2015

Detecting Intruders via Touchscreen Usage Patterns on a Smartphone

Yu Qiu
Bard College

Detecting Intruders via Touchscreen Usage Patterns on a Smartphone

A Senior Project submitted to
The Division of Science, Mathematics, and Computing
of
Bard College

by
Yu Qiu

Annandale-on-Hudson, New York
May, 2015

Abstract

Modern phone, tablet, and computer interfaces are very sensitive to a wide variety of patterns made by their owners...and sometimes others! A mock smartphone interface was created and used to collect touch-related button clicking events from 23 different phone users in the application context of three task types: clicking on a single button, dialing phone numbers and typing words on a virtual keyboard. Different methods of classification (naive Bayes and support vector machine) were evaluated to determine whether they could effectively discriminate one user's touch from that of other persons ("intruders") based on a relatively small quantity of data from each person collected over a period of a few minutes. The best classification method could distinguish the intruder from the owner about 70% of the time.

Contents

List of Figures

Dedication

To my parents and Y.Wang

Acknowledgments

I would like to express my gratitude to my adviser Prof. Sven Anderson, for his guidance throughout this study. I would also like to thank all professors in Bard computer science department: Prof Robert McGrail, Prof. Rebecca Thomas, Prof. Keith O'Hara and again Prof. Sven Anderson.

I would like to thank my friends and my parents, without whom I would not become who I am.

1
Introduction

1.1 Background

Mobile phones have become an important part of daily life. They contain more personal data than computers and other digital devices and there is a growing need to secure these data. If other people gain access to one's phone, they also have access to your personal messages, photos and even bank accounts. Many improvements to authentication process have been made, such as Apple's Touch ID and two-factor authentication widely adopted by mobile applications.

The authentication methods we encounter in daily life are usually explicit authentication. The most eminent example is the passwords to your accounts. On the other hands, passive authentication is another technology that could improve the security of human-computer interfaces. It detects the anomalies in usage patterns, even after the explicit authentication procedure. When the application monitor the usage of your device, and once there is an anomaly in the user's behavior, the application decides the account is intruded, and asks another explicit authentication.

Different users have different usage pattern when interacting with the touchscreen. Some characteristics of the user may be expressed unconsciously as they move among and use different applications. In this project, we try to perform passive authentication (anomaly detection) with the pattern of touchscreen usage, and examines that if the pattern of touchscreen usage contains enough information to deducing differentiate the owner, a frequent user, from others who may briefly attempt to access the phone's applications and data.

Successful researches on this question have utilized additional information, such as electrical properties of the human body or proximity data. In this project, we confined ourselves to the information obtained from the Android system on a common mobile phone Nexus 6 with no special hardware or systems software.

The project focuses on the usage pattern in one frequent interaction type, button clicks. Continuous movements on touchscreen, such as swiping, pinching and drawing, would provide much more information about the user. For example, the length and curvature of the zooming gesture and swipes reflects the hands shape of the users (Feng et al., 2014). But the environment which need security the most includes texting, sending emails and making phone calls. In these scenarios, users mostly interact with the phone by clicking on virtual dial pads and on virtual keyboards. Therefore, by focusing on clicking behaviors, we also examine the possibility of applying the passive authentication techniques in these contexts, since most data that could be gathered in these scenario are clicks on the touchscreen.

1.2 Previous Research

Harrison et al. used a special touchscreen, and showed that the electrical properties of humans and their attire were very useful information for user identification. The special touchscreen measures impedance of a user to the environment across a range of AC fre-

quencies. The researchers built an impedance profile for each user. They used support vector machine as the classifier, and in the experiment to identify the touch between pair of participants, they reached an average accuracy of 97.8% when 200 frequencies (between 1KHz and 3.5MHz) were used. They achieved a high accuracy with the extra information of the user's body impedance. However, a special touchscreen is necessary to collect this type of information (Harrison et al., 2012).

Kolly et al. performed large scale study to research the users' touch screen behavior on standard user interface elements, by publishing a quiz game that logs touch data. Their data were gathered from $14,000$ users around the world. With a sample set of this size their study provides a basic idea about whether touch data is sufficient for user identification in practice. The features they extracted are the mean and the maximal pressure, the minimal and the maximal gradient of the pressure, the hold time of a touch event, the mean X and Y position (relative to the center of the touch element), and the variance of the touch event in X and Y direction. Using a naive Bayes classifier to classify a set of gestures from one unknown user, they could identify a specific user from a set of 5 users, with a precision of about 80%, and in 2 users with a precision of 94%. Their research shows that it is possible to perform user identification based only on users' touch screen behavior, though the number of different users was quite limited (Kolly et al., 2012).

Feng at el. developed a touch-based identity protection service. It incorporated contextual app information. They implemented a background service on several Android devices to implicitly collect touch screen usage data. Biometric features, such as swipe/zoom speed, click gap, contact size, touch location and swipe/zoom length curvature, were extracted from the touch event data. Along with context information, these features of a touch were first put into a filtering decision tree. Only touches with the same running application, similar direction, swipe/zoom length, curvature and slope were compared to each other, using dynamic time warping in order to measure the similarity between two time sequences.

Dynamic time warping computes the Euclidean distance between any two input sequences of feature vectors and finds the optimal sequence alignment using dynamic programming. The label of the new touch was produced by one nearest neighbor classifier. They achieved a 91% true positive rate and a 93% true negative rate in an uncontrolled application context. The classification procedure of this system heavily relies on continuous gestures like swiping and zooming, as many of the features are from continuous gestures. It may work not as well in a context of discrete events such as button clicks (Feng et al., 2014).

Zaliva et al. Incorporated proximity data into the user identification task. They used a Samsung Galaxy S4 mobile phone equipped with a touch-screen sensor by Synaptics, and wrote a data collection daemon which requires a modified kernel based on CyanogenMod. Artificial neural network and support vector machines were used in the classification. The feed-forward neural network with two hidden layers with 50-75 and 30 neurons respectively, and a logistic sigmoid activation function at the output layer, achieved a precision of 99.8% in their anomaly detection experiment. The support vector machines only got a 0.67 F1-score. They also develop a sequential classification approach using a probability ratio test of artificial neural network outputs. The overall result shows that after 5 touches, or in 12.6 seconds on average, they correctly distinguish the primary user from any of 14 other known users. Their research overcomes the lack of continuous data on clicking gestures by collecting proximity data, since proximity data restores the whole trajectory of the fingers. However, only a few models of mobile phones have the sensor for proximity data, and obtaining it requires modification at the kernel level. In this project, we explore the possibility to perform user identification from button clicks, without proximity data (Zaliva et al., 2015).

There are quite a few researches in this area these years. The four above are the ones that relates to and inspired this project. These researches shows that touchscreen usage pattern, and other information helps to perform user identification and anomaly detection.

They also demonstrate several ways to extract features from touchscreen usage data, and to do the classification.

2
Capturing Touchscreen Usage Patterns

The application of machine learning to user discrimination depends on a reasonably sized data set collected from many different people. In this project, in order to get data on touchscreen usage pattern, an experiment was designed, and participants were recruited to perform some tasks in a mobile application. Because of the time constraint, implementing a daemon by modifying the kernel (in (Zaliva et al., 2015)) or crowdsourcing user study (in (Kolly et al., 2012)) were not adopted as the data collection method for this project.

In the first part of this study, we build a mock application that allows us to collect detailed data on finger movements during use of a touchscreen usage. Participants were recruited on campus to perform specified tasks in a customized Android application on a Nexus 6 mobile phone running the Android 5.01 operating system. The tasks were designed to focus on one of the most common interactions: button clicks. The application gathered data while the participants performed the tasks.

2.1 Extracting Touch Events

The data collection application used only the standard Android SDK. Only collecting data that is available in Android system simulates the restricted access to data that future applications of this project would have.

MotionEvent is the Android class used to report movement events. For example, when the user first touches the screen, the system delivers a touch event to the appropriate View with the action code ACTION_DOWN and a set of axis values that include the X and Y coordinates of the touch and information about the pressure, size and orientation of the contact area (Developer.android.com, 2015). MotionEvent objects pack several history data points in a batch for efficiency; in the application, the history is unpacked in order to capture all data.

