touch targeting correlates strongly with the touch position on the touchscreen. We showed that hand grip while performing swiping and scrolling gestures are user–specific. We have also shown that users make subtle changes in their grip and hand pose when touch gesture mistakes are made. By using these implicit signals, we have demonstrated a number of enhancements to standard interaction.

Sensors perhaps are the most integral component in our work. A mobile phone is a sensor-laden device, making it an ideal platform for this purpose. We have used two input modalities: hand movement and back-of-device grip. While inertial sensors were used to measure the former, the latter sensing method was achieved by augmenting a standard mobile phone with a wrap-around capacitive sensor. These two modalities have been shown to contain valuable information that can be used to improve interaction.

We have proposed the use of machine learning techniques to enhance standard interaction using sensor data from two input modalities. We looked at the non-parametric Gaussian Process algorithm and non-probabilistic Support Vector Machine classifiers as well as decision tree based Random Forest classifiers. Using these we constructed: 1) implicit grip-based interaction models to estimate future touch location, 2) user-specific models based on grip and hand motion to implicitly identify users and 3) we trained the models to recognise touch gesture mistakes.

Our goal in applying these algorithms on mobile sensor data was to show that by modelling implicit interaction, we can provide measurable enhancements to the interaction thus offering benefits to the user. In this thesis we have shown that these models can be potentially used to improve touch targeting experience, complement existing phone authentication mechanism and detect mistakes from touchscreen interaction automatically, but more generally we

believe our work stands as an example of the value of modelling implicit components of standard interactions.

8.1 Summary of Contributions

This thesis makes a number of contributions, which were motivated by our thesis statement in Section 1.1. In support of the assertion that implicit interactions improves standard explicit interactions, we have made the following contributions:

- Chapter 5 presented grip and hand movement models based on Gaussian Process regression which describes the relationship between BoD sensor inputs and touch locations. We demonstrated how these models can be used to make predictions on general touch areas on the touch screen. We achieved prediction performance that is much better than random at a few hundred milliseconds before the actual touch. In addition, we implemented a real-time and simplified version of touch area predictor based on Gaussian Process models to demonstrate how our technique can be applied to the actual mobile phone to predict touch location.
- Chapter 6 presented a machine learning technique to detect swipe mistakes from grip and hand movement during cognitive demanding interaction. We showed that swipe mistakes manifests themselves as subtle flinches in the hand grip and movement which are made unconsciously during the interaction. Using SVM and Random Forest classifiers, we demonstrated that these flinches can be classified accurately several hundred milliseconds after the mistake was made. The automated detection of mistakes is a valuable component in touch interaction to allow users to recover from error gracefully.
- Chapter 7 presented two studies on identifying user implicitly from the way the device is being grasped/moved during normal interaction. We replicated two state of the art implicit identification techniques based on phone call answering pick-up motion and two native touch gestures and augmented them with hand grip and movement modalities. We showed that hand grip and movement during interaction are user-specific and using SVM classifiers, these can be identified with a higher level of accuracy compared to the previous techniques.

8.2 Future Work

There are a number of interesting avenues for future work which could follow from the results presented in this thesis. The following sections will describe these.

8.2.1 Modelling Dynamic Hand Grip and Movement

In this thesis, all of the predictions were based on the models constructed using static components of the hand grip and movement. GP models in Chapter 5, were trained using sensor values at different time points prior to the touch and we used these to predict touch locations. Similarly, in Chapter 6 and 7, the features used to train the SVM classifiers were mean values extracted from different segments in the sensor timeseries.

Despite achieving good prediction performances, our static modelling approach may lose useful information encoded at a certain segment of the sensor timeseries. For example, our models did not capture the whole changes in hand grip and motion changes when performing a particular task, rather it only captured snapshots of the changes. Using the whole timeseries as a feature will allow us to construct models that represent the whole interaction and enable us to conduct a more realistic analysis.

Implementing dynamic modelling and measuring how its performance compares to static modelling is a useful area of work in the future. To do this, a small change in the sensor data pre-processing is required. Instead of extracting mean values from several blocks in the timeseries, the models can be trained using the entire timeseries. This requires decimating or interpolating the signals to fix the length of the timeseries thus, producing a higher dimensional feature data. Theoretically, this will not scale up the computational complexities of the learning algorithms, thus making it an appealing approach to model the interaction on.

8.2.2 Realistic Mobile Phone Usage Scenarios

In Chapter 7, the phone answering task we used in the first user study was constrained to two simple pick-up variations, that is picking up the phone with right/left hand and bringing it to the right/left ear. In reality, this is not always the case, in fact, during the user study, we saw that some participants naturally brought the phone to the opposite ear (right hand to the left ear). We believe that covering more than two pick-up variations is an interesting avenue for future work in this area, particularly when building robust model involving security.

In the second user study of the same chapter, all user classifiers were constructed using data collected from different sessions in a day. Although we achieved good identification performance from this study, this represents only one phone usage scenario. Therefore, in the future it is useful to see how different scenarios such as evaluating the models using data collected from from different days or weeks affect the identification performance.

Similarly in Chapter 6 our data was collected in a very controlled setting using well-defined cognitive tasks. It is an important step to further demonstrate that detection is possible in a more realistic context. For example, it is unclear how user movement would effect predic-

tions derived from the accelerometer, or how the signals would change when a user was less engaged with the task.

Based on these observations, the next step therefore is to consider a wider range of realistic mobile phone usage scenarios when designing experimental user studies. In addition to that, using a larger population also is another potential direction of future work.

8.2.3 Additional Sensing Input for Back-of-Device Interaction

Our 24 back-of-device (BoD) sensors are based on the popular, robust and yet low–cost capacitive sensing technology. As depicted in all of our results, these sensors were able to capture salient information from the grip contact to model implicit BoD interaction. Although grip contact alone contains sufficient predictive power, we believe that using additional sensor input for BoD interaction would be an interesting line of work in the future.

One of the important grip characteristics that related closely to grip contact is grip strength. This can be be measured using force sensors [121, 128]. In terms of application, this seems feasible since sensing technology that combines capacitive and force sensors is already made available for the masses[1] which can be practically adapted to BoD. The use of pressure information will add an extra dimension to the BoD grip interaction, for example in Chapter 5, grip pressure can be used as another measurement (or in combination to grip contact) to construct GP models for touch location prediction.

8.2.4 Effect of Different Type of Gesture and Error

In Chapter 6, we have only studied one type of gesture, and one type of error. It would be particularly interesting to look at whether similar performance is possible within a completely different domain. For example, when entering text, many different error types are possible: physical errors (hitting the wrong key), cognitive errors (spelling a word incorrectly) and system errors (incorrectly auto-correcting a word). It is possible that the signals from the user (if they exist) would be different in these three cases although detecting these different situations would allow us to make smart correction systems.

