A SMARTPHONE-BASED POINTING TECHNIQUE IN CROSS-DEVICE INTERACTION

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Juechen Yang

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Program:
Software Engineering

April 2019

Fargo, North Dakota

North Dakota State University
Graduate School

Title

A Smartphone-Based Pointing Technique In Cross-Device Interaction

By

Juechen Yang

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Jun Kong
Chair

Dr. Zhili(Jerry) Gao

Dr. Changhui Yan

Approved:

July 31, 2019
Date

Dr. Kendall Nygard
Department Chair

ABSTRACT

In this paper, the author develops a novel cross-device cursor position estimation system for transferring a mobile device's four-direction 2D movement to a cursor's four-direction movement on a large display device; this is achieved through the use of sound-source localization and machine-learning algorithms. This system is implemented in two steps. First, the system starts the cursor's position initialization by taking advantage of the theory of sound-source localization. Second, the system transfers the mobile device's movement to the cursor's movement by means of a machine-learning model. This newly developed system improves usability of cross-device applications by offering intuitive 2D move gesture and multi-user interaction context and removes physical distance restrictions. A pilot test has been conducted, and the results have demonstrated that naïve Bayes and gradient boosting are suitable for detecting the 2D movement of a mobile device.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

The use of mobile devices, especially smartphones and tablet computers, is becoming popular in the U.S. in recent years. According to a technology report from the Pew Research Center, smartphone ownership rates remained high at 77% in 2018, and from 2016 to 2018, the influx rate of tablet computers increased from 12% to 20% (http://www.pewinternet.org/fact-sheet/mobile/). These facts indicate that mobile devices will continue to be a major player in the field of information technology in the 21st century.

Public display devices have been widely deployed in diverse environments. These devices, in most cases, serve as broadcast platforms that raise no interest from bypassers (Sarabadani Tafreshi, Soro, and Tröster 2018). First, the content in most public displays is static and uninteresting, and there is no user-triggered interaction. Viewers can only passively receive information from the screen. For example, a crowded shopping mall contains many large electronic screens that advertise various brands. However, bystanders rarely care about these advertisements because they do not desire the products. Second, a single public display device only allows one person to operate it at a time; this causes substantial waiting times. The use of these public displays also raises health concerns due to the high-frequency use of contaminated input hardware, such as the touchscreen and the game handler. To summarize, direct methods of interaction present numerous issues that cross-device interactions could solve.

Many studies have been engaged in discovering a method and style that can be applied to cross-device interactions. One impressive study has created a “drag-and-drop” style of interaction that allows the user to transfer data objects between two touchable devices (Ikematsu and Siio 2015). Another interesting application called Gradual Engagement can automatically detect transferrable data objects between devices that are physically close to each other

(Marquardt et al. 2012). To implement the data transfer, this application requires the user to drag the detected data object to the display zone on the destination screen.

The two aforementioned applications are only viable when all the screen regions of the large display device are physically accessible. However, current trends have indicated that the size of large display devices will continue to increase due to the decreasing price of hardware and the demand for experience and usability. Paay et al. have noted a crucial point for all interaction methods: a larger target display (termed a "large display device" in this study) leads to more efficient and effective interactions (Paay et al. 2017). This observation raises an important consideration for cross-device interactions; if the size of a large display device is extremely big that not all the regions are physically accessible, then it could be arduous to build a seamless connection between a mobile device and a large display device. This theory has not yet been thoroughly studied.

Considering the issues that have been expressed in prior studies, this project is motivated to produce a low-cost, intuitive, and physically unconstrained cursor position estimation system in order to improve the experience of a single user and to enable the availability of multi-user participation. The attribute of "low-cost" implies, on the user side, that the system only uses built-in sensors to capture a device's movements. The attribute "intuitive" is reflected by the "sliding move" gesture, which can be grasped by users without any learning or reasoning process. The attribute "physically unconstrained" means that the experience of controlling the cursor remotely is offered. Thus, the chance for multiple users to interact with the large display simultaneously is provided.

The purpose of this study centers on two objectives. First, this project uses sound localization to facilitate cursor initialization and to enable coarse cursor position estimation.

Second, the project aims to enable precise cursor position estimation through the detection of the four-direction mobile device's movement on a 2D surface by means of machine learning classification. Although, in terms of direction, the current position estimation system can only distinguish the direction of a device's movement based on four directions, the estimation system is still a core research section that contributes to the precise mapping system because, on a 2D flat surface, an object's motion in any direction can be considered as a vector that can be decomposed, in turn, into two vectors that are perpendicular to each other. The four-direction movement, to some extent, reflect the two vectors and, thus, should be treated as the core research objective for mapping a mobile device's movement to a cursor's movement.

The project is composed of two steps. First, there is a cursor initialization app that can allow the user to initiate the cursor's position on a large display device through sound localization, thus improving the adaptability of the system when interacting with extremely large devices. Second, the device's movement translation system transfers the 2D mobile device's four-direction movement to the cursor's four-direction movement by machine-learning models. The project also encompasses a data-analysis pipeline for characterizing data into statistical features (mean, standard deviation, min-max difference, and power energy) and spectral features (dominant frequency and spectral energy) and provides a comprehensive study of different machine-learning (ML) algorithms and feature selection sets. The conclusions explain what features and what ML algorithms should be used to classify the four-direction movement and the stand statuses(no movement) during the time interval used for 1 pixel movement of a typical computer mouse, and increase the seamless experience of the interaction.

In terms of cursor initialization, the project uses sound localization to determine the 2D coordinates of the cursor starting position on the large display screen. At the same time, a

resolution conversion has been applied so as to ensure the cursor position estimation area on the large screen is physically equal to the size of the mobile device. In the data-analysis section, both 10-folds cross-validation and a confusion matrix have been implemented, and multiple feature-selection methods have been conducted in order to find the most relevant features that contribute to the machine-learning model.

The results of this study reveal that three classifiers, in particular, gradient boosting, linear discriminant analysis (LDA), and naïve Bayes, have demonstrated a high performance through the use of not only all features but also multiple feature sets that are generated by feature-selection methods. Feature-selection tests indicate that features that combine speed and mean or speed and median can contribute the most to the recognition rate. However, performance of classification can be boosted by using features that include all vectors (acceleration, angular-velocity, and speed) but that limit feature categories only to mean or median.

RELATED WORK

Three Domains Of Interaction-Sensing Techniques

Prior studies have proposed numerous notable findings with regards to cursor position estimation and selection in cross-device applications. To summarize all the related studies, there are principally three domains to consider. First, the use of direct touch on a large display device has been applied to a large number of applications. Strohmeier (Strohmeier 2015) has introduced an interaction framework that uses the mobile device as the operational commander to initiate designated operations and to implement them through direct finger touch. For example, users can pick a color on their personal devices and can then draw a shape on the large target display using a finger motion.

Schmidt et al. have provided a novel solution (Schmidt et al. 2012) that combines the physical touch initiated from a mobile device with its orientation to indicate the target interaction region and to manipulate various operations. The restriction of this framework is that it does not allow the remote control of the target region and, thus, creates barriers to multiple users interacting with the large display simultaneously. Another project called SleeDCursor (von Zadow et al. 2014) is a target-region-selection application that uses a touch-based system to provide users with increased flexibility in that they can initiate the binding of a device through close-coupling (where one selects the closest device to interact with). However, through the aforementioned interaction techniques, users are still forced to maintain physical proximity to the large public screen in order to exchange information. Consequently, if multiple users initiate data transfers from the public screen simultaneously, they can still commonly interfere with one another and, thus, produce a negative experience. All these direct-touch applications have the same drawback—the strict requirement that the user must have physical access to the screen of

the target large display device. However, in this project, the renovated cursor position estimation system can offer users a remote controlling experience that significantly improves the flexibility of the usage.