The data collected from each Android MotionEvent includes these attributes

- The type of the MotionEvent. The ones we are interested in are ACTION_DOWN, ACTION_MOVE and ACTION_UP
- The time of the motion, in milliseconds, which is the non-sleep uptime since boot
- The ID of the pointer
- The absolute X and Y coordinates of the motion
- The X and Y coordinates of the View from which the motion is dispatched.
- The pressure of the down motion, between 0 and 1, with some exceptions
- The contacting size of the touch in the down motion, normalized by Android system, between 0 and 1
- Orientation, length of major axis and length of minor axis, of the eclipse that represents the touch

The data from MotionEvent objects are collected by overloading the onTouchEvent() method of each Activity and each View. The MotionEvent object dispatched from a view have coordinates relative to the position of the view, so the absolute X and Y are calculated by the application.

The onClick() method in View.OnClickListener interface does not provide access to the MotionEvent objects. Instead of using OnClickListener to implement the application logic, a substitute of OnClickListener is implemented to process clicks and record data from the touch.

2.2 Task Design and Implementation

The tasks for the participants to perform focus on clicks, rather than swipes or other continuous motion, because continuous gestures are rare in certain application context, though they could provide more features for classification. The three tasks are

1. clicking a single button
2. dialing phone numbers
3. typing words on a virtual keyboard

These tasks simulate the common scenario where clicking on a touchscreen occurs, while continuous gestures, such as swiping and zooming, seldom occur. The data was made fully anonymous according to procedures approved by the Bard institutional review board.

All tasks consist of several repetitions, where the participants perform the the task on different content. The ordering of the tasks is the same, but the ordering in which the content of each repetition of the task appears is in random. Each task activity provides a SKIP button, at the bottom of the screen, that can be used to skip any repetition of the tasks at anytime. When a repetition of the tasks is finished the SKIP button turns into a NEXT button. The participants must use this button to go to next round of the tasks.

2.2.1 Single Button Click Task

The first task is to click on a single large button several times. The button is placed at the center of the screen, to simulate a pop-up window (or Fragment in Android) for confirming in many mobile applications.

The number of button presses in this task's instruction is 2, 3, or 5. Double clicks are common in a desktop computer interfaces, since the input device for clicking is a mouse. However, consecutive clicks are rarely used as another dimension of interaction on mobile devices. This additional type of clicking is provided by long clicks in Android. Multiple clicks of a button are interpreted as many separate clicks, consistent with most Android applications. The instruction for the participants to click multiple times primarily collects more input data than a single click and makes the task more interesting.

The number of the remaining clicks to be made is presented to the participants as the text on the button, and it is updated when the participant clicks. When the participants finish a round of this task, the center button also turns into a NEXT button. Figure 2.2.1 is a screenshot of this task.

2.2.2 Dial pad task

The second task simulates dialing a specific phone number. The dial pad mocks the layout of the dial pad in the Android 5.0.1 built-in Google Dialer version 2.0 (Figure 2.2.2), except the position of the delete button. This customized dial pad also does not have the letters under the numbers. However, it resembles the common dial pad layout model enough to ensure the quality of data.

The phone numbers for the participants to dial are my phone number, Bard College's number and Bard College at Simon's Rock's number on their websites. The numbers provide a variety of number combinations. Some of them have the area codes local to

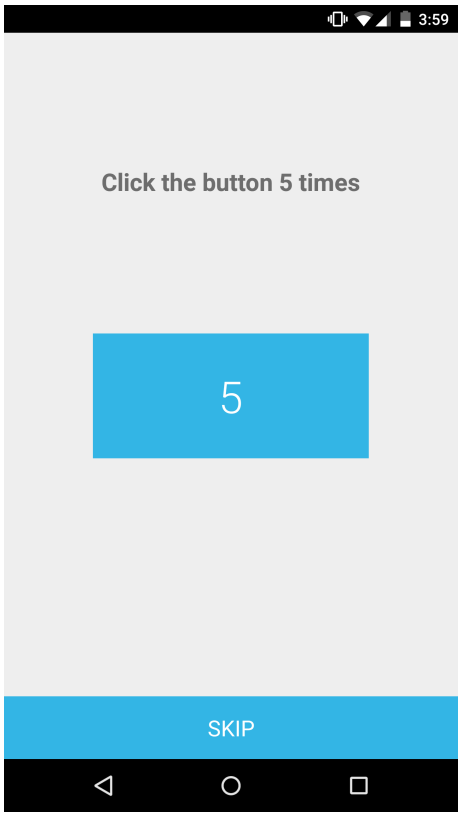

Figure 2.2.1: Screenshot of the single button clicking task

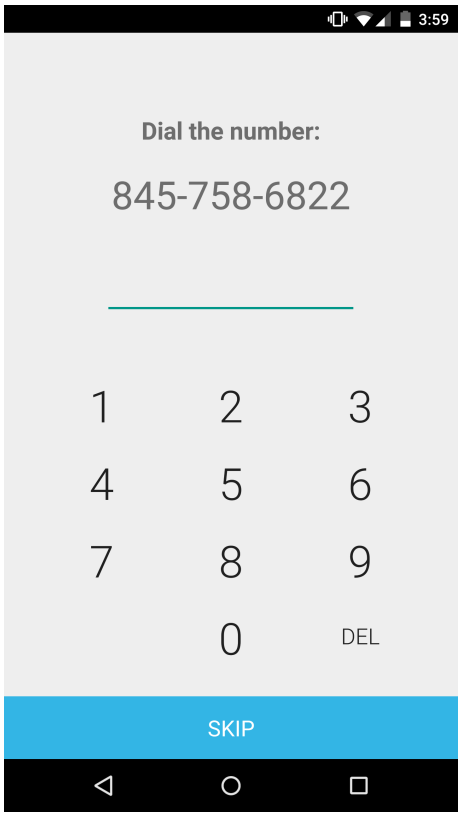

Figure 2.2.2: Screenshot of the dial pad task

Bard that participants recruited on campus might be familiar with and the area codes that they are unfamiliar with.

A click on the number buttons will immediately appear on the EditText that displays the numbers dialed, above the dial pad and below the prompt phone number. After the phone number has been dialed correctly, the dial pad and the EditText are locked. If a participant continues clicking on the dial pad, no new numbers will be entered, except the information of these clicks is also logged.

2.2.3 Keyboard Input Task

The third task is to type a specific word using a virtual keyboard. The virtual keyboard is customized in order to record all MotionEvent information. Each key is modeled by a separate button. The layout of the keyboard is made to be like the Android built-in Google Keyboard 4.0, which is a QWERTY keyboard. The customized keyboard only supports typing by clicking each key, and only the English alphabet can be input. The delete button is at the bottom-right corner, appearing in red.

The prompted words for this task are “BARD”, “APPLE”, “GOOGLE”, “TIME” and “HACKER”. The first four words should be more familiar to the participants. “TIME” is picked because it is very commonly used, as it ranks 55 in the 100 most common English words found in writing around the world (Oxforddictionaries.com, 2015). “HACKER” is picked as a less common word.

Every character that a user types appears in the EditText display above the keyboard and below TextView of the prompted word. After the word is entered correctly, the keyboard is disabled, in the same way as in the dial pad task, until the participant uses the NEXT button to go to next round.

Figure 2.2.3 is a screenshot of this task.

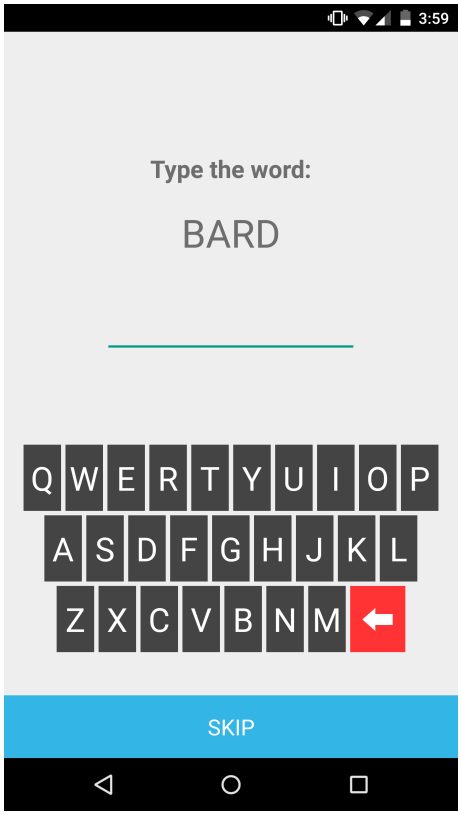

Figure 2.2.3: Screenshot of the single button clicking task

2.3 Summary

The three tasks above simulates the environment where users clicks on the buttons. The application logs every MotionEvent the application receives to provide raw data of clicks to investigate in this project.