As discussed in Section 6.5, we found classification performance to remain fairly constant over time, suggesting that the error information is present very early in the time series. Visual inspection of Figure 6.5 suggests that the error signal appears slightly earlier in the accelerometer data than it does in the BoD data. This is worth investigating further, for which a larger dataset is required to ensure sufficient statistical power to identify these changes.

[1] http://www.apple.com/uk/iphone-6s/3d-touch/

8.3 Summary and Conclusions

In this thesis we have shown how implicit sensing from standard interactions can be used to enhance touch interaction and usability of authentication system in mobile devices. We have built both probabilistic and non-probabilistic models and used their predictions to estimate future front-of-device touch locations, identify users transparently and detect swipe errors automatically. Further, we have shown that the combination use of BoD grip and hand movement in modelling implicit interaction can lead to better prediction performances.

Although there are still many outstanding issues to consider, we believe that the work presented here in this thesis is a compelling case for the power of machine learning techniques in solving problems in this area. Implicit interaction particularly using BoD is often neglected in the literature, and our results show that this has resulted in loss of useful and interesting information which can be used to improve the quality of interaction using mobile devices.

Bibliography

[1] H. J. Ailisto, M. Lindholm, J. Mantyjarvi, E. Vildjiounaite, and S.-M. Makela, "Identifying people from gait pattern with accelerometers," *Proc. SPIE - Int. Soc. Opt. Eng.*, vol. 5779, no. May 2016, pp. 7–14, 2005. [Online]. Available: http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=863374

[2] J. Angulo and E. Wästlund, "Exploring Touch Screen Biometrics for User Identification on Smart Phones," *Priv. Identity 2011, IFIP AICT*, pp. 130–143, 2012. [Online]. Available: http://www.springerlink.com/index/24722X8970259317.pdf

[3] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge Attacks on Smartphone Touch Screens," *USENIX Conf. Offensive Technol.*, pp. 1–7, 2010. [Online]. Available: http://www.usenix.org/event/woot10/tech/full{_}papers/Aviv.pdf

[4] F. Bagalà, C. Becker, A. Cappello, L. Chiari, K. Aminian, J. M. Hausdorff, W. Zijlstra, and J. Klenk, "Evaluation of accelerometer-based fall detection algorithms on real-world falls." *PLoS One*, vol. 7, no. 5, p. e37062, jan 2012. [Online]. Available: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3353905{&}tool=pmcentrez{&}rendertype=abstract

[5] L. Bao and S. Intille, "Activity recognition from user-annotated acceleration data," *Pervasive Comput.*, pp. 1–17, 2004. [Online]. Available: http://www.springerlink.com/index/9AQFLYK4F47KHYJD.pdf

[6] P. Baudisch and G. Chu, "Back-of-device interaction allows creating very small touch devices," *Proc. 27th Int. Conf. Hum. factors Comput. Syst. - CHI 09*, no. c, p. 1923, 2009. [Online]. Available: http://dl.acm.org/citation.cfm?doid=1518701.1518995

[7] J. Bergstrom-Lehtovirta and A. Oulasvirta, "Modeling the functional area of the thumb on mobile touchscreen surfaces," in *Proc. 32nd Annu. ACM Conf. Hum. factors Comput. Syst. - CHI '14*. New York, New York, USA: ACM Press, apr 2014, pp. 1991–2000. [Online]. Available: http://dl.acm.org/citation.cfm?id=2611528.2557354

[8] J. Bergstrom-Lehtovirta, A. Oulasvirta, and S. Brewster, "The effects of walking speed on target acquisition on a touchscreen interface," in *Proc. 13th Int.*

Conf. Hum. Comput. Interact. with Mob. Devices Serv. - MobileHCI '11. New York, New York, USA: ACM Press, aug 2011, p. 143. [Online]. Available: http://dl.acm.org/citation.cfm?id=2037373.2037396

[9] Y. Bernaerts, M. Druwé, S. Steensels, J. Vermeulen, and J. Schöning, "The office smartwatch: development and design of a smartwatch app to digitally augment interactions in an office environment," in *Proc. 2014 companion Publ. Des. Interact. Syst. - DIS Companion '14*. New York, New York, USA: ACM Press, jun 2014, pp. 41–44. [Online]. Available: http://dl.acm.org/citation.cfm?id=2598784.2602777

[10] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[11] L. Bottou and C. Lin, "Support vector machine solvers," *Large scale kernel Mach.*, pp. 1–27, 2007. [Online]. Available: http://140.112.30.28/{~}cjlin/papers/bottou{_}lin.pdf

[12] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: http://link.springer.com/article/10.1023/A{%}3A1010933404324

[13] A. Butler, S. Izadi, and S. Hodges, "SideSight," in *Proc. 21st Annu. ACM Symp. User interface Softw. Technol. - UIST '08*. New York, New York, USA: ACM Press, oct 2008, p. 201. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1449715.1449746

[14] B. Buxton, "ntegrating the periphery and context: A new taxonomy of telematics," in *Proc. Graph. Interface*, 1995, pp. 239–246.

[15] A. Campbell, T. Choudhury, S. Hu, H. Lu, M. K. Mukerjee, M. Rabbi, and R. D. Raizada, "NeuroPhone," in *Proc. Second ACM SIGCOMM Work. Networking, Syst. Appl. Mob. handhelds - MobiHeld '10*. New York, New York, USA: ACM Press, aug 2010, p. 3. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1851322.1851326

[16] J. Candia, M. C. González, P. Wang, T. Schoenharl, G. Madey, and A.-L. Barabási, "Uncovering individual and collective human dynamics from mobile phone records," *J. Phys. A Math. Theor.*, vol. 41, no. 22, p. 224015, jun 2008. [Online]. Available: http://arxiv.org/abs/0710.2939

[17] C.-C. C.-c. C.-C. C. Chang and C.-j. C. C.-J. Lin, "A Library for Support Vector Machines," *ACM Trans. Interlligent Syst. Technol.*, vol. 2, no. 3, p. 39, 2011. [Online]. Available: http://www.cs.cmu.edu/{~}pakyan/compbio/references/Chang{_}LIBSVM{_}2001.pdf$\delimiter"026E30F$nhttp://dl.acm.org/citation.cfm?id=1961199

[18] C.-C. Chang and C.-J. Lin, "LIBSVM," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, apr 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=1961189.1961199

[19] W. Chang, K. Kim, H. Lee, J. Cho, B. Soh, J. Shim, G. Yang, S.-j. Cho, and J. Park, "Recognition of Grip-Patterns by Using Capacitive Touch Sensors," in *2006 IEEE Int. Symp. Ind. Electron.* IEEE, jul 2006, pp. 2936–2941. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4078859