The second domain uses additional pointing devices, such as laser pointers, to help the server identify the position of the mobile device. The developer of PointerPhone (Seifert, Bayer, and Rukzio 2013) has systematically built multiple applications that use laser pointers and cameras on the server's system to precisely detect the laser-point motion and to heighten a user's ability of controlling the large display screen remotely. Another hybrid technique with a gesture-assisted, head-based coarse pointing style has been introduced in this work(Nancel et al. 2013). This technique has created predefined gesture combinations in order to trigger the pointing task, and the technique used an equipped headset to perform a precise position estimation of the point thereafter. For example, a user could initiate a tap gesture on the touchpad surface followed by a drag operation so as to activate the pointing task and enable any area of the large display to be reached with absolute precision. Nonetheless, this approach contains some shortcomings as well. The framework requires certain additional devices, resulting in a high-cost setup for the user. Second, gesture-initiated pointing tasks increase the complexity of manipulation. Under certain circumstances, the user may have a higher chance of triggering an undesired operation, and this results in a negative experience and poor usability. This project distinguishes its methods from the applications mentioned above by leveraging built-in sensors, rather than additional devices, to enable position estimation from the device's movement to the cursor's movement; this adjustment reduces the usage complexity of cross-device interaction applications.

The third domain utilizes all available built-in sensors, such as cameras, accelerometers, and gyroscopes, in order to sense the mobile device's movement. This study (Boring, Jurmu, and

Butz 2009) has proposed three interaction styles to mimic the movement of the device and to map it to the large display pointer. Gestures such a “tilting” and “scrolling” were created to evaluate the motion by means of a built-in accelerometer that calculated the value of acceleration continuously. The author’s project has chosen the gesture of the “sliding move,” which is regarded as more intuitive than “tilting” or “scrolling” since this movement style performs similarly to the cursor action (such as moving up and down or left and right) and could be easier for users to understand and learn. Furthermore, the implementation has been renovated by means of collecting data from motion sensors (accelerometer and gyroscope) instead of a camera.

In terms of pairing devices, many techniques have been experimented with. Rekimoto (Rekimoto 2004) has built the “SyncTap” and has constructed a collaborative pairing style for cross-device interactions that allows multiple users to pair devices with a single tap on the touchscreen. Peng et al. created “Point&Connect” (Peng et al. 2009) which has a technique for combining devices by leveraging the built-in microphone and acoustic signals. Yuan et al. have proposed using a cross-device tracking framework (Yuan et al. 2018) to identify “same” devices in terms of user typing actions and then building secure cross-device communication.

This paper has proposed a pairing style with an extended ability not only to pair the devices by means of a web socket as the connection channel but also to initialize the cursor’s estimated location on the large display by the application of sound localization through a microphone array. This design innovatively explores a new interaction medium that can use the movement of a mobile device on a flat surface, such as a desk, so as to move the cursor in four directions on the large display, which is comparable to a computer mouse. Under this style, interactions can be both easy and intuitive.

Sensor-Based Daily Activities Detection

Several studies have sought to improve the utilization and analysis of data generated from accelerometers by sensing the daily activities of people. A study by Chen, Yang, and Jaw (Chen, Yang, and Jaw 2016) has introduced the use of accelerometers for detecting a person's fall. Their study has also detailed a basic workflow for parsing and filtering the data retrieved from the accelerometer. Their project has introduced and investigated features such as sum vector, rotation angle, and slope to detect falls with a degree of both specificity and sensitivity. Furthermore, their study has noted a critical decision-making strategy: it is no longer sufficient to determine results based on the generated data by simply proposing different thresholds in making predictions. Instead, machine-learning models and algorithms should be applied to extract patterns from the observed data and to help solve complex problems.

Another fall-detection study performed by Rakhman et al. (2014) has tried to detect fall-down activity through the magnitude of both the accelerometer and gyroscope and through the rotation angle of the mobile device. They have proposed an in-house algorithm to calculate all the features needed and to discover the thresholds on values for fall-down determination. Moreover, they have categorized fall activity into four subcategories, such as "fall forward" and "fall backward", in order to measure the accuracy rate.

A gait-sensing study by Ferrero et al. (2015) has comprehensively investigated how to sense human gaits based on the data collected from an accelerometer. Their study has introduced some crucial data-preprocessing steps, including linear interpolation, data normalization, and noise filtration. Because of the earth's gravitational force, it is ideal to incorporate all three dimensions' acceleration data in an analysis. However, if a mobile device can be placed

perpendicular to the ground throughout the sensing session, then the model should be adjusted to assume that only one dimension is affected by gravity.

The major difference between the author's project and other sensor-based, activity-detection projects concerns the "time window." The time window used in sensing daily activities is normally 1 or 2 seconds. However, this project is pursuing an extremely sensitive system that uses a 0.015 second time window in order to detect a device's movement. This requirement has imposed challenging tasks, such as preprocessing the raw data and tagging the classification samples.

RESEARCH OVERVIEW

Motivating Examples

Generally, for cross-device applications, it is unavoidable that one selects the object and interacts with it through many approaches. A number of studies have introduced their unique methods of selecting an object. In order to improve the usability of existing studies and applications, this project has selected three attributes: low-cost, intuitive, and physically unconstrained in finding new solutions.

Low-Cost

A low-cost cursor position estimation application indicates that the amount of software or hardware involved in a system should be minimized. This study has followed the objective of being low-cost by only using built-in sensors for detecting the four-direction movement of the mobile device rather than applying additional devices (Nancel et al. 2013)(Seifert et al. 2013). Thus, this design improves user usability and experience.

Intuitive

Instead of using gestures such as tilting (Boring et al. 2009), this project has developed an intuitive gesture that can be termed as “sliding move” for controlling the cursor’s movement via the mobile device. This gesture is intuitive since the cursor’s movement is highly consistent with the sliding gesture such that the learning time of a user can be significantly reduced. Furthermore, for controlling the cursor on the display screen, users are used to sliding a computer mouse on a 2D flat surface and controlling the position of the cursor.

Physically Unconstrained

Physically unconstrained typically means that the implementation of interaction methods should always avoid the requirement that the user be physically close to the large display device.

It is recommended that the interaction application has a remote-control mechanism that offers freedom to the user. In that context, the user can participate in cross-device interaction if the large display is visible to the user. At the same time, multi-user participation becomes viable since the user does not need to stand in front of the large display and to interfere with others. Thus, this paper has proposed a new cursor position estimation system that can provide remote-control function, rather than a touch-based control mechanism (Strohmeier 2015) (von Zadow et al. 2014); this innovation allows the user to manipulate objects on the large display devices from his or her mobile device.