3
Features and Their Distribution

3.1 Feature Extraction

We used machine learning techniques, more specifically classifiers to predict the identity of users to whom a click belongs. Therefore raw data of MotionEvents needed to be translated into vectors representing clicks. Some features were extracted from the raw data described in Chapter 2.

In our model, each click generated two MotionEvents, one of the type ACTION_DOWN and the other of the type ACTION_UP, with possible ACTION_MOVE events between the pair. This interpretation of click allows us to extract several features of clicking from the raw data:

- The duration of the click computed from the difference in time between the MOTION_DOWN event and the MOTION_UP event, in millisecond
- The absolute coordinates of the down motion, in pixels
- The relative coordinates to the view dispatching the MotionEvent, in pixels

- The difference of X and Y coordinates, computed from the difference in the absolute coordinates between the MOTION_DOWN event and the MOTION_UP event, in pixels

- The pressure of the down motion, between 0 and 1

- The contact size of the touch in the down motion, normalized by Android system, between 0 and 1

Many applications interpret a click as a down motion of a touch. The Android API Guide writes, "this (onClick()) is called when the user either touches the item (when in touch mode), ... and presses the suitable 'enter' key or presses down ..." (Android, 2015). However, there is often a movement during clicking. Although this is irrelevant to most applications, we consider that this motion includes some information about the user's touching style. The difference of absolute coordinates in the feature set is used to represent this motion.

The ability to capture and process touch inputs varies among hardware devices. Some phones may use one number as the pressure and size at the same time. However, the Nexus 6 used for data collection differentiates between these two attributes, providing another dimension. The Nexus 6 represents the touch motion as a circle rather than as an eclipse, thus the lengths of two axes are the same. The length of axis and touch size have very similar distributions; consequently, the former is omitted from further discussion.

The absolute coordinates of the click are largely determined by the coordinates of the button. However, since participants may react differently to buttons located at different positions on the screen, the absolute coordinates are included as features. The relative coordinates of the click are not normalized with respect to the size of the button, because participants may also react differently to buttons of different size.

3.2 Feature Statistics

3.2.1 Basic Statistics

Table 3.2.1 shows the basic statistics for the features that were collected. It includes the minimum, the maximum, the mean and the standard deviation of each feature.

Feature	Unit	Min	Max	Mean	Standard deviation
Δt	milliseconds	27	158	83.445	21.132
ΔX	pixels	-38	32	-0.015	3.23
ΔY	pixels	-48	36	-0.01	3.253
Relative X	pixels	1	775	221.291	186.996
Relative Y	pixels	3	329	153.565	49.254
Absolute X	pixels	133	1361	753.996	313.62
Absolute Y	pixels	1122	2109	1583.871	227.113
Pressure	none	0.225	1.15	0.526	0.157
Size	none	0.004	0.043	0.016	0.007

Table 3.2.1: Statistics about features

Despite some outliers, almost all movement values (ΔX and ΔY) are 0, which means that most participants did not move during clicks. These two features are unlikely to be useful in classification.

3.2.2 Distributions

These features are used in different classifiers in Chapter 4, and these classifiers have some assumptions about the distribution of the data they models. Therefore, the distribution of each feature indicates if it is suitable for the classifiers. Also, the individual distribution of each participants indicates the difficulty to differentiate them, and the usefulness of each feature in distinguishing users. For example, a feature on which each participant has a unique distribution, may be more useful to classifiers.

The distribution of each feature was plotted as a whole, along with each participant's histogram of that feature. The individual histograms were plotted as in Figure 3.2.1. Each row in the figure is a bird's-eye view of the each participant's histogram on the

particular feature. Darker squares indicate that more data fall into that range. Although each participant may have different clicks, the variance is small, since the tasks they are prompted to perform are the same. Therefore, this figure indicates the variation in the disttributions of measured features.

Figure 3.2.1 shows the Δt distribution of each participant. Each individual distribution of Δt is quite different from others, suggesting that the duration of the click could be a useful feature for classification.

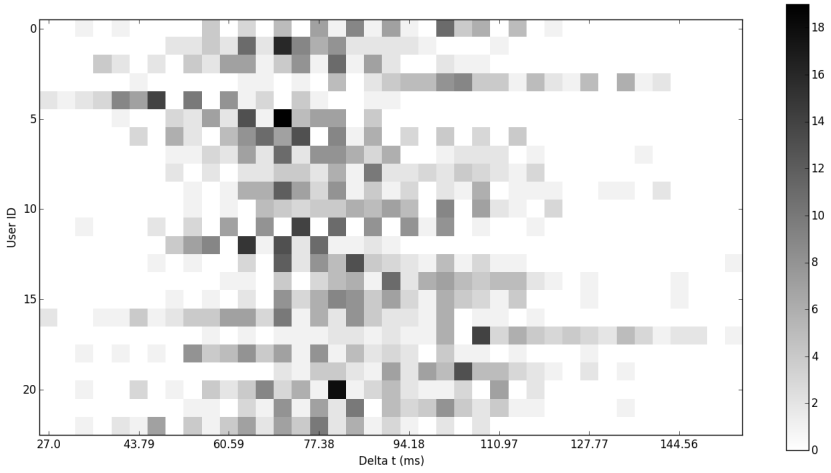

Figure 3.2.1: Distribution of Δt (the pressing time of the click) of each individual

The individual absolute x-coordinate distribution in Figure 3.2.3 confirms the observation that the absolute coordinates of the click are largely determined by the position of the button, as most of them tend to be in the three clusters in similar ranges. The three clusters are also obvious in the overall distribution 3.2.3. This clustering behavior resulted from the task design in data collection procedure. Some outliers might represent typing mistakes.

The pressure features seem to have distributions that better distinguish users, as shown in Figure 3.2.5. Thus, they may be more useful to a classifier.

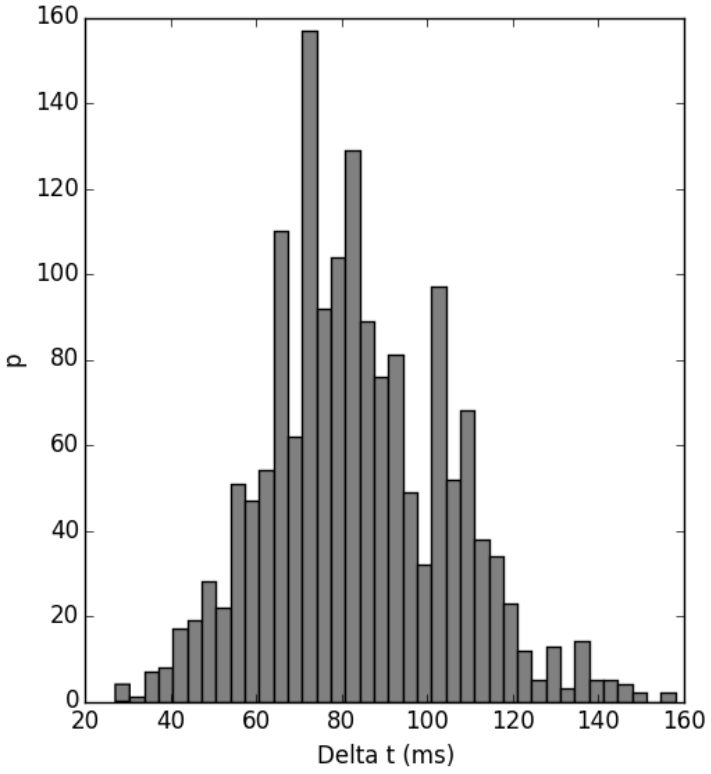

Figure 3.2.2: Overall distribution of Δt

The distribution of contact size in Figure 3.2.8 and Figure 3.2.7 indicates that its values are discrete. A careful look at the individual distributions (Figure 3.2.8) suggests that contact size does not distinguish users better than pressure (Figure 3.2.5), but it is more useful than absolute x-coordinates (Figure 3.2.3). However, its small number of different discrete values makes it less informative for the goal of classification, because it has larger granularity, and it would be treated as a continuous variable by the Gaussian naive Bayes classifier in Section 4.1.