[20] R. Chavarriaga, P. W. Ferrez, J. d. R. Millán, and J. Millan, "To err is human: Learning from error potentials in brain-computer interfaces," *Adv. Cogn. Neurodynamics ICCN 2007*, pp. 777–782, 2008. [Online]. Available: http://link.springer.com/chapter/10.1007/978-1-4020-8387-7{_}134

[21] R. Chavarriaga, A. Biasiucci, K. Forster, D. Roggen, G. Troster, and J. D. R. Millan, "Adaptation of hybrid human-computer interaction systems using EEG error-related potentials." *Conf. Proc. ... Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. IEEE Eng. Med. Biol. Soc. Annu. Conf.*, vol. 2010, pp. 4226–9, jan 2010. [Online]. Available: http://europepmc.org/abstract/med/21096899

[22] R. Chen, M. F. She, X. Sun, L. Kong, and Y. Wu, "Driver recognition based on dynamic handgrip pattern on steeling wheel," *Proc. - 2011 12th ACIS Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel Distrib. Comput. SNPD 2011*, pp. 107–112, jul 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6063552

[23] L.-P. Cheng, F.-I. Hsiao, Y.-T. Liu, and M. Y. Chen, "iRotate grasp," in *Adjun. Proc. 25th Annu. ACM Symp. User interface Softw. Technol. - UIST Adjun. Proc. '12*. New York, New York, USA: ACM Press, 2012, p. 15. [Online]. Available: http://dl.acm.org/citation.cfm?id=2380296.2380305

[24] L.-P. Cheng, H.-s. Liang, C.-Y. Wu, and M. Y. Chen, "iGrasp," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst. - CHI '13*. New York, New York, USA: ACM Press, 2013, p. 3037. [Online]. Available: http://dl.acm.org/citation.cfm?id=2479514http://dl.acm.org/citation.cfm?doid=2470654.2481422

[25] V. Cheung and S. D. Scott, "Revisiting hovering: interaction guides for interactive surfaces," *ACM Int. Conf. Interact. Tabletops Surfaces*, pp. 355–358, 2012. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2396636.2396699

[26] E. S. Choi, W. C. Bang, S. J. Cho, J. Yang, D. Y. Kim, and S. R. Kim, "Beatbox music phone: Gesture-based interactive mobile phone using a tri-axis accelerometer,"

in *Proc. IEEE Int. Conf. Ind. Technol.*, vol. 2005. IEEE, 2005, pp. 97–102. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=1600617

[27] S. Choi, J. Han, S. Kim, S. Heo, and G. Lee, "ThickPad: A Hover-Tracking Touchpad for a Laptop," in *Proc. 24th Annu. ACM Symp. Adjun. User interface Softw. Technol. - UIST '11 Adjun.* New York, New York, USA: ACM Press, oct 2011, p. 15. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2046396.2046405

[28] N. Clarke, S. Karatzouni, and S. Furnell, "Flexible and Transparent User Authentication for Mobile Devices," *Emerg. Challenges Secur. Priv. Trust*, vol. 297, pp. 1–12, 2009.

[29] J. Clawson, K. Lyons, A. Rudnick, R. A. Iannucci, and T. Starner, "Automatic whiteout++," in *Proceeding twenty-sixth Annu. CHI Conf. Hum. factors Comput. Syst. - CHI '08*. New York, New York, USA: ACM Press, apr 2008, p. 573. [Online]. Available: http://dl.acm.org/citation.cfm?id=1357054.1357147

[30] M. Conti, I. Zachia-Zlatea, and B. Crispo, "Mind how you answer me!: transparently authenticating the user of a smartphone when answering or placing a call," in *Proc. 6th ACM Symp. Information, Comput. Commun. Secur.*, 2011, pp. 249–259.

[31] S. A. Coombes, K. M. Gamble, J. H. Cauraugh, and C. M. Janelle, "Emotional states alter force control during a feedback occluded motor task." *Emotion*, vol. 8, no. 1, pp. 104–13, feb 2008. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/18266520

[32] H. Crawford, "A framework for continuous, transparent authentication on mobile devices," Ph.D thesis, University of Glasgow, 2012. [Online]. Available: http://theses.gla.ac.uk/4046/

[33] J. Cutting and L. Kozlowski, "Recognizing friends by their walk: Gait perception without familiarity cues," *Bull. Psychon. ...*, 1977. [Online]. Available: http://people.psych.cornell.edu/{~}jec7/pubs/friends.pdf

[34] B. Dal Seno, M. Matteucci, and L. Mainardi, "Online detection of P300 and error potentials in a BCI speller." *Comput. Intell. Neurosci.*, vol. 2010, p. 307254, jan 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1740646.1840654

[35] A. De Luca, A. Hang, F. Brudy, C. Lindner, and H. Hussmann, "Touch me once and i know it's you!" in *Proc. 2012 ACM Annu. Conf. Hum. Factors Comput. Syst. - CHI '12*. New York, New York, USA: ACM Press, may 2012, p. 987. [Online]. Available: http://dl.acm.org/citation.cfm?id=2207676.2208544

[36] M. O. Derawi, C. Nickel, P. Bours, and C. Busch, "Unobtrusive User-Authentication on Mobile Phones Using Biometric Gait Recognition," in *2010 Sixth Int. Conf. Intell. Inf. Hiding Multimed. Signal Process.* IEEE, oct 2010, pp. 306–311. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5638036

[37] H. Drewes, A. De Luca, and A. Schmidt, "Eye-gaze interaction for mobile phones," *Mobil. '07 Proc. 4th Int. Conf. Mob. Technol. Appl. Syst. 1st Int. Symp. Comput. Hum. Interact. Mob. Technol.*, vol. 07, pp. 364–371, 2007. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1378063.1378122http://dl.acm.org/citation.cfm?id=1378122

[38] B. A. Eriksen and C. W. Eriksen, "Effects of noise letters upon the identification of a target letter in a nonsearch task," *Percept. Psychophys.*, vol. 16, no. 1, pp. 143–149, jan 1974. [Online]. Available: http://www.springerlink.com/index/10.3758/BF03203267

[39] P. W. Ferrez and J. d. R. Millán, "You Are Wrong!—Automatic Detection of Interaction Errors from Brain Waves," *Proc. 19th Int. Jt. Conf. Artif. Intell.*, 2005. [Online]. Available: http://infoscience.epfl.ch/record/83269

[40] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, "Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 1, pp. 136–148, jan 2013. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6331527

[41] D. Gafurov, K. Helkala, and T. Søndrol, "Biometric gait authentication using accelerometer sensor," *J. Comput.*, vol. 1, no. 7, pp. 51–59, nov 2006. [Online]. Available: http://academypublisher.com/ojs/index.php/jcp/article/view/277https://www.academypublisher.com/{~}academz3/ojs/index.php/jcp/article/view/01075159