Workflow

The general working flow of this cursor position estimation application can be demonstrated in two flow charts (Figure 1 and 2), which refer to the cursor position initialization and the cursor's movement estimation steps. The first workflow can be depicted in a series of sub-steps. First, the user launches the application from the client-side and requests a connection to the webserver, which is the public display that runs the application. Then the server responds to the mobile device with a successful connection message. Second, the user requests cursor initialization from the client-side, and the server starts a listening session to identify touch-down sound. Third, the user produces a touch-down sound during the listening session, and then the server determines whether it is a qualified sound input for triggering the initialization of the cursor. If the sound is qualified, the server computes the location coordinates (X, Y) of sound via sound localization, and it sends the location object to the mobile device. Subsequently, the mobile device can use the location coordinates (X, Y) to display an area of contents on the server's screen using (X, Y) as the upper left vertex and use the physical length and width of the

mobile device as the size. Fourth, the user can move the mobile device on the 2D flat surface to any location on the server's screen and interact with its contents on the mobile device's screen.

The second workflow starts from the data collection in which the user performs ten "sliding move" movements such that the mobile device can record the raw data generated from the built-in sensors. Five of them are left-right-based movements; this means that each "sliding move" contains the following gestures: 2 seconds of "stand," 3 seconds of "move right," and 2 seconds of "move left." Another five of them are up-down-based movements; this means that each "sliding move" contains the following gestures: 2 seconds of "stand," 3 seconds of "move up," and 2 seconds of "move down." Subsequently, the mobile device fits this data into an algorithm and builds a machine-learning model to detect pixel movements. Moreover, a series of continuous pixel movements are used to construct a cursor's movement. Finally, the cursor's movement is implemented on the screen of large display devices with the distance equal to the number of pixel movements inside the cursor's movement.

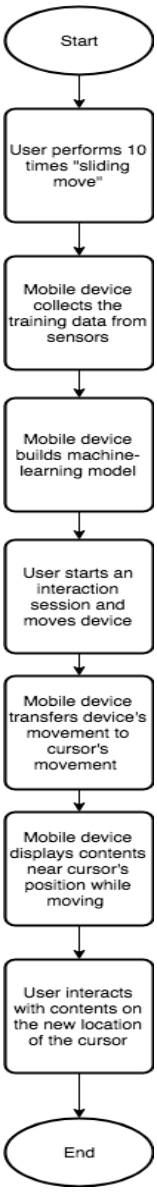

Figure 1. Flow chart of cursor's movement estimation

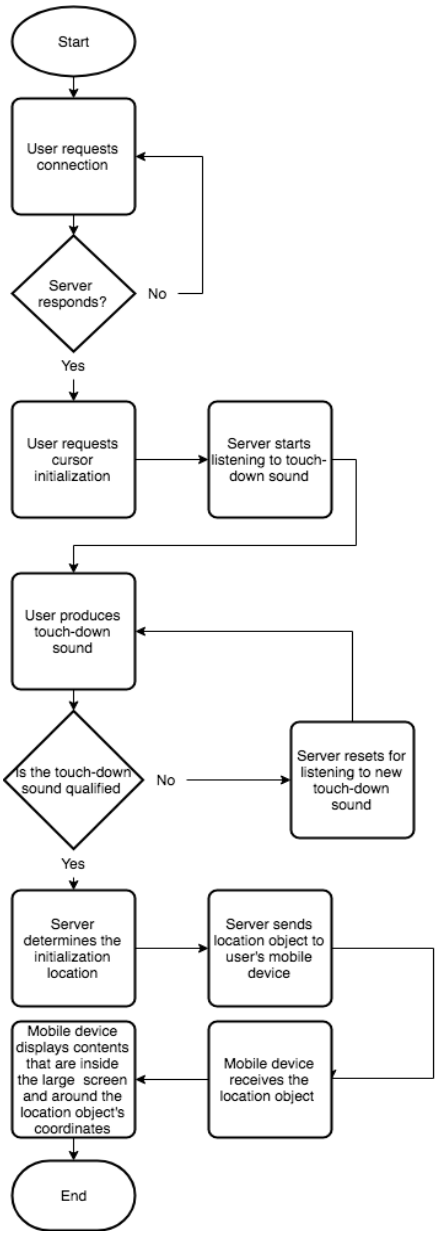

Figure 2. Flow chart of cursor initialization

CURSOR INITIALIZATION DESIGN

Basically, cursor initialization benefits the user when he or she is involved in cross-device interactions, where the target large display screen is extremely large. The user can perform a coarse position-estimation to find out an area that he or she is interested in, and then use cursor movement simulation to precisely locate the designated object displayed on the large screen.

Overview

The general workflow in cursor initialization can be represented in the following steps. First, the user briefly examines the content on the large display screen and coarsely selects an area where he or she is willing to interact. Second, the user estimates the touch-down area on the flat surface based on the positioning layout of the selected area, the center point of the screen's top bar, and the position of the microphone array. Third, the user places the mobile device onto the touch-down area with a sound generated from the collision between the mobile device and flat surface. Fourth, the large display device computes the angle direction relative to the microphone array and then provides the coordinates of the user-interested area. Finally, the mobile device displays that desired area for the user to interact within.

Estimate Interaction Area

A user can estimate the interaction area that he or she is interested in through coarse position-estimation. First, a user must define a rectangle working area (on a flat surface) whose physical width and height are both M times the width and height of the large display screen. The value of M should be set between 0 and 1 since this adjustment will downsize the working area and will render it more accessible than the large display screen. Second, a microphone array is set on the top center of the working area (see Figure 3A). Third, when the user finalizes a desired

area A on the large display (see Figure 3B), he or she should estimate the relative position A' (see Figure 3A) on the working area based on the center of the microphone array. One should assume that the physical distance between the center of area A and the top center of the large display is "cld " then the estimated distance "cld' "on the working area is equal to cld * M. The position angle α on the large display device is equal to α' on the working area. Finally, the user can make a touch-down sound on the area of A' in order to initialize the cursor on the area where he or she is interested.

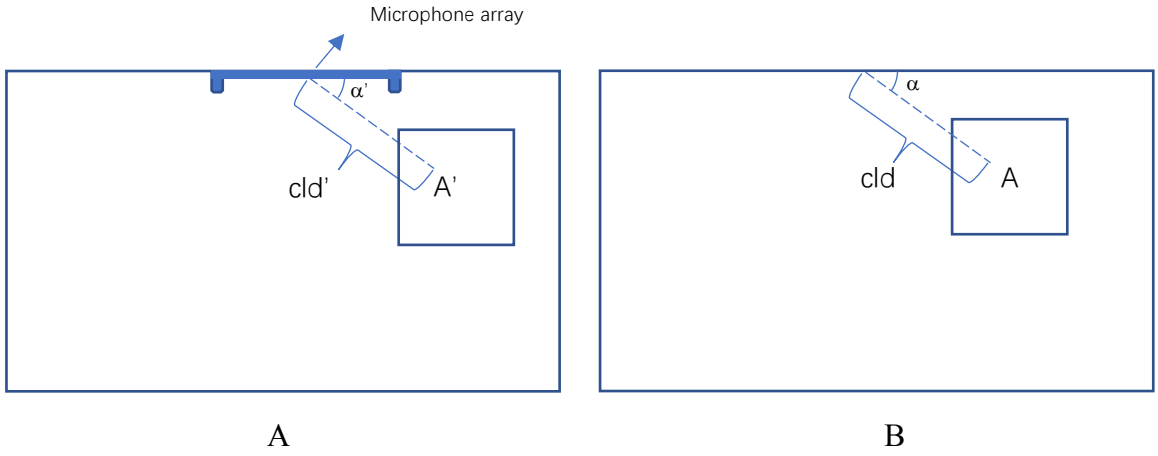

Figure 3. Determine initialization area (A and B)

Estimate Initial Position Coordinates

The cursor initialization is completed by taking advantage of a mobile device, a flat surface, and a large display device so as to produce a touch-down sound that can be captured by a microphone array equipped with the large display device. This touch-down sound is then analyzed as the cue for computing the coordinates that represent the location of the sound source. The angle of the sound source aligned to the center of the microphone array can be obtained

through the calculation of the time-delay difference between each channel of the microphone array (see Figure 4).