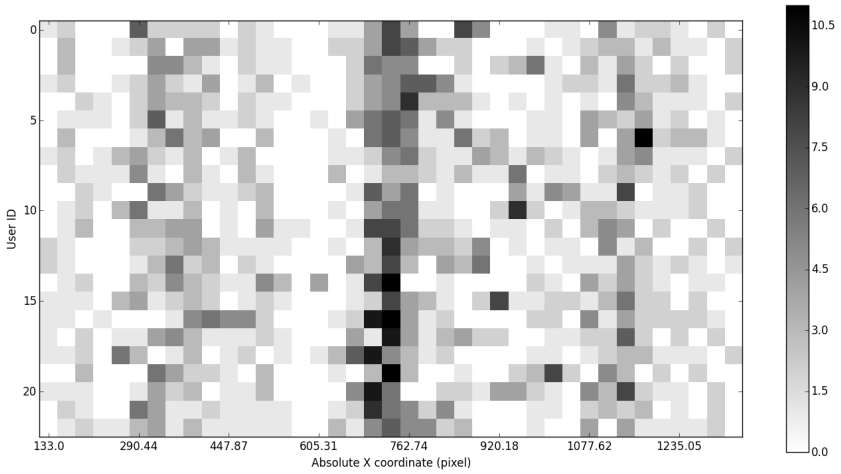

Figure 3.2.3: Distribution of absolute X coordinate of each individual. Three clusters are resulted from the task design.

3.2.3 Chi-Squared Tests

As shown in section 3.2.1 Figure 3.2.2 and Figure 3.2.6, some of the features have distributions that appear to be nearly normal. A Pearson's chi-squared test was performed to determine which features are consistent with a normal distribution.

The chi-squared value

$$\chi^2 = \sum_i \frac{(O_i - E_i)^2}{E_i}$$

where O_i is the number of the observation of type i, and E_i is the number of expected frequency of type i (Wikipedia, 2015). If the observation conforms to the expected value for a normal distribution, this value will be small. The χ^2 value is compared to a chi-squared distribution with a degrees of freedom of the number of observed frequencies minus the expected distribution's degree of freedom, in order to get the p-value of the test to determine whether the null hypothesis, that the distribution is normal, should be rejected.

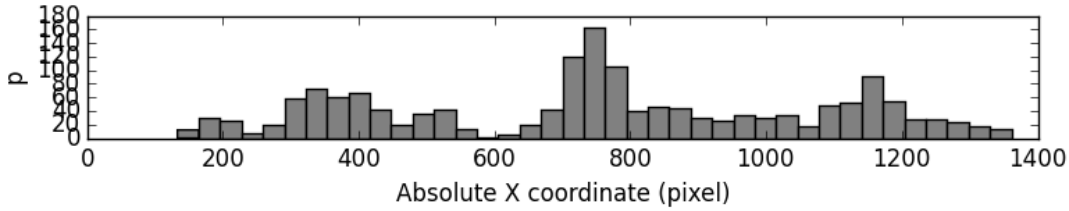

Figure 3.2.4: Overall distribution of absolute x-coordinate

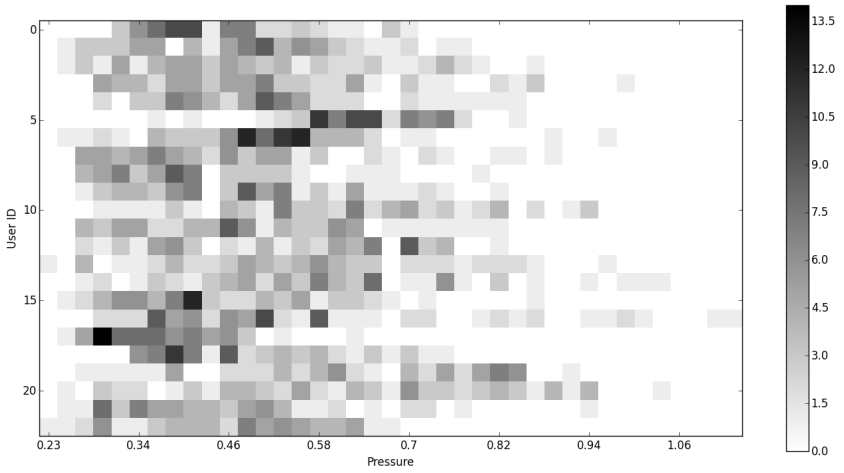

Figure 3.2.5: Distribution of pressure of each individual

The chisquare function in the Scipy library was used to perform this test (Docs.scipy.org, 2015), assuming the expected distribution is a normal distribution with the same mean and standard deviation as the feature data. The chi-squared test is affected by the number of observations. For example, when $O_1 = 5, O_2 = 15, E_1 = 10, E_2 = 10$, the χ^2 of the data is 5, while when $O_1 = 50, O_2 = 150, E_1 = 100, E_2 = 100$, the χ^2 is 50. Their χ^2 scores are different, even though their distributions are the same. Therefore, the number of observations for each O value was scaled proportionally, in order to make the total

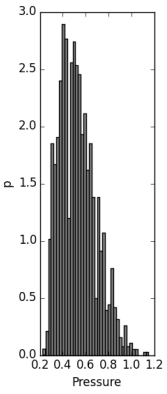

Figure 3.2.6: Overall distribution of pressure

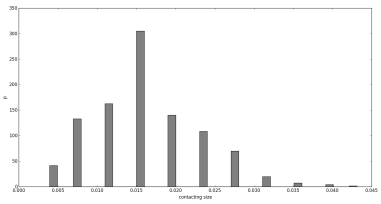

Figure 3.2.7: Overall distribution of contacting size

observation of every chi-squared tests used in this project the same, for the consistency of further comparison (such as between overall and individual distribution). The data is put into 10 bins in order to perform the chi-squared test.

Results of the χ^2 test are shown in Table 3.2.2.

Feature	χ^2	Average χ^2 by participants
Δt	24.19	195.79
Relative X	772.27	1492.48
Relative Y	241.65	414.66
Absolute X	247.12	511.05
Absolute Y	625.98	1024.61
Pressure	50.13	299.85
Size	743.73	804.30

Table 3.2.2: The χ^2 score for each feature, all $p < 0.01$

None of the features are normally distributed, as all p-values are close to 0. However, some of the features, such as, Δt, the pressure and the relative Y coordinates, have distributions more similar to normal distributions than other features. The high χ^2 score of the contacting size feature might be a result of its discrete nature, as shown in Figure 3.2.8 and Figure 3.2.7.

Table 3.2.2 shows that the relative y-coordinate has a low χ^2 score while the relative x-coordinate has a high χ^2 score. The reason for this might be that the height of the buttons

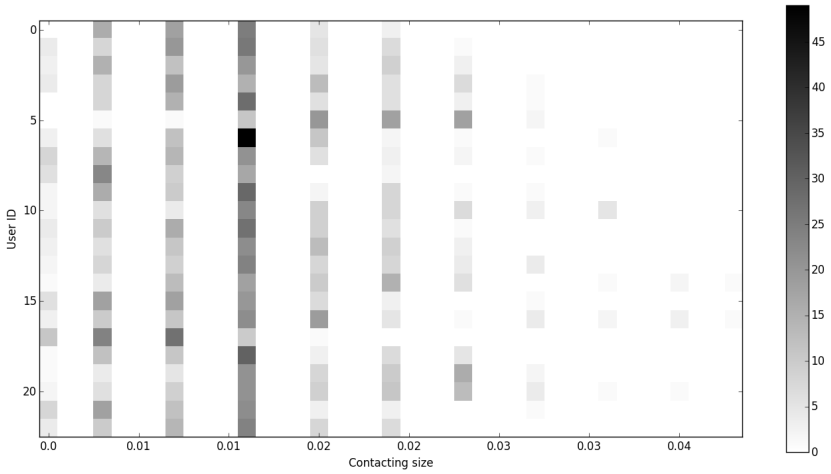

Figure 3.2.8: Distribution of contacting size of each individual

in the data collection application has little difference, but the width of the buttons varies. This situation appears in reality, and a button whose width is smaller than its height is rare.

Besides providing insights, the distance between each feature's distribution and a normal distribution, measured by the chi-squared test is relevant to the assumption of Gaussian naive Bayes classifier. (Section 4.1) The classifier assumes each feature has a normal distribution, and calculates the probability according to the normal distribution. Therefore, the non-normal distribution of features may affect the accuracy of the Gaussian naive Bayes classifier.

3.3 Summary

The raw data of MotionEvents were translated into features that represents a click, as described in Section 3.1. From the distribution of features 3.2.2 and the chi-squared test, we found that some features, such as the pressing time Δt and pressure, had more distinguishable distributions among individual participants than absolute coordinates. The

variables Δt and pressure also have the lowest χ^2 score with respect to a normal distribution, so these features would not make so much deflection to the Gaussian naive Bayes classifier as other features.