[42] C. Giuffrida, K. Majdanik, M. Conti, and H. Bos, "I sensed it was you: Authenticating mobile users with sensor-enhanced keystroke dynamics," in *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8550 LNCS. Springer International Publishing, 2014, pp. 92–111. [Online]. Available: http://link.springer.com/10.1007/978-3-319-08509-8{_}6

[43] M. Goel, J. Wobbrock, and S. Patel, "GripSense," *Proc. 25th Annu. ACM Symp. User interface Softw. Technol. - UIST '12*, p. 545, 2012. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2380116.2380184

[44] J. Goodman, G. Venolia, K. Steury, and C. Parker, "Language modeling for soft keyboards," in *Proc. 7th Int. Conf. Intell. user interfaces - IUI '02*. New

York, New York, USA: ACM Press, jan 2002, pp. 194–195. [Online]. Available: http://portal.acm.org/citation.cfm?doid=502716.502753

[45] M. S. Gordon, D. K. Hong, P. M. Chen, J. Flinn, S. Mahlke, and Z. M. Mao, "Tango: Accelerating Mobile Applications through Flip-Flop Replication," *GetMobile Mob. Comput. Commun.*, vol. 19, no. 3, pp. 10–13, dec 2015. [Online]. Available: http://dl.acm.org/citation.cfm?id=2867070.2867075

[46] M. Greene, "Review of Gun Safety Technologies," National Institute of Justice, Tech. Rep., 2013. [Online]. Available: https://www.ncjrs.gov/pdffiles1/nij/242500.pdf

[47] J. Gu and G. Lee, "TouchString," in *Proc. 24th Annu. ACM Symp. Adjun. User interface Softw. Technol. - UIST '11 Adjun.* New York, New York, USA: ACM Press, oct 2011, p. 75. [Online]. Available: http://dl.acm.org/citation.cfm?id=2046396.2046430

[48] M. Haescher, D. J. C. Matthies, and B. Urban, "Anomaly Detection with Smartwatches as an Opportunity for Implicit Interaction," in *Proc. 17th Int. Conf. Human-Computer Interact. with Mob. Devices Serv. Adjun. - MobileHCI '15*. New York, New York, USA: ACM Press, aug 2015, pp. 955–958. [Online]. Available: http://dl.acm.org/citation.cfm?id=2786567.2794308

[49] Y. Higuchi and T. Okada, "User Interface Using Natural Gripping Features," *NTT Docomo Tech. J.*, vol. 15, no. 3, pp. 17—-24, 2014.

[50] C. Hill, "Wearables the future of biometric technology?" *Biometric Technol. Today*, vol. 2015, no. 8, pp. 5–9, sep 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0969476515301387

[51] K. Hinckley and H. Song, "Sensor synaesthesia," in *Proc. 2011 Annu. Conf. Hum. factors Comput. Syst. - CHI '11*. New York, New York, USA: ACM Press, may 2011, p. 801. [Online]. Available: http://dl.acm.org/citation.cfm?id=1978942.1979059

[52] K. Hinckley, J. Pierce, E. Horvitz, and M. Sinclair, "Foreground and Background Interaction with Sensor-enhanced Mobile Devices," *ACM Trans. Comput. Interact.*, vol. 12, no. 1, pp. 31–52, mar 2005. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1057237.1057240http://doi.acm.org/10.1145/1057237.1057240

[53] K. Hinckley, S. Heo, M. Pahud, C. Holz, H. Benko, A. Sellen, R. Banks, K. O'Hara, G. Smyth, and B. Buxton, "Pre-Touch Sensing for Mobile Interaction," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2016.

[54] C. B. Holroyd and M. G. H. Coles, "The neural basis of human error processing: Reinforcement learning, dopamine, and the error-related negativity." *Psychol. Rev.*, vol. 109, no. 4, pp. 679 – 709, 2002.

[55] H. Hotelling, "Relations Between Two Sets of Variates on JSTOR," *Biometrika*, pp. 321–377, 1936. [Online]. Available: http://www.jstor.org/stable/2333955?seq=1{#}page{_}scan{_}tab{_}contents

[56] E. L. Hutchins, J. D. Hollan, and D. A. Norman, "Direct Manipulation Interfaces," *HumanComputer Interact.*, vol. 1, no. 4, pp. 311–338, nov 2009. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1207/s15327051hci0104{_}2

[57] A. Jain and V. Kanhangad, "Exploring orientation and accelerometer sensor data for personal authentication in smartphones using touchscreen gestures," *Pattern Recognit. Lett.*, vol. 68, pp. 351–360, dec 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865515002056

[58] M. Jakobsson, E. Shi, P. Golle, and R. Chow, "Implicit authentication for mobile devices," *... 4th USENIX Conf. ...*, 2009. [Online]. Available: http://www.usenix.org/event/hotsec09/tech/full{_}papers/jakobsson.pdf

[59] M. Joselli and E. Clua, "gRmobile: A Framework for Touch and Accelerometer Gesture Recognition for Mobile Games," in *2009 VIII Brazilian Symp. Games Digit. Entertain.* IEEE, 2009, pp. 141–150. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5479100

[60] W. Ju and L. Leifer, "The Design of Implicit Interactions: Making Interactive Systems Less Obnoxious," *Des. Issues*, vol. 24, no. 3, pp. 72–84, jul 2008. [Online]. Available: http://www.mitpressjournals.org/doi/abs/10.1162/desi.2008.24.3.72?journalCode=desi{#}.VwPISHUrKV4

[61] W. Ju, B. A. Lee, and S. R. Klemmer, "Range: Exploring Implicit Interaction through Electronic Whiteboard Design," *CSCW '08 Proc. 2008 ACM Conf. Comput. Support. Coop. Work*, pp. 17–26, 2008. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1460563.1460569http://dl.acm.org/citation.cfm?id=1460569

[62] J. Kangas, D. Akkil, J. Rantala, P. Isokoski, P. Majaranta, and R. Raisamo, "Gaze Gestures and Haptic Feedback in Mobile Devices," *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, pp. 435–438, 2014. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2556288.2557040http://doi.acm.org/10.1145/2556288.2557040

[63] J. A. Kauffman, A. M. Bazen, S. H. Gerez, and R. N. J. Veldhuis, "Grip-pattern recognition for smart guns," pp. 379–384, mar 2003. [Online]. Available: http://www.stw.nl/prorisc/proc-2003/saip/kauffman.pdf

[64] M. Kay, K. Rector, S. Consolvo, B. Greenstein, J. O. Wobbrock, N. F. Watson, and J. a. Kientz, "PVT-Touch: Adapting a Reaction Time Test for Touchscreen Devices," *Proc. 7th Int. Conf. Pervasive Comput. Technol. Healthc.*, pp. 248–251, 2013.