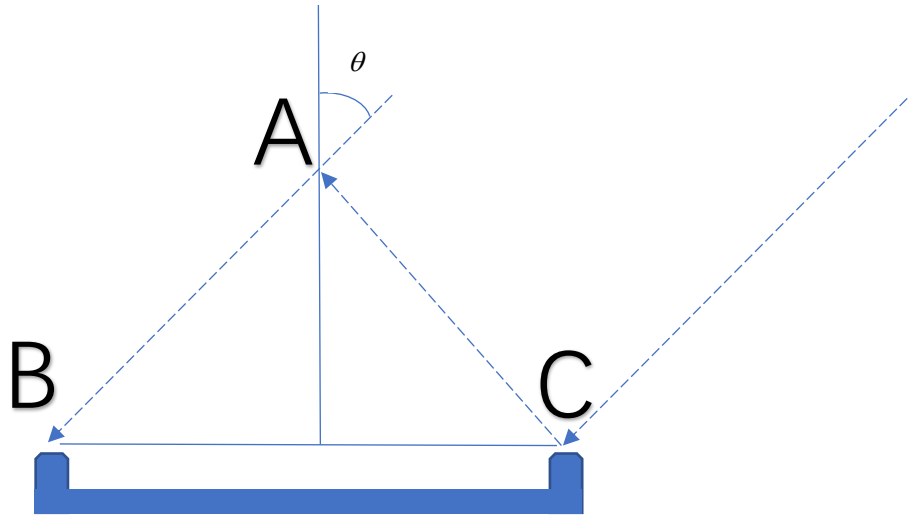

Figure 4. Sound localization schema

As Figure 4 shows, the distance between A and B is evaluated by the production of a time delay difference (TDD) and the speed of the sound. The distance between A and B can be a negative value if the sound source is located on the other side of the microphone array's central point, and this causes the TDD to be negative. A dual-delay algorithm was implemented with "binaural sound source localization" (http://www.laurentcalmes.lu/soundloc_software.html) as a reference. The distance between B and C is captured by the measurement of the distance between the two channels of the microphone array. Thus, the angle of the sound source θ can be calculated via Formula 1:

$$\theta = \arcsin\frac{AB}{BC} \quad (1)$$

Once the angle has been calculated, the sound intensity can be evaluated through the computation of the decibel level of the touch-down sound. Since environmental noise could easily interfere with sound localization, a threshold was proposed to determine whether the

sampled sound is a real touch-down sound that is qualified to trigger the initialization. If the intensity of the sampled sound is larger than the threshold, the mobile device is registered as a client of the target device. Sound intensity also helps to estimate the relative source sound (x, y), which is implemented in Formula 2:

$$x = sin\,\theta \cdot ds$$
$$y = \cos\theta \cdot ds \quad (2)$$

In the above formula, ds is an estimated measurement of the distance between the source touch-down sound and the microphone array using sound intensity. The coordinates are used as the relative pixel-based start-point on the target device screen, which takes the center of its top boundary as the original base point. The actual start-point (X, Y) is estimated based on Formula 3:

$$X = \frac{HR}{2} + x$$
$$Y = 0 + y \quad (3)$$

Where HR represents the horizontal resolution of the large display device. However, since θ can vary from -90° to 90° and the value ds is not restricted by the size of the large display screen, it is possible that (X, Y) could be out of the range of the large display screen. If either X or Y is out of the range, then the application should automatically adjust the value to its closest boundary in order to avoid a positioning error. For example, if the target device has a resolution of 1024px * 768px and if the source sound coordinates are (500, 866), then the actual location start point is calculated as follows: (1,024/2 + 500)px, (0 + 866)px. In this case, the system adjusts the value of Y from 866px to 768px since the vertical positioning is higher than the maximum boundary (866px > 768px).

The estimated area is strictly equivalent to the physical size of the mobile device. A key attribute called “dots per inch” (DPI) was used to obtain the relative pixel-based width (RPW) and relative pixel-based height (RPH) for the mobile device on the large display by means of Formula 4:

$$RPW = \frac{mx}{mdpi} \cdot ldpi$$
$$RPH = \frac{my}{mdpi} \cdot ldpi \quad (4)$$

In the above formula, mx and my are the horizontal and vertical resolutions of the mobile device, and mdpi and ldpi are the DPIs for the mobile device and large display, respectively.

Finally, the server sends a screenshot image and its screen DPI to the client-side mobile device. When the image is received by the mobile device, it only displays the partial area of that screenshot that uses (X, Y) as the base point and has the width of RPW and height of RPH. These designs can deliver the experience of having a virtual screen on a flat surface. If, moreover, the user touches down on the coordinates relative to the center point of the microphone array, it is possible to begin the interaction at exactly the same coordinates relative to the top center of the large display.

DETECTING CURSOR MOVEMENTS

Overview

The cursor movement simulation indicates the core process of position estimation. In other words, this section is trying to transfer the mobile device's four-direction movement to the cursor's four-direction movement. There are two critical parameters for simulating the device's movement to the cursor's movement. The first involves the movement status detection, which means detecting whether the device is moving or not in real time; the second parameter entails the direction.

According to a study by Bohan, Thompson, and Samuelson (2003), a typical cursor movement takes 1.002 seconds to finish and travels 18.75 mm, which is 66 px on the monitor that the aforementioned study used (a 19-inch monitor with a resolution of 1400 * 1050). However, one should not simply determine each cursor movement based on the data collected every 1.002 seconds because the cursor will jump from one location to another rather than moving continuously.

Therefore, a typical cursor movement simulation should be broken into pixel-movement detection for classification. Pixel-movement detection can be described as follows: (1) Collecting sensors' raw data during a time interval (0.015s), which is the time used by the cursor to travel 1 pixel (2) Extracting features of the raw data to compose as one sample. (3) Six classes are predefined (referred to as "stand_on_x," "move_right," "move_left," "stand_on_y," "move_up," and "move_down") in order to indicate different statuses. (4) This sample is classified into one of the six predefined groups. Thus, through the implementation of this process, both movement status and direction detection can be addressed.

Pixel-Movement Experiment Design

Facility Setup

The application in this study was developed on an Android platform using API level 16. The actual test run was performed on a Samsung Galaxy S4, which is an Android mobile device. A cross-device large display server was set up on a traditional desktop device embedded with a Windows operating system. Any Java-supported desktop device using any operating system could also serve as the target device. Moreover, an Andrea SoundMAX Superbeam Array Microphone was deployed in the system to enable sound localization.

Design Details

In order to sample the data for all six predefined class, the pixel-movement classification experiment was designed to have a 7-second data collection session applied on x and y axes separately that included three separate actions: 2 seconds of "stand," 3 seconds of "move_right/up," and 2 seconds of "move left/down" to the axis. Tables 1 and 2 list the detailed actions in the data collection session. The raw data from the accelerometer and gyroscope were sampled at a rate of 590 hertz since, on average, 4,131 raw data instances were fetched from the experimental phone in 7 seconds. In addition, the time spent for a 1 px movement was 0.015s; since in 1.002 seconds, the cursor can travel 66 pixels.