4
Intruder Detection via Classification

The objective of classification in this context is to detect the intruders from the real owner based on limited data. A classifier can seek to provide user identity (one of N) or to label a click event into one of two classes: the owner class or an intruder class. In most partitions of the training and the testing data, the clicks are shuffled, so no information concerning the clicking sequence is incorporated.

We used naive Bayes classifiers and support vector classifiers, because they are commonly used in similar classification tasks. They are also used in previous researches discussed in Section 1.2.

4.1 Naive Bayes Classifier

4.1.1 How It Works

The naive Bayes classifier is the classifier used in Kolly paper (Kolly et al., 2012). Naive Bayes classifier is based on Bayes' theorem and presumed independence of the variables of interest. Given a vector X representing feature values, it will choose the label Y that

has the maximum probability $P(Y|X)$. In this case, the vector X represents the features of a click, and the label Y represents the identity of the user.

In this context, Bayes' theorem is

$$P(Y \mid X) = \frac{P(X,Y)}{P(X)} = \frac{P(X \mid Y)P(Y)}{P(X)}.$$

Applying Bayes' theorem to $P(Y = y_i \mid X = x_k)$ yields

$$P(Y = y_i \mid X = x_k) = \frac{P(X = x_k \mid Y = y_i)P(Y = y_i)}{P(X = x_k)} \tag{4.1.1}$$

where the denominator $P(X = x_k) = \sum_j P(X = x_k \mid Y = y_j)P(Y = y_j)$.

X is a vector, so let $X = \langle X_1, X_2, \ldots, X_n \rangle$. We need to know $P(X_1, \ldots, X_n \mid Y)$ to compute $P(Y \mid X_1, \ldots, X_n)$. To simplify the computation, a strong assumption of conditional independence is made.

Recall that X is conditionally independent from Y given Z if and only if

$$(\forall i, j, k) P(X = x_i \mid Y = y_j, Z = z_k) = P(X = x_i \mid Z = z_k)$$

Then

$$P(X_1, X_2 \mid Y) = P(X_1 \mid X_2, Y)P(X_2 \mid Y) = P(X_1 \mid Y)P(X_2 \mid Y),$$

and we see that

$$P(X_1, \ldots, X_n \mid Y) = \prod_{i=1}^{n} P(X_i \mid Y).$$

Equation 4.1.1 becomes

$$P(Y = y_k \mid X_1 \ldots, X_n) = \frac{P(Y = y_k) \prod_i P(X_i \mid Y = y_k)}{\sum_j (P(Y = y_j) \prod_i P(X_i \mid Y = y_j)}$$

The denominator does not depend on y_k, so Y is the y_k which produces the largest value of $P(Y = y_k) \prod_i P(X_i \mid Y = y_k)$ (Mitchell, 1997).

These probabilities could come from the statistics of the training set if the values of X were discrete.; however, the features of clicks are continuous. To deal with continuous

data, we assume that the input data is randomly drawn from an underlying distribution. The distribution used in the classifier for out purpose is the Gaussian distribution. In the Gaussian naive Bayes model,

$$P(X = x_i \mid Y = y_k) = \frac{1}{\sigma_{ik}} \exp(\frac{-(x - \mu_{ij})^2}{2\sigma_{ij}})$$

where μ_{ij} is the mean of X_i given $Y = y_k$ and σ_{ij} is the standard deviation of X_i given $Y = y_k$ (Hill et al., 2006).

4.1.2 User Identification

Our first classification task used a naive Bayes classifier to decide which user performs each the click, using all the features except the task ID. Given values of the features, the naive Bayes classifier predicts the user ID from the features. The user ID is the one with the greatest probability for a click to have that ID given the feature data.

Naive Bayes classifiers assume that features are independent from each other. However this is not true in our data. The absolute coordinates and the relative coordinates are not completely independent, since if the button is the same, larger absolute coordinates values would be associated with larger relative coordinates values, and vice versa. The pressure and the contact size of the click are also not completely independent, since pressing harder would result in larger contact size, though it is possible to have a small contact size and a large pressure.

Most of the features are continuous, except the contact size. Therefore the Gaussian naive Bayes classifier is used to deal with continuous data. This assumes that values of one feature are normally distributed, an assumption that is relatively accurate on some features, like the time duration Δt and the pressure, as shown in Section 3.2.3. However, this assumption is inaccurate on most features which relate touch coordinates, as they depend on the position and the size of the button. Nonetheless, we first use Gaussian naive Bayes classifier with all features except the task ID, to get a baseline score.

For evaluation 10-fold cross validation is adopted. The data of each participant is shuffled and then the data is evenly divided into 10 groups. Each group has 10% of each participant's data. At each time, one of the groups is the testing set, and the other nine are used to train the model.

In this experiment, each click motion are classified into 23 categories, representing the 23 participants of the study. The overall accuracy is 16.58%, which is about 4 times better than chance performance (4.35%).

To address the independence assumption of naive Bayes classifier and the normal distribution assumption of Gaussian naive Bayes classifier, we ignore all the features involving coordinates. Another run with Gaussian naive Bayes classifier with the same training and testing schema was conducted. Only three features: the pressing time Δt, pressure and contacting size were considered.

Abandoning the features involving coordinates is reasonable in practice. Consider some future application addressing the same intruder detection problem as in this project. It might, for example, run as a daemon to monitor the system. The users might not want to share the absolute coordinates of the clicks, due to privacy concerns, and relative coordinates are hard to get from other applications on the device. Consequently, we tried to perform the classification tasks with only the Δt, pressure and size features.

This time, the accuracy of classification is 17.88%. This change in result is within the range of random variation. The gain of using features according with the assumptions is counterbalanced by the loss of information in the ignored features.

Experiment	Correct	Wrong	Total	Percentage Correct
All features	268	1348	1616	16.58%
Without coordinates features	289	1327	1616	17.88%

Table 4.1.1: Detailed results of user identification experiments with different feature sets

4.1.3 Anomaly Detection

Detecting if a different person is using the device than the true owner, is called anomaly detection (Kolly et al., 2012). In the context of this project, anomoly detection is based on the features of the clicks on touchscreen. A binary classification was conducted in order to simulate the task of anomaly detection. The clicks were re-labelled to be one participants against all others. We picked each user as the owner of the phone, his/her clicks were labelled "1", and the labels of others' clicks were "0". Then 90% of each category was used as training data, and the other 10% was testing data.

This procedure is repeated 10 times for each participant, because the testing set is small and the accuracy subject to random fluctuation. The overall accuracy is 48.44%, which is worse than the 50% accuracy of randomly guessing. In fact, the classifier has a tendency to classify all clicks into the same category.

Because the participant chosen as the owner has far fewer clicks than all other participants together, guessing that all clicks are not from the owner is leads to higher accuracy. We modified the testing set, in order to have same number of clicks from the owner and from others. The result is more reasonable than the previous test. The accuracy of this experiment is 52.35%.

The biased classification may arise from the unbalanced quantity of data used in training. The training set also had far fewer clicks from the owner than from others. We balanced the training data, by only using same amount of data with each label. After the training routine was modified, the performance of the classifier improved to 54.83%

Unlike the user identification experiments in Section 4.1.2, in this binary classification, using the feature set with only Δt, pressure and contact size, would make an improvement on the result. Training and testing with same number clicks from the owner and from others, and using only these three features, the classifier has an accuracy of 65.61%, though

Experiment	Correct	Wrong	Total	Percentage Correct
Unbalanced training set and testing set	18091	19259	37350	48.44%
Unbalanced training set, balanced testing set	1808	1632	3440	52.35%
Balanced training set and testing set	1886	1554	3440	54.83%

Table 4.1.2: Detailed results of anomaly detection experiments with all features and different training set and testing set arrangements

without the information about coordinates. In this problem, the additional information provided by the features involving coordinates helped to separate the clicks of the two categories, rather than bringing noise to the data because of its non-normal distribution.

Experiment	Correct	Wrong	Total	Percentage Correct
All features	1886	1554	3440	54.83%
Without coordinates features	2257	1183	3440	65.61%

Table 4.1.3: Detailed results of anomaly detection experiments with balanced training and testing sets and different feature sets

4.2 Support Vector Classifier

4.2.1 How it works

Support vector classifiers are powered by support vector machines. It is adopted as the classifier in Harrison's research and Zaliva's. SVMs performed very well with the human impedance data, with an accuracy of 97.8% in classifying into two users in Harrison's paper (Harrison et al., 2012), but not so well in Zaliva's research, where SVMs only got a 0.67 F1-score in anomaly detection experiment (Zaliva et al., 2015).