[65] J. Kela, P. Korpipää, J. Mäntyjärvi, S. Kallio, G. Savino, L. Jozzo, and S. Di Marca, "Accelerometer-based gesture control for a design environment," *Pers. Ubiquitous Comput.*, vol. 10, no. 5, pp. 285–299, aug 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1149818.1149819

[66] D. T. Kemp, "Stimulated acoustic emissions from within the human auditory system," *J. Acoust. Soc. Am.*, vol. 64, no. 5, p. 1386, nov 1978. [Online]. Available: http://scitation.aip.org/content/asa/journal/jasa/64/5/10.1121/1.382104

[67] H. Ketabdar, M. Roshandel, and K. A. Yüksel, "Towards using embedded magnetic field sensor for around mobile device 3D interaction," in *Proc. 12th Int. Conf. Hum. Comput. Interact. with Mob. devices Serv. - MobileHCI '10*. New York, New York, USA: ACM Press, sep 2010, p. 153. [Online]. Available: http://dl.acm.org/citation.cfm?id=1851600.1851626

[68] A. U. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A Survey of Mobile Cloud Computing Application Models," *IEEE Commun. Surv. Tutorials*, vol. 16, no. 1, pp. 393–413, 2014. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6553297

[69] H. Khan, U. Hengartner, and D. Vogel, "Usability and Security Perceptions of Implicit Authentication: Convenient, Secure, Sometimes Annoying," in *Elev. Symp. Usable Priv. Secur. (SOUPS 2015)*, 2015, pp. 225–239. [Online]. Available: https://www.usenix.org/conference/soups2015/proceedings/presentation/khan

[70] K. Kim, W. Chang, S. Cho, and J. Shim, "Hand grip pattern recognition for mobile user interfaces," *IAAI'06 Proc. 18th Conf. Innov. Appl. Artif. Intell.*, vol. 2, pp. 1789–1794, 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1597122.1597138$\delimiter"026E30F$nhttp://www.aaai.org/Papers/IAAI/2006/IAAI06-013

[71] P. Klasnja and W. Pratt, "Healthcare in the pocket: mapping the space of mobile-phone health interventions." *J. Biomed. Inform.*, vol. 45, no. 1, pp. 184–98, feb 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1532046411001444

[72] J. W. Krakauer and R. Shadmehr, "Consolidation of motor memory," *Trends Neurosci.*, vol. 29, no. 1, pp. 58–64, 2006.

[73] S. Kratz and M. Rohs, "HoverFlow," in *Proc. 11th Int. Conf. Human-Computer Interact. with Mob. Devices Serv. - MobileHCI '09*. New York, New York, USA: ACM Press, sep 2009, p. 1. [Online]. Available: http://dl.acm.org/citation.cfm?id=1613858.1613864

[74] S. Kratz, P. Chiu, and M. Back, "PointPose," in *Proc. 2013 ACM Int. Conf. Interact. tabletops surfaces - ITS '13*. New York, New York, USA: ACM Press, oct 2013, pp. 223–230. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2512349.2512824

[75] P.-O. Kristensson and S. Zhai, "Relaxing Stylus Typing Precision by Geometric Pattern Matching," in *Proc. IUI '05*. New York, New York, USA: ACM Press, jan 2005, pp. 151–158. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1040830.1040867

[76] P. Kry and D. Pai, "Grasp Recognition and Manipulation with the Tango," pp. 551–559, jul 2006. [Online]. Available: http://www.youtube.com/watch?v=WGR16nifYss

[77] J. Kwapisz, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explor. ...*, vol. 12, no. 2, pp. 74–82, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=1964918

[78] M. Kwon, Z. Dou, W. Heinzelman, T. Soyata, H. Ba, and J. Shi, "Use of Network Latency Profiling and Redundancy for Cloud Server Selection," in *2014 IEEE 7th Int. Conf. Cloud Comput.* IEEE, jun 2014, pp. 826–832. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6973820

[79] M. Lehne, K. Ihme, A. M. Brouwer, J. B. F. Van Erp, and T. O. Zander, "Error-related EEG patterns during tactile human-machine interaction," in *Proc. - 2009 3rd Int. Conf. Affect. Comput. Intell. Interact. Work. ACII 2009*. IEEE, sep 2009, pp. 1–9. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5349480

[80] Q. Li, J. A. Stankovic, M. A. Hanson, A. T. Barth, J. Lach, and G. Zhou, "Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information," in *Proc. - 2009 6th Int. Work. Wearable Implant. Body Sens. Networks, BSN 2009*. IEEE, jun 2009, pp. 138–143. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5226903

[81] J. Liu, Z. Wang, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "uWave: Accelerometer-based personalized gesture recognition and its applications," *2009*

IEEE Int. Conf. Pervasive Comput. Commun., pp. 1–9, mar 2009. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4912759

[82] Y. Liu and D. Hatzinakos, "Human acoustic fingerprints: A novel biometric modality for mobile security," in *2014 IEEE Int. Conf. Acoust. Speech Signal Process.* IEEE, may 2014, pp. 3784–3788. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6854309

[83] G. D. Logan, "Repetition priming and automaticity: Common underlying mechanisms?" *Cogn. Psychol.*, vol. 22, no. 1, pp. 1–35, jan 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/001002859090002L

[84] I. S. MacKenzie, *Human-Computer Interaction: An Empirical Research Perspective*. Newnes, 2013. [Online]. Available: https://books.google.com/books?hl=en{&}lr={&}id=k0kBgyCaokAC{&}pgis=1

[85] J. Madison, *Damn You, Autocorrect!* Ebury Publishing, 2012. [Online]. Available: https://books.google.com/books?hl=en{&}lr={&}id=hYiD4l8S8SYC{&}pgis=1

[86] K. V. Mardia and P. E. Jupp, *Directional Statistics*. John Wiley & Sons, 2009. [Online]. Available: https://books.google.com/books?hl=en{&}lr={&}id=PTNiCm4Q-M0C{&}pgis=1

[87] P. Margaux, M. Emmanuel, D. Sébastien, B. Olivier, and M. Jérémie, "Objective and subjective evaluation of online error correction during P300-based spelling," *Adv. Human-Computer Interact.*, vol. 2012, pp. 1–13, jan 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2484487.2484491

[88] P. Mazuruse, "Canonical correlation analysis: Macroeconomic variables versus stock returns," *J. Financ. Econ. Policy*, vol. 6, no. 2, pp. 179–196, may 2014. [Online]. Available: http://www.emeraldinsight.com/doi/10.1108/JFEP-09-2013-0047