Moreover, a visual text interface was provided to notify the user of the designated action to perform at a given timestamp. Normally, the visual reaction for a human is 0.25 seconds (https://backyardbrains.com/experiments/reactiontime). This application notified the user to perform an action 0.25 seconds before the actual recording time. This adjustment was designed to ensure that the user-recognized timestamp was strictly in accordance with the machine's recording time.

Table 1. Action design

Actions	Timestamp
Stand	0 to 2 second
Move right	2 to 5 second
Move left	5 to 7 second
Stand	0 to 2 second
Move up	2 to 5 second
Move down	5 to 7 second

Table 2. Labeling design

Labels	Timestamp
0(stand_on_x)	0 to 1 second
1(move_right)	3 to 4 second
2(move_left)	6 to 7 second
3(stand_on_y)	0 to 1 second
4(move_up)	3 to 4 second
5(move_down)	6 to 7 second

The time frame as shown in Table 1 and Table 2 was designed due to an important issue encountered during the data collection. This issue raised a question of how to label the benchmark class correctly. This problem is closely related to the motive for designing this experiment. Each test may contain thousands of raw instances, and each device has its own mechanical delays because of the variance in the buildup of CPU (central processing unit) speeds. For example, on the threshold of performing actions in Second 2 or Second 5 (indicated by a red circle in Figure 5), the designated class could be mislabeled even when the visual delay

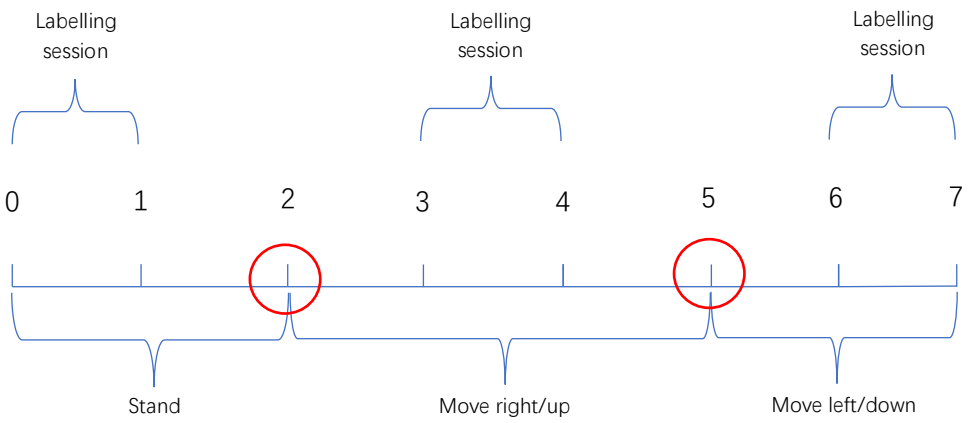

Figure 5. Graph demo of experiment design

time is added prior to the actual action timestamp. For example, presumably, the program labels the raw instances collected after Second 2 as "move right." However, when the user receives the visual notice and starts to move the device, the timer may have already reached Second 2.2. Thus, the raw instances fetched between Second 2 and Section 2.2 are mislabeled. The reason for this design is twofold. First, sufficient samples for labeling require at least 1 second, which is the typical cursor movement duration (use 1 to replace 1.002 for easy calculation), to be assigned to the labeling session. Additionally, at least 1 second between the labeling session and the action change threshold is also necessary in order to accommodate potential mislabels. Second, it is important to shorten the data collection session as much as possible because this data collection process is required when the user is trying to control the cursor by means of the device's movements. It is a usability requirement that the application should maintain a low complexity and should avoid overtaxing the user's patience. If an application has many complex preprocessing steps that consume a substantial amount of time before actual usage, users tend to lose interest in the application. This design allowed the experiment simultaneously to maximize the duration of effective sampling data and to minimize the complexity of using the application. Thus, the total time is measured in formula 5:

$$Min(tls + tlmp + tlmn + tds + 2tdmp + tdmn) \quad (5)$$

In the above equation, tls, tlmp, and tlmn are the times used for labeling "stand," "move right/up," and "move left/down," while tds, tdmp, and tdmn are required sessions to prevent mislabeling. Since both the labeling session and the mislabel preventing session require 1 second as the minimum time interval, the total required time for data collection is 7s, based on the above formula. Furthermore, since the action "move right/up" is located in the middle, it required two sessions to prevent mislabeling (see Figure 5).

The experiment was performed ten times. Five of the tests were left-right-movement based, and five of them were up-down-movement based. Ten raw datasets were produced by leveraging the aforementioned strategy. These datasets were then used in downstream analysis in order to determine the best models and features for predicting the mobile device's movement on the 2D flat surface.

PREPROCESSING AND FEATURE EXTRACTION

Data Preprocessing

There were ten raw datasets generated from the ten tests, and the preprocessing method was applied to each raw dataset. The preprocessing of the raw instances was composed of two steps. First, only instances with a predefined class label were kept to construct the filtered raw dataset. According to the experiment design, more than half of the raw instances could be potentially labeled incorrectly (1 s to 3 s and 4 s to 6 s in Figure. 5); thus, these raw instances were dropped in order to avoid mislabeling. Second, the filtered raw dataset was divided into sub-datasets to facilitate statistical calculations and feature extraction. Nine raw instances were assigned to each sub-dataset sequentially on the time frame so as to test different machine-learning algorithms with the goal of making the application extremely sensitive to the device's movement detection. Nine raw instances were used as the grouping metrics because a pixel movement typically finished in 0.015 seconds, and there were, on average, 4,131 instances collected in 7 seconds from each raw dataset. Therefore, to accurately classify each pixel movement, nine raw instances were required. Subsequently, each group of nine raw instances was transformed into a sample that represented a pixel movement, with features calculated by feature-extraction methods. The label values (0, 1, 2, 3, 4, 5) typically represent the device's statuses ("stand_on_x," "move_right," "move_left," "stand_on_y," "move_up," and "move_down"). The label that occurred most frequently in the group were assigned as the label for the corresponding sample. Finally, all the generated samples from a single raw dataset constructed a preprocessed dataset, and ten preprocessed datasets, marked from "data 0" to "data 9," were produced with, on average, 196 samples in each.

Feature Extraction

A total of 63 features were extracted from the raw data via mathematical or statistical computation. For the raw data, tri-axial accelerometer and gyroscope data, indicated as (ax, ay, az) and (gx, gy, gz), were collected to compute these features. Furthermore, tri-axial speed values (vx, vy, vz) were captured through the use of acceleration and timestamp at each raw instance. Formula 6 depicts the detailed computation method.

$$V_i = \begin{cases} a_i \cdot t_i & where\ i = 0 \\ V_{i-1} + a_i \cdot t_i & where\ i > 0 \end{cases} \tag{6}$$

In general, feature categories can be classified into two domains, represented as the time domain and frequency domain (Table 3). All the 63 extracted features are listed in Table 4.