Support vector machines construct a maximum margin separator. The algorithm finds the hyperplane that separates two classes with the largest possible distance to example points, as the decision boundary (Russell and Norvig, 2010).

The separator is a hyperplane that divides the space into two according the sign of the linear discriminant function:

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b.$$

SVM classifiers finds the decision boundary with the maximum margin. The margin is defined as

$$m_D(\boldsymbol{w}) = \frac{1}{2}\hat{\boldsymbol{w}}^T(\boldsymbol{x}_+ - \boldsymbol{x}_-), \tag{4.2.1}$$

where the $\boldsymbol{x}_+$ is the closest point to the hyperplane among the positive examples, $\boldsymbol{x}_-$ is the closest point among the negative examples, and $\hat{\boldsymbol{w}}$ is the unit vector in the direction of $\boldsymbol{w}$ (Carugo and Eisenhaber, 2009).

Let the hyperplane be equidistant from $\boldsymbol{x}_+$ and $\boldsymbol{x}_-$, we see that,

$$f(\boldsymbol{x}_+) = \boldsymbol{w}^T\boldsymbol{x}_+ + b = a, \tag{4.2.2}$$

$$f(\boldsymbol{x}_-) = \boldsymbol{w}^T\boldsymbol{x}_- + b = -a, \tag{4.2.3}$$

for some constant $a > 0$. Equation 4.2.1 minus Equation 4.2.2 yields

$$\boldsymbol{w}^T(\boldsymbol{x}_+ - \boldsymbol{x}_-) = 2a. \tag{4.2.4}$$

(Carugo and Eisenhaber, 2009)

Multiplying the data points by a fixed number will increase the margin by the same amount, but the position of the margin has not changed. Let a = 1, and divide Equation 4.2.4 by $2\|\boldsymbol{w}\|$ yields,

$$m_D(\boldsymbol{w}) = \frac{1}{2}\hat{\boldsymbol{w}}^T(\boldsymbol{x}_+ - \boldsymbol{x}_-) = \frac{1}{\|\boldsymbol{w}\|}$$

Then finding the hyperplane with the maximum margin becomes the optimization problem:

$$\underset{w,b}{\text{maximize}} \frac{1}{\|\boldsymbol{w}\|}$$

$$\text{subject to} \quad y_i(\boldsymbol{w}^T\boldsymbol{x} + b) > 1$$

for $i = 1, \ldots, n$. Maximizing $\frac{1}{\|\boldsymbol{w}\|}$ is equivalent to minimizing $\|\boldsymbol{w}\|^2$, then the optimization problem is:

$$\underset{w,b}{\text{minimize}} \frac{1}{2}\|\boldsymbol{w}\|^2$$

$$\text{subject to} \quad y_i(\boldsymbol{w}^T\boldsymbol{x} + b) > 1$$

for $i = 1, \ldots, n$. (Carugo and Eisenhaber, 2009)

This SVM classifier works if the data is linearly separable, but it is not always the case in practice. Therefore we add a slack variable ξ to the constraint, yielding

$$y_i(\boldsymbol{w}^T\boldsymbol{x} + b) > 1 - \xi_i.$$

If $\xi_i > 1$, the example i is misclassified, so the sum of the slack variables bounds the number of misclassified examples. Therefore minimizing the sum of ξ minimizes the number of misclassified examples. The optimization problem becomes

$$\underset{w,b}{\text{minimize}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i$$

$$\text{subject to} \quad y_i(\boldsymbol{w}^T\boldsymbol{x} + b) > 1 - \xi_i$$

for $i = 1, \ldots, n$. This formulation is called the soft-margin SVM (Carugo and Eisenhaber, 2009).

In many applications a nonlinear classifier provides better accuracy. Support vector machines have the ability to embed the data into a higher-dimensional space, using the kernel trick (Russell and Norvig, 2010).

For example, using polynomial kernel of degree 2 on a 2-dimensional feature space maps the features x_1, x_2 into a 3-dimensional space $x_1^2, \sqrt{2}x_1x_2, x_2^2$. A hyperplane at $x_1^2 + x_2^2 = m^2$ defines a circle decision boundary at the origin in 2-dimensional space. Figure 4.2.1 illustrates this example (Courses.cs.ut.ee, 2015).

$$\Phi : R^2 \to R^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{(2)}x_1x_2, x_2^2)$$

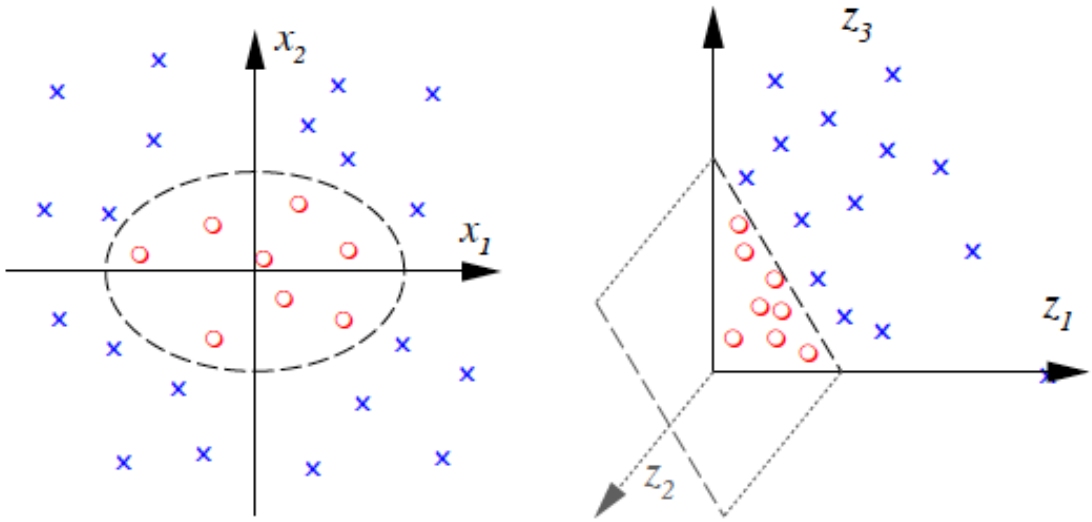

Figure 4.2.1: Illustration of a degree 2 polynomial kernel for SVM by Sven Laur (Courses.cs.ut.ee, 2015)

4.2.2 User Identification

Firstly, support vector classifier performed user identification experiment. This experiment was done in order to compare SVM to naive Bayes classifier on the same experiment.

The interface from scikit-learn library to the libsvm library is used to perform the classification. A 10-fold cross validation of support vector classifier trained on all clicking data and features, results in an accuracy of 16.28%, which is similar to the accuracy of 16.58% from naive Bayes classifier.

4.2.3 Anomaly Detection

Naive Bayes classifier has a better accuracy in the anomaly detection experiment than the user identification experiment, and support vector classifiers are binary classifiers. There-

Classifier	Correct	Wrong	Total	Percentage Correct
naive Bayes	268	1348	1616	16.58%
SVM	28	144	172	16.28%

Table 4.2.1: Detailed results of user identification experiments using naive Bayes classifier and support vector classifier

fore, we expected the SVM classifier's performance of the binary anomaly classification problem to be better than the result of the user identification experiment in Section 4.2.2.

Given the time constraint, only 1 training and testing iteration was conducted for every participant as the "owner", instead of 10 iterations as with naive Bayes classifier. Training with all features results in an accuracy of 62.50%. This result is better than the result using a naive Bayes classifier (54.83%) on the same problem.

Classifier	Correct	Wrong	Total	Percentage Correct
naive Bayes	1886	1554	3440	54.83%
SVM	215	129	344	62.50%

Table 4.2.2: Detailed results of anomaly detection experiments using naive Bayes classifier and support vector classifier trained with all features

In order to compare with the best result from naive Bayes classifier, a support vector classifier model was also trained with only Δt, pressure and contacting size. Its accuracy is 63.95%, which is similar to the accuracy of the support vector classifier trained with full features (62.50%), and less than the accuracy of Gaussian naive Bayes classifier in the same experiment (65.61%). Trimming of features involving coordinates did not give us a better result this time, because fewer features put the same example into a lower-dimensional space, and this makes them harder to separate by a hyperplane.