[89] A. Miller, J. Brumaghim, W. Edwin, C. Iii, A. Ehlers, B. D. Giardina, L. F. Katz, R. Heslegrave, J. Phillip, W. Iacono, H. Lyytinen, H. Nordby, H. Rau, G. Sartory, J. M. Riek, M. Catherine, and D. G. Gilbert, "The error-related negativity: an event-related brain potential accompanying errors," *Psychophysiology*, vol. 27, no. 4, p. 34, 1990. [Online]. Available: http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:The+error-related+negativity:+An+event-related+brain+potential+accompanying+errors.{#}0

[90] M. Nauman and T. Ali, "Token: Trustable keystroke-based authentication for web-based applications on smartphones," *Inf. Secur. Assur.*, pp. 286–297, 2010. [Online]. Available: http://www.springerlink.com/index/T71818XH370R3835.pdf

[91] A. Ng, "The effects of encumbrance on mobile interactions," in *Proc. 16th Int. Conf. Human-computer Interact. with Mob. devices Serv. - MobileHCI '14*. New York, New York, USA: ACM Press, sep 2014, pp. 405–406. [Online]. Available: http://dl.acm.org/citation.cfm?id=2628363.2634268

[92] J. T. Noteboom, K. R. Barnholt, and R. M. Enoka, "Activation of the arousal response and impairment of performance increase with anxiety and stressor intensity." *J. Appl. Physiol.*, vol. 91, no. 5, pp. 2093–101, nov 2001. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/11641349

[93] A. Ostberg and N. Matic, "Hover Cursor: Improving Touchscreen Acquisition Of Small Targets With Hover-enabled Pre-selection," in *Proc. 33rd Annu. ACM Conf. Ext. Abstr. Hum. Factors Comput. Syst. - CHI EA '15*. New York, New York, USA: ACM Press, 2015, pp. 1723–1728. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2702613.2732903http://dl.acm.org.emedien.ub.uni-muenchen.de/citation.cfm?id=2702613.2732903

[94] J. Park, "Plastic optical fiber sensor for measuring driver-gripping force," *Opt. Eng.*, vol. 50, no. 2, p. 020501, feb 2011. [Online]. Available: http://opticalengineering.spiedigitallibrary.org/article.aspx?articleid=1157733

[95] L. Paul, S. Wyke, S. Brewster, N. Sattar, J. M. Gill, G. Alexander, D. Rafferty, A. K. McFadyen, A. Ramsay, and A. Dybus, "Increasing physical activity in stroke survivors using STARFISH, an interactive mobile phone application: a pilot study," *Top. Stroke Rehabil.*, jan 2016. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/10749357.2015.1122266

[96] H. Pohl and R. Murray-Smith, "Focused and casual interactions," *Proc. SIGCHI Conf. Hum. Factors Comput. Syst. - CHI '13*, no. 1, p. 2223, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2470654.2481307

[97] K. K. Rachuri, M. Musolesi, C. Mascolo, P. J. Rentfrow, C. Longworth, and A. Aucinas, "EmotionSense: a mobile phones based adaptive platform for experimental social psychology research," *Proc. 12th ACM Int. Conf. Ubiquitous Comput. - Ubicomp '10*, p. 281, 2010. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1864349.1864393

[98] C. E. Rasmussen and C. K. Williams, "Gaussian Processes for Machine Learning," p. 248, 2006.

[99] I. Raso, R. Hervás, and J. Bravo, "m-Physio : Personalized Accelerometer-based Physical Rehabilitation Platform," *UBICOMM 2010 Fourth Int. Conf. Mob. Ubiquitous Comput. Syst. Serv. Technol.*, no. c, pp. 416–421, 2010.

[100] N. Ravi, N. Dandekar, and P. Mysore, "Activity recognition from accelerometer data," *Proc. Natl.*, pp. 1541–1546, 2005.

[101] J. Reason, *Human error*. Cambridge University Press, 1990.

[102] J. Rekimoto, "SmartSkin: An Infrastructure for Freehand Manipulation on Interactive Surfaces," *Proc. SIGCHI Conf. Hum. factors Comput. Syst. Chang. our world, Chang. ourselves - CHI '02*, no. 4, p. 113, 2002. [Online]. Available: http://portal.acm.org/citation.cfm?doid=503376.503397http://dl.acm.org/citation.cfm?id=503376.503397

[103] J. Rekimoto, T. Ishizawa, C. Schwesig, and H. Oba, "PreSense: Interaction Techniques for Finger Sensing Input Devices," in *Proc. 16th Annu. ACM Symp. User interface Softw. Technol. - UIST '03*. New York, New York, USA: ACM Press, nov 2003, pp. 203–212. [Online]. Available: http://dl.acm.org/citation.cfm?id=964696.964719

[104] J. Rekimoto, H. Oba, and T. Ishizawa, "SmartPad: A Finger-Sensing Keypad for Mobile Interaction," in *CHI '03 Ext. Abstr. Hum. factors Comput. Syst. - CHI '03*. New York, New York, USA: ACM Press, apr 2003, p. 850. [Online]. Available: http://portal.acm.org/citation.cfm?doid=765891.766030

[105] I. H. Robertson, T. Manly, J. Andrade, B. T. Baddeley, and J. Yiend, "'Oops!': performance correlates of everyday attentional failures in traumatic brain injured and normal subjects." *Neuropsychologia*, vol. 35, no. 6, pp. 747–58, jun 1997. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/9204482

[106] S. Rogers, J. Williamson, C. Stewart, and R. Murray-Smith, "AnglePose : Robust , Precise Capacitive Touch Tracking via 3D Orientation Estimation," in *Proc. CHI 2011*. New York, New York, USA: ACM Press, may 2011, pp. 2575–2584. [Online]. Available: http://dl.acm.org/citation.cfm?id=1979318

[107] B. Rosman, S. Ramamoorthy, M. H. Mahmud, and P. Kohli, "On user behaviour adaptation under interface change," in *Proc. 19th Int. Conf. Intell. User Interfaces - IUI '14*. New York, New York, USA: ACM Press, 2014, pp. 273–278. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2557500.2557535

[108] H. Saevanee and P. Bhattarakosol, "Authenticating user using keystroke dynamics and finger pressure," in *2009 6th IEEE Consum. Commun. Netw. Conf. CCNC 2009*. IEEE, jan 2009, pp. 1–2. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4784783

[109] H. Saevanee and P. Bhatarakosol, "User Authentication Using Combination of Behavioral Biometrics over the Touchpad Acting Like Touch Screen of Mobile

Device," in *2008 Int. Conf. Comput. Electr. Eng.* IEEE, dec 2008, pp. 82–86. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=4740951

[110] H. Salamin, A. Polychroniou, and A. Yinciarelli, "Automatic recognition of personality and conflict handling style in mobile phone conversations," in *2013 14th Int. Work. Image Anal. Multimed. Interact. Serv.* IEEE, jul 2013, pp. 1–4. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6616146