Table 3. Extracted features on domains

Domains	**Feature categories**
Time domain	Mean
	Standard deviation
	Minimum-maximum difference
	Median
	Energy
Frequency domain	Dominant frequency
	Spectral energy

Table 4. All 63 features

Vectors	Axes	Mean	Standard deviation	Minimum-maximum difference	Median	Energy	Dominant frequency	Spectral energy
Acceleration	X	mean*ax	std*ax	min_max_gap*ax	median*ax	energy*ax	main_freq*ax	spectral_energy*ax
	Y	mean*ay	std*ay	min_max_gap*ay	median*ay	energy*ay	main_freq*ay	spectral_energy*ay
	Z	mean*az	std*az	min_max_gap*az	median*az	energy*az	main_freq*az	spectral_energy*az
Rotation	X	mean*gx	std*gx	min_max_gap*gx	median*gx	energy*gx	main_freq*gx	spectral_energy*gx
	Y	mean*gy	std*gy	min_max_gap*gy	median*gy	energy*gy	main_freq*gy	spectral_energy*gy
	Z	mean*gz	std*gz	min_max_gap*gz	median*gz	energy*gz	main_freq*gz	spectral_energy*gz
Speed	X	mean*vx	std*vx	min_max_gap*vx	median*vx	energy*vx	main_freq*vx	spectral_energy*vx
	Y	mean*vy	std*vy	min_max_gap*vy	median*vy	energy*vy	main_freq*vy	spectral_energy*vy
	Z	mean*vz	std*vz	min_max_gap*vz	median*vz	energy*vz	main_freq*vz	spectral_energy*vz

The frequency-domain features were captured through Fast Fourier Transform (FFT), which transfers a signal from a time-domain to a frequency-domain (Welch 1967). Formula 7 provides the equation:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N}kn} \tag{7}$$

In this above formula, x_n indicates sensors' readings on time domain, and N indicates the length of this signal. The real number part of the computed X_k indicates the amplitude

spectrum of each frequency domain. The dominant frequency is captured through finding a frequency value that has the maximum amplitude.

The energy of each axis was computed by adding up the square numbers of the raw instances in a signal. The spectral energy was calculated using the same method but with the raw value transformed from a time-domain to a frequency-domain by FFT. Formula 8 (Parseval's theorem) and 9 (cf. Stein and Jonathan Y. 2000) describe the detailed calculation:

$$\text{Enery(y)} = \sum_{n=1}^{N} x_n^2 \tag{8}$$

$$Spectral(\text{y}) = \sum_{n=1}^{N} (FFT(x_n))^2 \tag{9}$$

where x_n indicates sensors' readings on time domain, and N indicates the length of this signal.

The Python programming language, with its powered libraries such as Pandas, Scikit-learn, and NumPy, was used in both the preprocessing and feature extraction. These libraries and tools contain powerful built-in functions that can capture statistical features with a high speed and accuracy.

CLASSIFICATION AND RESULTS

Several studies have been conducted to determine which classification algorithm is the most accurate candidate for extracting patterns from built-in sensors' data. Algorithms such as the support vector machine (SVM) (Zhang et al. 2006), k-nearest neighbors (Preece et al. 2009), and naïve Bayes (Bao and Intille 2004) have been used to extract data patterns in order to verify whether daily activities can be detected. In terms of multi-class classification tasks, the study by Lee and Kwan (2018) has suggested that random forest and gradient boosting are the most favored candidates for personal-activity classification. To better fit the data model and classification task in this study, all the classifiers previously mentioned, along with additional candidates, were included in the pool to ascertain whether there were any novel findings.

Basic Cross-Validation

Cross-validation is an evaluation tool that examines whether a model is an effective predictor for data that is completely new and differs from the existing dataset. The simplest way to avoid this "overfit" issue is what is known as a "holdout method." This method typically splits the dataset into two groups: one group is used for training, and the other group is used for testing. The amount of training and testing is generally assigned at a ratio of 7:3. However, there is an evident weakness that can produce a high variance in the model. The result of each test classification may rely on the endpoint of the training or testing set. Therefore, the strategy of splitting the dataset becomes a critical factor that can affect the evaluation results.

10-Folds Cross-Validation

This investigation applied a 10-folds cross-validation to avoid the aforementioned bias. 10-folds cross-validation is a specific case of a general method known as "K-folds cross-validation." Through the use of this K-folds cross-validation, the dataset could be split into K

subsets, and on each subset the holdout method was performed once. Each iteration only used one of the K subsets as the test group and the other K-1 subsets as the training group. This method notably improved the holdout since it mitigated the impact of the data-division strategy. As the K value increased, the variance in the evaluation results declined. Moreover, K = 10 was used because the number of test times in this study equals 10. Thus, to keep the consistency, the author chose 10 as the value for parameter K.

A 10-folds cross-validation was applied on a dataset that contained 1,969 samples. This dataset was combined by appending ten preprocessed datasets together. Ten classifiers were evaluated by the accuracy performance (Figure 6 and Table 5); these classifiers included AdaBoost (adaptive boosting), decision tree, gradient boosting, LDA (linear discriminant analysis), linear SVM (support vector machine), naïve Bayes, nearest neighbors, neural network, random forest, and RBF (radial basis function) SVM.

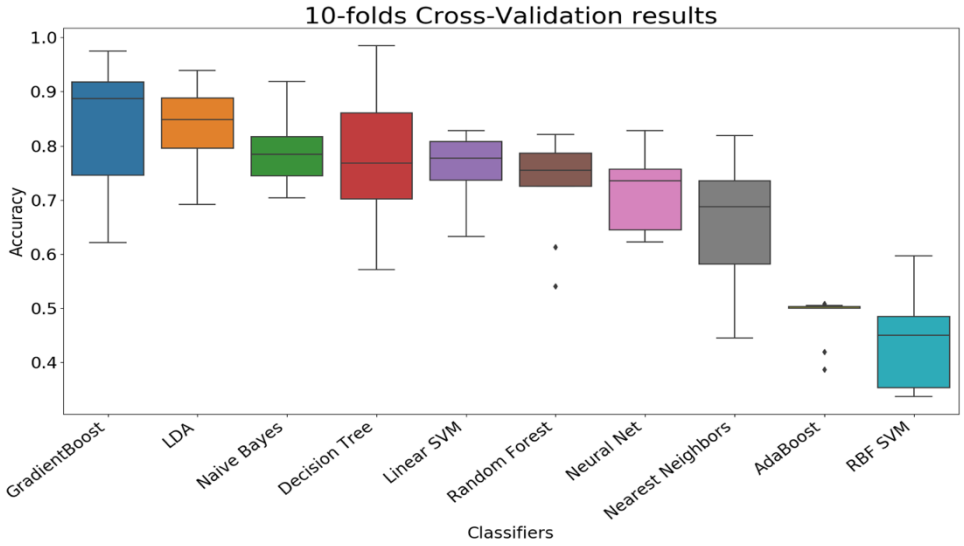

Figure 6. Cross-validation results for all features (Boxplot)

In Figure 6, from left to right, the performance of all classifiers is sorted by descending order of mean and by ascending order of standard deviation. From the observations, gradient boosting, LDA, and naïve Bayes are the top three classifiers, which have reached the average

accuracy of 83.65%, 83.43%, and 79.42%; their standard deviation is measured at 11.18%, 7.43%, and 6.68%, respectively. Based on the average accuracy, gradient boosting should be selected as the agent for this classification problem; however, gradient boosting has a higher standard deviation (11.18% versus 7.43%) but a similar average accuracy (83.65% versus 83.43%) compared with LDA. This comparison suggests that the performance of gradient boosting may significantly vary when different training and testing data are used for classification problems. Thus, LDA is recommended when all features are included.