Classifier	Correct	Wrong	Total	Percentage Correct
naive Bayes	2257	1183	3440	65.61%
SVM	220	124	344	63.95%

Table 4.2.3: Detailed results of anomaly detection experiments using naive Bayes classifier and support vector classifier trained without features involving coordinates

4.2.4 SVC with Polynomial kernel

Another anomaly detection experiment was conducted using a support vector classifier with polynomial kernel of degree 2. The feature set only contains Δt, pressure and size. The accuracy was 67.67%, which is better than support vector classifier with linear kernel (63.95%), and the naive Bayes classifier.

Kernel	Correct	Wrong	Total	Percentage Correct
Linear	220	124	344	63.95%
Polynomial (degree 2)	233	111	344	67.67%

Table 4.2.4: Detailed results of anomaly detection experiments using support vector classifier with linear kernel and polynomial kernel of degree 2, trained without features involving coordinates

These results confirm the observations made in Section 4.1 on naive Bayes classifier. The results from using support vector classifier are similar to the results from using naive Bayes classifier. Other than speed, there is no advantage to use naive Bayes classifier over support vector classifier.

4.3 Task Specific Classification

In Section 4.1 and Section 4.2, information about the context is not considered, for example, a click on the button in the single button click task and a click on the number button on the dial pad task are treated equally. However, context information can be useful or even important to the touchscreen usage patterns of users. In this section, we tried to perform the anomaly detection experiment with the data from each task separately.

The three tasks that make up the mock application: single button clicks, dialing phone numbers and typing on a virtual keyboard, simulate three different environments where button clicks commonly appear. The activity is an important part of the context of clicking events, and it may have an impact to the usage pattern of the participants. We will not

discuss how each participant behaves differently during these three tasks. The goal of this experiment is only determining the identity of the user in a certain context.

We speculate that a task-specific model learned from these classifiers would be more powerful, because some features would have better distribution and be more distinguishable between users in a particular context than in general. The relative coordinates would benefit from this. Almost all buttons from one task have the same size, except the NEXT button in the dial pad task and the keyboard task. So the distribution of relative coordinates within a single task does not have the clusters in the overall distributions that result from the different shape of the buttons.

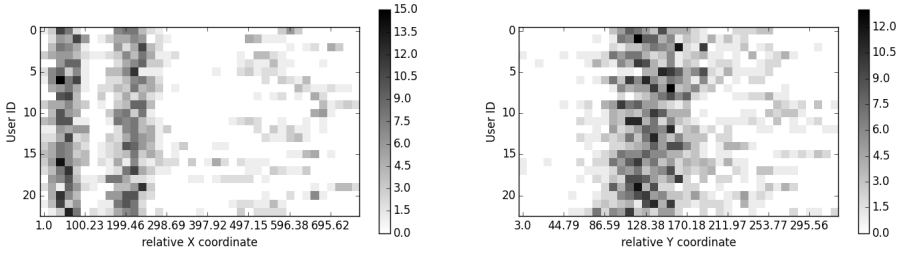

Figure 4.3.1: Individual distribution of relative coordinates in all tasks

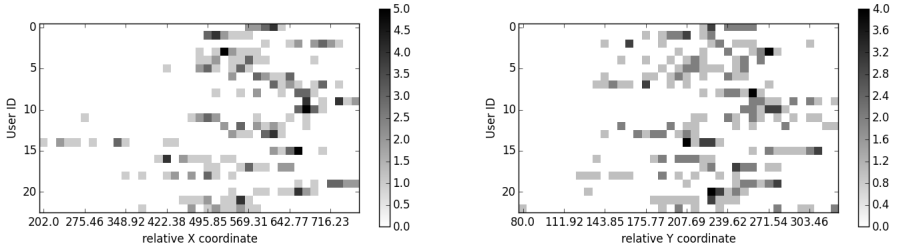

Figure 4.3.2: Individual distribution of relative coordinates in Task 1

The individual distributions of relative coordinates of all tasks and each task are plotted as described in Section 3.2.1. Two dense areas on the left can be seen in Figure 4.3.1, the distribution in all tasks. People usually press the center of buttons, so the relative coordinates largely depend on the shape of the button, rather than the usual practice of the

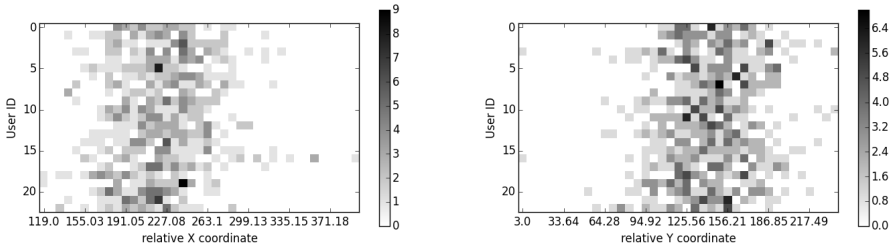

Figure 4.3.3: Individual distribution of relative coordinates in Task 2

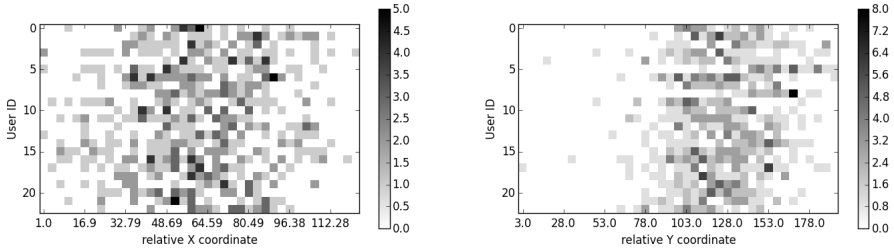

Figure 4.3.4: Individual distribution of relative coordinates in Task 3

user. However, within a single task, where button size is nearly constant, the distribution of each participant's data becomes more distinct. The most obvious example is Task 1, clicking on a single button, shown in Figure 4.3.2. The button is large enough so that the positions of each participant's clicks reflects their usual practice. For instance, User 14 clicked most on the left side of the button. This fact helped to distinguish this user from others.

The same anomaly detection classification described in Section 4.1.3 was used to examine if a fixed context actually helps the classification. Table 4.3.1 shows the result of this experiment. For single task trials, the training set and testing set are all from that particular task only. Also the clicks on the NEXT button to go to next tasks are excluded.

The results in Table 4.3.1 show that, for classification on the same clicking data set, the feature set of Δt, pressure, size and relative coordinates performs the best. This increase in accuracy indicates that relative coordinates are more informative for classification than the absolute coordinates.

Feature sets		Task 1	Task 2	Task3	All tasks
All Features	Correct	697	865	833	1951
	Wrong	223	575	647	1489
	Total	920	1440	1480	3440
	Accuracy	75.12%	60.07%	56.28%	56.72%
Without features involving coordinates	Correct	690	985	961	2290
	Wrong	230	455	519	1150
	Total	920	1440	1480	3340
	Accuracy	75.00%	68.40%	64.93%	66.57%
Δt, pressure, size and relative coordinates	Correct	790	992	971	2328
	Wrong	130	448	509	1112
	Total	920	1440	1480	3340
	Accuracy	85.87%	68.89%	65.61%	67.67%

Table 4.3.1: The anomaly detection classification using naive Bayes classifier with different feature sets, on each task and all tasks

The biggest improvement from relative coordinates is in the classifier on Task 1, and in general, the classifiers have the best performance on Task 1. This is surprising, as this task is considered the simplest and the least capable to capture information. The good accuracy of classifiers on Task 1 confirms our observation from Figure 4.3.2, which is that clicking on larger buttons discloses more information about the users clicking habits. Smaller buttons require more attention from the participants. The user's conscious control would weaken the expression of touching pattern, and the small area of the buttons might affect the ability to capture the touching pattern.

The same classification was also done using support vector classifiers. The results are shown in Table 4.3.2. These results confirm the observation from the same experiment using naive Bayes classifier. The feature set of Δt, pressure, size and relative coordinates performs the best, and Task 1 is the easiest to classify.