[111] P. Saravanan, S. Clarke, D. H. Chau, and H. Zha, "LatentGesture," *Proc. Second Int. Symp. Chinese CHI - Chinese CHI '14*, pp. 110–113, 2014. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2592235.2592252

[112] M. Sato, I. Poupyrev, and C. Harrison, "Touché," in *Proc. 2012 ACM Annu. Conf. Hum. Factors Comput. Syst. - CHI '12*. New York, New York, USA: ACM Press, may 2012, p. 483. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2207676.2207743

[113] A. Savitzky and M. J. E. Golay, "Smoothing and Differentiation of Data by Simplified Least Squares Procedures." *Anal. Chem.*, vol. 36, no. 8, pp. 1627–1639, jul 1964. [Online]. Available: http://dx.doi.org/10.1021/ac60214a047

[114] F. Schaub, R. Deyhle, and M. Weber, "Password entry usability and shoulder surfing susceptibility on different smartphone platforms," in *Proc. 11th Int. Conf. Mob. Ubiquitous Multimed. - MUM '12*. New York, New York, USA: ACM Press, dec 2012, p. 1. [Online]. Available: http://dl.acm.org/citation.cfm?id=2406367.2406384

[115] M. K. Scheffers and M. G. Coles, "Performance monitoring in a confusing world: error-related brain activity, judgments of response accuracy, and types of errors." *J. Exp. Psychol. Hum. Percept. Perform.*, vol. 26, no. 1, pp. 141–151, 2000.

[116] A. Schmidt, "Implicit human computer interaction through context," *Pers. Ubiquitous Comput.*, vol. 4, no. 2, pp. 191–199, jun 2000. [Online]. Available: http://www.springerlink.com/index/U3Q14156H6R648H8.pdf

[117] S.-e. C. Sensors, "CapTouch Programmable Controller for Single-Electrode Capacitance Sensors," 2009. [Online]. Available: http://www.analog.com/media/en/technical-documentation/data-sheets/AD7147.pdf

[118] J. A. Seoane, C. Campbell, I. N. M. Day, J. P. Casas, and T. R. Gaunt, "Canonical Correlation Analysis for Gene-Based Pleiotropy Discovery," *PLoS Comput. Biol.*, vol. 10, no. 10, p. e1003876, oct 2014. [Online]. Available: http://dx.plos.org/10.1371/journal.pcbi.1003876

[119] A. Serwadda and V. V. Phoha, "When kids' toys breach mobile phone security," in *Proc. 2013 ACM SIGSAC Conf. Comput. Commun. Secur. - CCS '13*. New York, New York, USA: ACM Press, 2013, pp. 599–610. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2508859.2516659

[120] X. Shang, "Grip-pattern recognition : applied to a smart gun," *J. Electron. Imaging*, vol. 17, no. 1, p. 011017, jan 2008. [Online]. Available: http://purl.org/utwente/doi/10.3990/1.9789036527323

[121] X. Shang and R. Veldhuis, "Grip-Pattern Recognition in Smart Gun Based on Likelihood-Ratio Classifier and Support Vector Machine," *Image Signal Process.*, pp. 289–295, 2008. [Online]. Available: http://www.springerlink.com/index/j2205408370g76j1.pdf

[122] N. Sharma and T. Gedeon, "Optimal time segments for stress detection," in *Proc. 9th Int. Conf. Mach. Learn. Data Min. Pattern Recognit.*, ser. Lecture Notes in Computer Science, P. Perner, Ed., vol. 7988. Berlin, Heidelberg: Springer Berlin Heidelberg, jul 2013, pp. 421–433. [Online]. Available: http://dl.acm.org/citation.cfm?id=2529191.2529223

[123] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004, no. February. [Online]. Available: http://eprints.ecs.soton.ac.uk/9580/

[124] E. Shi, Y. Niu, M. Jakobsson, and R. Chow, "Implicit authentication through learning user behavior," *Inf. Secur.*, 2011. [Online]. Available: http://www.springerlink.com/index/M57U551U3133475M.pdf

[125] Z. Sitova, J. Sedenka, Q. Yang, G. Peng, and G. Zhou, "HMOG: A New Biometric Modality for Continuous Authentication of Smartphone Users," *arXiv Prepr. arXiv ...*, no. 3, pp. 1–14, jan 2015. [Online]. Available: http://arxiv.org/abs/1501.01199$\delimiter"026E30F$npapers3://publication/uuid/E1482186-B2BE-45F5-A3E3-7766741C5934

[126] R. J. Sternberg, *Cognitive psychology.* Harcourt Brace College Publishers, 1996.

[127] H. Steven, "How Do Users Really Hold Mobile Devices?" pp. 1–24, 2013. [Online]. Available: http://www.uxmatters.com/mt/archives/2013/02/how-do-users-really-hold-mobile-devices.php

[128] C. Stewart, E. Hoggan, L. Haverinen, H. Salamin, and G. Jacucci, "An exploration of inadvertent variations in mobile pressure input," *Proc. 14th Int. Conf. Hum. Comput.*

Interact. with Mob. Devices Serv. - MobileHCI '12, pp. 35–38, 2012. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2371574.2371581

[129] J. R. Stroop, "Studies of interference in serial verbal reactions." *J. Exp. Psychol.*, vol. 18, no. 6, pp. 643–662, 1935.

[130] D. Sturman and D. Zeltzer, "A survey of glove-based input," *IEEE Comput. Graph. Appl.*, vol. 14, no. 1, pp. 30–39, jan 1994. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=250916

[131] S. Suh, K. Yi, C. Choi, D. Park, and C. Kim, "Mobile LCD device with transparent infrared image sensor panel for touch and hover sensing," in *Dig. Tech. Pap. - IEEE Int. Conf. Consum. Electron.* IEEE, jan 2012, pp. 213–214. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6161834

[132] B. Taylor and V. M. Bove Jr., "Graspables : Grasp-Recognition as a User Interface," *Chi 2009*, pp. 917–925, 2009. [Online]. Available: http://portal.acm.org/citation.cfm?id=1518701.1518842

[133] B. T. Taylor and V. M. Bove, "The bar of soap: a grasp recognition system implemented in a multi-functional handheld device," *Proc. ACM CHI 2008 Conf. Hum. Factors Comput. Syst.*, vol. 2, pp. 3459–3464, 2008. [Online]. Available: http://doi.acm.org/10.1145/1358628.1358874

[134] C. Thompson, J. White, and B. Dougherty, "Using smartphones to detect car accidents and provide situational awareness to emergency responders," *Oper. Syst.*, pp. 29–42, 2010. [Online]. Available: http://www.springerlink.com/index/g531q1x66167h620.pdf