Table 5. Cross-validation results for all features

Classifiers	**Accuracy mean**	**Accuracy std**
Gradient boosting	83.65%	11.18%
LDA	83.43%	7.43%
naïve Bayes	79.42%	6.68%
Decision tree	76.91%	11.66%
Linear SVM	76.24%	5.60%
Random forest	72.70%	6.22%
Neural net	72.47%	7.06%
Nearest neighbors	66.51%	11.05%
AdaBoost	48.24%	4.03%
RBF SVM	44.27%	8.66%

In machine learning, besides the accuracy rate, the manner in which the results of the error predictions are distributed is also critical for downstream analysis. The distribution of errors can be determined through an examination of the confusion matrix of the classification results proposed by each classifier. In this project, the confusion matrices of the top three classifiers (gradient boosting, LDA, naïve Bayes) were demonstrated (Table 6).

Table 6. Confusion matrices for top three classifiers

Gradient boosting						
stand_on_x	move_right	move_left	stand_on_y	move_up	move_down	**Classified as**
208	1	0	116	0	0	**stand_on_x**
0	277	14	0	38	0	**move_right**
0	1	311	0	5	19	**move_left**
70	1	0	261	0	0	**stand_on_y**
0	26	3	0	289	0	**move_up**
0	0	22	0	0	307	**move_down**
LDA						
stand_on_x	move_right	move_left	stand_on_y	move_up	move_down	**Classified as**
218	0	0	107	0	0	**stand_on_x**
0	307	0	0	22	0	**move_right**
0	6	307	0	18	5	**move_left**
127	1	0	204	0	0	**stand_on_y**
0	29	9	1	279	0	**move_up**
0	0	0	0	1	328	**move_down**
naïve Bayes						
stand_on_x	move_right	move_left	stand_on_y	move_up	move_down	**Classified as**
206	2	2	115	0	0	**stand_on_x**
0	242	23	0	64	0	**move_right**
0	16	252	0	52	16	**move_left**
73	3	0	255	1	0	**stand_on_y**
0	3	19	0	296	0	**move_up**
0	2	14	0	0	313	**move_down**

The confusion matrix generally shows the distribution of correct and error predictions. Each row label indicates the predicted class, and the column label indicates the actual class. For example, based on the confusion matrix of naïve Bayes, the first observation of 206 falls under the column of "stand_on_x" and the row of "stand_on_x" as well. This result means that 206

samples are predicted as “stand_on_x” and also belong to the “stand_on_x” class, and this means, ultimately, that these 206 samples have been predicted correctly. However, the second horizontal observation 2 indicates that two samples are predicted as “stand_on_x” but actually belong to the “move_right” class; this means that these samples have been predicted incorrectly.

Some notable findings can be ascertained from the confusion matrix. First, it seems that all these three classifiers have managed an exemplary performance in distinguishing “move” and “stand” regardless of direction since there are few observations in the cells of “stand_on_x/y” that are classified as “move_right/left/up/down.” Second, commonly, samples are classified into incorrect labels where the difference only lies in an axis (x or y) compared with the original label. For example, there are many error predictions that fail to distinguish whether the sample is in the “stand_on_x” group or the “stand_on_y” group.

FEATURE SELECTION

In machine learning, feature selection plays an important role that can substantially impact not only the learning accuracy of the prediction model but also the efficiency and user experience of the application. Feature selection represents the process of fetching a subset that contains the most relevant features from an original feature set based on statistical algorithms and has been proven to be accurate through both theoretical and practical success in multiple application scenarios (Liu et al. 2018) (Ali and Aittokallio 2019). To determine what features should be selected, multiple methods were applied in this project, and these methods can be grouped into two categories: algorithm-based methods and manual feature selection methods.

Algorithm-Based Methods

Three algorithm-based methods were tested in this project: linear correlation analysis, select k best, and recursive feature elimination (RFE). All were evaluated by applying 10-folds cross-validation on the feature sets proposed.

Linear Correlation Analysis

Linear correlation is a statistical method to investigate the strength of association between two features in order to obtain the most relevant features and to remove irrelevant features through an examination of the strength between each feature and the labeled class. Moreover, a Pearson correlation coefficient was computed for each pair of features by means of Formula 10:

$$r = \frac{\sum_i (x_i - \bar{x})((y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2}\sqrt{\sum_i (y_i - \bar{y})^2}} \quad (10)$$

In this formula, x_i and y_i represent the values of two features, and $\bar{x}$ and $\bar{y}$ are the mean values of each feature. The result is always a decimal number between -1 and 1. If this

number is close to 1, then the two variables X and Y reveal a high positive correlation. If this number is close to -1, then the two variables reveal a high negative correlation. A threshold of 0.5 (https://www.dummies.com/education/math/statistics/how-to-interpret-a-correlation-coefficient-r/) was proposed so as to keep features that have absolute values of their correlations with a label class larger or equal to 0.5. A feature set containing four features was generated: “mean*vz,” “median*vz,” “energy*vz,” and “spectral_energy*vz”.

Select K Best

A select K best method uses a specific function to score each feature and to select the highest K scoring features. This project computed an analysis of variance (ANOVA) F-value between the label and each feature, and it used K = 10 to perform this task because the author aimed to investigate the performance of each classifier when the number of features increased compared with a linear correlation analysis. Selected features were as follows: “median*vz," “mean*vz," “spectral_energy*vz,” “energy*vz,” “spectral_energy*vy,” “energy*vy,” “spectral_energy*vx,” “energy*vx,” “median*vy,” and “mean*vy”.

Recursive Feature Elimination

Recursive feature elimination (RFE) is a method that proposes certain candidate features by gradually focusing on a smaller set of features. Usually, it starts with a trained estimator to assign an importance value to each feature, and then the feature with the lowest importance value is eliminated from the candidate pool. This process is continued recursively until the desired number of features has been satisfied.

This study employed a linear SVM as a trained estimator since it has a high accuracy and an efficient generalization ability for removing features recursively (Yan and Zhang 2015).

With this algorithm fitted into the dataset, the RFE model proposed an optimized number of 31 features.

Algorithm-Based Results

Cross-validation results for the top three classifiers in each method have been demonstrated in Table 7. From the observation of proposed features, there are 33 unique features selected by all three methods, and 84.85% (28 out of 33) are time-domain-related; this potentially suggests that frequency-domain features are not as important as time-domain features for this classification task. In addition, LDA, naïve Bayes, and gradient boosting are potential candidates to be recognized as the best classifier for detecting device movement since they are ranked as the top three classifiers in at least two feature-selection methods. Furthermore, 39.4% (13 out of 33) of the proposed features are speed-related, and both the select K best and the linear correlation selected features that are all speed-related; this may indicate that speed is a critical vector in terms of determining device movement.