Unlike support vector classifiers in Section 4.2, we see that using naive Bayes classifier resulted in better accuracy in task specific anomaly detection experiment. This may indicates that the task specific data points are not clearly linear separable in the feature space, but the individual distributions are more distinguishable than the individual distribution

Feature sets		Task 1	Task 2	Task3
All Features	Correct	717	916	905
	Wrong	203	524	575
	Total	920	1440	1480
	Accuracy	77.93%	63.61%	61.15%
Without features involving coordinates	Correct	652	931	908
	Wrong	268	509	572
	Total	920	1440	1480
	Accuracy	70.87%	64.65%	61.35%
Δt, pressure, size and relative coordinates	Correct	717	977	957
	Wrong	203	463	523
	Total	920	1440	1480
	Accuracy	77.93%	67.85%	64.66%

Table 4.3.2: The anomaly detection classification using support vector classifier with different feature sets, on each task and all tasks

from data of all tasks. Therefore, naive Bayes classifier could take advantage of this, and improve its accuracy.

5
Conclusion

5.1 Conclusion

In user identification experiments, the classifiers in this project had a best accuracy of 17.88%. We did not do as well as Kolly et al. (2012) and Zaliva et al (2015). Both researchers took different approaches to represent the touches. Kolly built a touch profile for each user by using maximum, minimum and mean of a features for classification, while Zaliva considered each individual touch gesture as a single object (Zaliva et al., 2015). In this project, clicks were treated individually like in Zaliva et al.; however, Zliva also augmented the click data with proximity measurements.

Performing anomaly detection using Gaussian naive Bayes classifier had an accuracy of 65.61%, significantly lower than Kolly obtained. The due difference may beto the better combined use of click data. In Kolly's profile-based approach, multiple clicks are combined. Classifying single clicks, as in this study, is more difficult, because a single clicks may not represent the user's clicking pattern as accurately as multiple clicks do. A classification schema for a sequence of clicks would improve the classification accuracy.

Performing anomaly detection using support vector classifier had an accuracy about 63.95%, which is similar to Zaliva's result using SVM. However, it is worse than Harrison's result (97.8% at best) (Harrison et al., 2012). The reason could be the human impedance data they use. It may be more differentiable among people than the feature of clicks used in this project, and also having the special touchscreen, then could pick the a particular frequency to get the most differentiable data to perform the classification.

We found that context information are useful, as the accuracy improved in task-specific anomaly detection experiment. This is consistent with Feng's opinion, even though they intended to build a general passive authentication application (Feng et al., 2014).

When restricted to a specific task context, we found that the relative position of a click to the button is a useful feature, and larger button would reveal more information of the user's clicking patterns. Users react differently to the buttons with different size, and there would be less regularity in one person's clicks on a small button. The size of the button is not considered in Kolly's research (Kolly et al., 2012). In this study the best accuracy 85.87% in anomaly detection was achieved with naive Bayes classifier on Task 1, which has the largest button.

5.2 Future Work

This project would definitely benefit from comparing more classification techniques. More features extracted from the raw data, or learned representation of clicks, could improve the classification. More data could be collected from subjects, either by doing longer data collection work, or implementing a daemon to monitor touchscreen inputs, similar to Zaliva's paper (Zaliva et al., 2015). The data collected from the application could include gyroscope information to provide device holding gesture information.

More specific to classification, balancing the data for naive Bayes classifier, decreased the amount of training data. If the "owner" had more data than others, the classification might improve.

Classification methods based on sequences of clicks might provide additional cues. We could perform keystroke dynamic analysis on touchscreen, or a predictable model from clicking history could be built and used to classify the new clicks.

This project could be applied to build a input method, or phone dialer application which only the owner of the device could use. In practice, the touch pattern of a new "intruder" may not have been seen by the model. Strategies to deal with unseen touch patterns are of interest. The application may misclassify clicks, so a reinforcement learning schema could be developed to improve the classifier model.

Bibliography

Android (2015). Input events — android developers. `http://developer.android.com/guide/topics/ui/ui-events.html`.

Carugo, O. and Eisenhaber, F. (2009). *Data Mining Techniques for the Life Sciences*. Methods in Molecular Biology. Humana Press.

Courses.cs.ut.ee (2015). Andmekaeve graafides — main / kernelmethodsforgraphs browse. `http://courses.cs.ut.ee/2011/graphmining/Main/KernelMethodsForGraphs`.

Developer.android.com (2015). Motionevent — android developers. `http://developer.android.com/reference/android/view/MotionEvent.html`.

Docs.scipy.org (2015). scipy.stats.chisquare scipy v0.15.1 reference guide. `http://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.stats.chisquare.html`.

Feng, T., Yang, J., Yan, Z., Tapia, E., and Shi, W. (2014). Tips: Context-aware implicit user identification using touch screen in uncontrolled environments. *HotMobile '14 Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*.

Harrison, C., Sato, M., and Poupyrev, I. (2012). Capacitive fingerprinting: exploring user differentiation by sensing electrical properties of the human body. *UIST '12 Proceedings of the 25th annual ACM symposium on User interface software and technology*.

Hill, T., Lewicki, P., and Lewicki, P. (2006). *Statistics: Methods and Applications : a Comprehensive Reference for Science, Industry, and Data Mining*. StatSoft.

Kolly, S., Wattenhofer, R., and Welten, S. (2012). A personal touch: recognizing users based on touch screen behavior. *PhoneSense '12 Proceedings of the Third International Workshop on Sensing Applications on Mobile Phones*.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill international editions - computer science series. McGraw-Hill Education.

Oxforddictionaries.com (2015). The oec: Facts about the language - oxford dictionaries. http://www.oxforddictionaries.com/words/the-oec-facts-about-the-language.

Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall series in artificial intelligence. Prentice Hall.

Wikipedia (2015). Pearson's chi-squared test. http://en.wikipedia.org/w/index.php?title=Pearson\%27s_chi-squared_test&oldid=656618724.

Zaliva, V., Melicher, W., Saha, S., and Zhang, J. (2015). Passive user identification using sequential analysis of proximity information in touchscreen usage patterns. *2015 Eighth International Conference on Mobile Computing and Ubiquitous Networking (ICMU)*, pages 161–166.

Appendices

Appendix A

Consent Form of the Data Collection User Study

Research Consent Form
Senior project in Computer Science department, Bard College
Yu Qiu

DESCRIPTION: You are invited to participate in my senior project. The objective of the study is to record patterns of interaction with a mobile phone touchscreen. You will be asked to perform simple tasks, including typing on a virtual keyboard, scrolling through texts, clicking on buttons, drawing shapes and zooming in/out on maps, in order to provide a variety of touch data. Your touch data will be made anonymous and only be identified by a random ID number.

TIME INVOLVEMENT: Your participation will take less than 5 minutes.

RISKS AND BENEFITS: I do not anticipate any risks to you participating in this study other than those encountered in day-to-day life. We cannot and do not guarantee or promise that you will receive any benefits from this study.

You will NOT receive payment or compensation for your participation.

PARTICIPANT'S RIGHTS**:** If you have read this form and have decided to participate in this project, please understand **your participation is voluntary** and you have the **right to withdraw your consent or discontinue participation at any time without penalty or loss of benefits to which you are otherwise entitled**. **The alternative is not to participate.** You have the right to refuse to answer particular questions. The results of this research study may be presented in my senior project, presented at scientific or professional meetings or published in scientific journals. Your data are only associated with a random id number that allows us to distinguish the data of each individual. Your identifying information (e.g., your name) will not be recorded or stored and therefore cannot and will not be revealed by any written materials arising from this study.

CONTACT INFORMATION:
Questions: If you have any questions, concerns or complaints about this research, its procedures, risks and benefits, contact me, through email, yq0283@bard.edu or text/call 845-706-6883, or contact my advisor, Sven Anderson, through email, sanderso@bard.edu or call 845-752-2322.

Independent Contact: If you are not satisfied with how this study is being conducted, or if you have any concerns, complaints, or general questions about the research or your rights as a participant, please contact the Bard Institutional Review Board (IRB) through e-mail: irb@bard.edu

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

By signing this form, I give consent for my touch data to be obtained during this study:

SIGNATURE _____ DATE _____

Appendix B

Bard IRB Decision Letter

Bard College Institutional Review Board

Date: March 25, 2015
To: Yu Qiu
Cc: Megan Karcher, Sven Anderson
From: Pavlina R. Tcherneva, IRB Chair
Re: IRB proposal 2015

DECISION: APPROVED

Dear Yu Qiu,

The Bard Institutional Review Board reviewed the revisions to your proposal. Your proposal is approved through May 25, 2015.

Please notify the IRB if your methodology changes or unexpected events arise.

We wish you the best of luck with your research.

Pavlina R. Tcherneva
tchernev@bard.edu
IRB Chair

PO Box 5000, Annandale-on-Hudson, New York 12504-5000 Phone 845-758-6822