[135] J. W. Tukey, *Exploratory Data Analysis*, 1977, vol. 2, no. 1999. [Online]. Available: http://xa.yimg.com/kq/groups/16412409/1159714453/name/ExploratoryDataAnalysis.pdf$\delimiter"026E30F$nhttp://www.springerlink.com/index/10.1007/978-1-4419-7976-6

[136] C. Vi and S. Subramanian, "Detecting error-related negativity for interaction design," in *Proc. 2012 ACM Annu. Conf. Hum. Factors Comput. Syst. - CHI '12*. New York, New York, USA: ACM Press, may 2012, pp. 493–502. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2207676.2207744

[137] C. T. Vi, I. Jamil, D. Coyle, and S. Subramanian, "Error Related Negativity in Observing Interactive Tasks," in *Proc. 32Nd Annu. ACM Conf. Hum. Factors Comput. Syst.* New York, New York, USA: ACM Press, apr 2014, pp. 3787–3796. [Online]. Available: http://dl.acm.org/citation.cfm?id=2556288.2557015

[138] S. N. Vorrink, H. S. Kort, T. Troosters, and J.-W. J. Lammers, "A Mobile Phone App to Stimulate Daily Physical Activity in Patients with Chronic Obstructive Pulmonary Disease: Development, Feasibility, and Pilot Studies." *JMIR mHealth uHealth*, vol. 4, no. 1, p. e11, jan 2016. [Online]. Available: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=4748139{&}tool=pmcentrez{&}rendertype=abstract

[139] D. L. Walters, A. Sarela, A. Fairfull, K. Neighbour, C. Cowen, B. Stephens, T. Sellwood, B. Sellwood, M. Steer, M. Aust, R. Francis, C.-K. Lee, S. Hoffman, G. Brealey, and M. Karunanithi, "A mobile phone-based care model for outpatient cardiac rehabilitation: the care assessment platform (CAP)." *BMC Cardiovasc. Disord.*, vol. 10, no. 1, p. 5, jan 2010. [Online]. Available: http://bmccardiovascdisord.biomedcentral.com/articles/10.1186/1471-2261-10-5

[140] Y. T. Wang, Y. Wang, and T. P. Jung, "A cell-phone based brain-computer interface for communication in daily life," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6320 LNAI, no. PART 2, pp. 233–240, oct 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1926560.1926597

[141] D. Weir, H. Pohl, S. Rogers, K. Vertanen, and P. O. Kristensson, "Uncertain text entry on mobile devices," in *Proc. 32nd Annu. ACM Conf. Hum. factors Comput. Syst. - CHI '14*. New York, New York, USA: ACM Press, apr 2014, pp. 2307–2316. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2556288.2557412

[142] J. Williamson, "Fingers of a Hand Oscillate Together: Phase Syncronisation of Tremor in Hover Touch Sensing," in *Proc. 2016 CHI Conf. Hum. Factors Comput. Syst.* New York, New York, USA: ACM Press, 2016, pp. 3433–3437. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2858036.2858235http://doi.acm.org/10.1145/2858036.2858235

[143] J. Williamson and R. Murray-Smith, "Rewarding the original," in *Proc. 2012 ACM Annu. Conf. Hum. Factors Comput. Syst. - CHI '12*. New York, New York, USA: ACM Press, 2012, p. 1717. [Online]. Available: http://dl.acm.org.proxy1.lib.umanitoba.ca/citation.cfm?id=2207676.2208301

[144] Wimmer and Boring, "HandSense: discriminating different ways of grasping and holding a tangible user interface," *Proc. TEI 2009*, pp. 16–19, 2009. [Online]. Available: http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=pubmed{&}cmd=Retrieve{&}dopt=AbstractPlus{&}list{_}uids=18169298137925634578related:Eko5MWRQJvwJ$\delimiter"026E30F$npapers://c80d98e4-9a96-4487-8d06-8e1acc780d86/Paper/p189

[145] R. Wimmer, "FlyEye," in *Proc. fourth Int. Conf. Tangible, Embed. embodied Interact. - TEI '10*. New York, New York, USA: ACM Press, jan 2010, p. 245. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1709886.1709935

[146] X. Xiao, T. Han, and J. Wang, "LensGesture," in *Proc. 15th ACM Int. Conf. multimodal Interact. - ICMI '13*. New York, New York, USA: ACM Press, dec 2013, pp. 287–294. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2522848.2522850

[147] C. Xu, P. H. Pathak, and P. Mohapatra, "Finger-writing with Smartwatch," in *Proc. 16th Int. Work. Mob. Comput. Syst. Appl. - HotMobile '15*. New York, New York, USA: ACM Press, feb 2015, pp. 9–14. [Online]. Available: http://dl.acm.org/citation.cfm?id=2699343.2699350

[148] S. Yazji, X. Chen, R. P. Dick, and P. Scheuermann, "Implicit user re-authentication for mobile devices," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5585 LNCS, pp. 325–339, 2009. [Online]. Available: http://www.springerlink.com/index/7361818811043t44.pdf

[149] J. Yoo, S. Lee, and C. Ahn, "Air Hook: Data preloading user interface," in *Int. Conf. ICT Converg.* IEEE, oct 2012, pp. 163–167. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6386805

[150] S. Zahid, M. Shahzad, S. Khayam, and M. Farooq, "Keystroke-based user identification on smart phones," *Recent Adv. Intrusion ...*, pp. 224–243, 2009. [Online]. Available: http://www.springerlink.com/index/J72H5012T87V3423.pdf

[151] C. Zhang, J. Tabor, J. Zhang, and X. Zhang, "Extending Mobile Interaction Through Near-Field Visible Light Sensing," in *Proc. 21st Annu. Int. Conf. Mob. Comput. Netw. - MobiCom '15*. New York, New York, USA: ACM Press, sep 2015, pp. 345–357. [Online]. Available: http://dl.acm.org/citation.cfm?id=2789168.2790115

[152] Y. Zhang, P. Xia, J. Luo, Z. Ling, B. Liu, and X. Fu, "Fingerprint attack against touch-enabled devices," in *Proc. Second ACM Work. Secur. Priv. smartphones Mob. devices - SPSM '12*. New York, New York, USA: ACM Press, oct 2012, p. 57. [Online]. Available: http://dl.acm.org/citation.cfm?id=2381934.2381947

[153] Y. Zhao, P. H. Pathak, C. Xu, and P. Mohapatra, "Demo: Finger and Hand Gesture Recognition using Smartwatch," in *Proc. 13th Annu. Int. Conf. Mob. Syst. Appl. Serv. - MobiSys '15*. New York, New York, USA: ACM Press, may 2015, pp. 471–471. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2742647.2745922