Table 7. Cross-validation for algorithm-based feature selection

Feature selection method	Selected features	classifiers	Accuracy _mean	Accuracy _std
Linear correlation	“mean*vz”, “median*vz”, “energy*vz”, “spectral_energy*vz”	naïve Bayes	73.20%	12.07%
		RBF SVM	72.57%	11.50%
		Neural Nnet	72.13%	13.20%
Select k best (Anova)	“median*vz", “mean*vz", “spectral_energy*vz”, “energy*vz”, “spectral_energy*vy”, “energy*vy”, “spectral_energy*vx”, “energy*vx”, “median*vy”, “mean*vy”	LDA	78.75%	8.13%
		Random forest	76.26%	9.59%
		Gradient boosting	76.12%	10.81%
Recursive feature elimination	“mean*ax”, “mean*ay”, “mean*az”, “mean*vx”, “mean*vy”, “mean*vz”, “mean*gx”, “mean*gz”, “min_max_gap*ax”, “min_max_gap*ay”, “min_max_gap*az”, “min_max_gap*vy”, “min_max_gap*gy”, “min_max_gap*gz”, “median*ax”, “median*ay”, “median*az”, “median*vx”, “median*vy”, “median*vz”, “median*gz”, “main_freq*gy”, “main_freq*gz”, “energy*ax”, “energy*az”, “energy*vx”, “energy*vz”, “energy*gy”, “energy*gz”, “spectral_energy*vy”, “spectral_energy*vz”	Gradient boosting	84.07%	10.74%
		LDA	82.82%	8.14%
		naïve Bayes	80.40%	4.26%

Manual Feature Selection

Manual feature selection is another method for pursuing a small set of features by removing irrelevant features. However, instead of using algorithms to determine feature importance, Manual feature selection proposes feature sets based on assumptions that may explain what features are related to the success of movement detection. For example, in physics, speed typically describes where and how fast the object is moving. Angular velocity detects whether the object has any rotation event. Thus, an assumption can be made as when the

machine-learning model tries to detect the four-direction movement of a mobile device, it's better using speed-related features rather than angular-velocity-related features to build the model. Because the key of determining four-direction movement is to detect where and how fast the object is moving. Movements on a flat surface barely produce rotation event.

Another hypothesis is that speed-related features should exclude standard deviation and minimum-maximum difference in order to have a better performance. Because the standard deviation and minimum-maximum difference can only measure the extent of speed alteration rather than indicate where and how fast the object is moving. The ideal feature category should be mean or median because they reflect the raw values of speed in a pixel-movement time interval.

The core process of manual feature selection is composed of five steps: (1) Evaluating the classification performance of feature sets that each of them relates to either one vector (acceleration, angular-velocity, and speed) or one feature category (mean, std, min_max_gap, median, energy, dominant frequency, and spectral energy). Since each vector or feature category has its unique physical or statistical meanings, it's better to understand how they uniquely impact the success of the classification. (2) Finding out the vector VH that its related features can produce a higher recognition rate than other vectors. (3) Finding out the feature category FCH that its related features can produce a higher recognition rate than other feature categories. (4) Selecting a combined feature set that contains features relates to VH and FCH simultaneously. (5) Evaluating the performance of this combined feature set to verify if the recognition rate can be higher than former feature sets. The objective of this section is to examine whether using features that relate simultaneously to the most relevant vector and most relevant feature category can reach a higher performance

Vector-Based Feature Selection

The difference between vectors acts as an imperative factor for the success of the classification task because different vectors represent their own unique physical significance. Therefore, it is critical to evaluate features related to a single vector or a single feature category separately. An analysis for selecting features based on different vectors is demonstrated in Table 8.

Table 8. Performance of vector-based feature selection

Vector selection	Selected features	Number of features	Classifiers	Accuracy_mean	Accuracy_std
Acceleration	"mean*ax", "mean*ay","mean*az","std*ax", "std*ay","std*az", "std*az","min_max_gap*ax", "min_max_gap*ay","min_max_gap*az","median*ax", "median*ay","median*az","energy*ax", "energy*ay","energy*az", main_freq*ax", "main_freq*ay","main_freq*az","spectral_energy*ax", "spectral_energy*ay","spectral_energy*az"	21	Gradient boosting	63.87%	5.27%
			Random forest	61.07%	5.26%
			Decision tree	57.12%	4.28%
Angular velocity	"mean*gx", "mean*gy","mean*gz","std*gx", "std*gy","std*gz", "std*gz","min_max_gap*gx", "min_max_gap*gy","min_max_gap*gz","median*gx", "median*gy","median*gz","energy*gx", "energy*gy","energy*gz", "main_freq*gx", "main_freq*gy","main_freq*gz","spectral_energy*gx", "spectral_energy*gy","spectral_energy*gz"	21	Decision tree	51.96%	5.31%
			Gradient boosting	51.96%	3.97%
			Random forest	50.67%	7.45%
Speed	"mean*vx", "mean*vy","mean*vz","std*vx", "std*vy","std*vz", "std*vz","min_max_gap*vx", "min_max_gap*vy","min_max_gap*vz","median*vx", "median*vy","median*vz","energy*vx", "energy*vy","energy*vz", "main_freq*vx", "main_freq*vy","main_freq*vz","spectral_energy*vx", "spectral_energy*vy","spectral_energy*vz"	21	LDA	80.03%	9.84%
			Random forest	79.98%	8.92%
			Gradient boosting	79.06%	6.92%
Acceleration and angular-velocity	"mean*gx", "mean*gy","mean*gz","std*gx", "std*gy","std*gz", "std*gz","min_max_gap*gx", "min_max_gap*gy","min_max_gap*gz","median*gx", "median*gy","median*gz","energy*gx", "energy*gy","energy*gz", "main_freq*gx", "main_freq*gy","main_freq*gz","spectral_energy*gx", "spectral_energy*gy","spectral_energy*gz","mean*gx", "mean*gy","mean*gz","std*gx", "std*gy","std*gz", "std*gz","min_max_gap*gx", "min_max_gap*gy","min_max_gap*gz","median*gx", "median*gy","median*gz","energy*gx", "energy*gy","energy*gz", "main_freq*gx", "main_freq*gy","main_freq*gz","spectral_energy*gx", "spectral_energy*gy","spectral_energy*gz"	42	Gradient boosting	65.08%	5.33%
			Random forest	62.20%	3.51%
			Decision Tree	59.59%	5.95%
Speed and acceleration	"mean*ax", "mean*ay","mean*az","std*ax", "std*ay","std*az", "std*az","min_max_gap*ax", "min_max_gap*ay","min_max_gap*az","median*ax", "median*ay","median*az","energy*ax", "energy*ay","energy*az", "main_freq*ax", "main_freq*ay","main_freq*az","spectral_energy*ax", "spectral_energy*ay","spectral_energy*az","mean*vx", "mean*vy","mean*vz","std*vx", "std*vy","std*vz", "std*vz","min_max_gap*vx", "min_max_gap*vy","min_max_gap*vz","median*vx", "median*vy","median*vz","energy*vx", "energy*vy","energy*vz", "main_freq*vx", "main_freq*vy","main_freq*vz","spectral_energy*vx", "spectral_energy*vy","spectral_energy*vz"	42	Gradient boosting	83.25%	10.90%
			naïve Bayes	83.01%	7.36%
			LDA	81.71%	7.05%
Speed and angular-velocity	"mean*vx", "mean*vy","mean*vz","std*vx", "std*vy","std*vz", "std*vz","min_max_gap*vx", "min_max_gap*vy","min_max_gap*vz","median*vx", "median*vy","median*vz","energy*vx", "energy*vy","energy*vz", "main_freq*vx", "main_freq*vy","main_freq*vz","spectral_energy*vx", "spectral_energy*vy","spectral_energy*vz","mean*gx", "mean*gy","mean*gz","std*gx", "std*gy","std*gz", "std*gz","min_max_gap*gx", "min_max_gap*gy","min_max_gap*gz","median*gx", "median*gy","median*gz","energy*gx", "energy*gy","energy*gz", "main_freq*gx", "main_freq*gy","main_freq*gz","spectral_energy*gx", "spectral_energy*gy","spectral_energy*gz"	42	LDA	82.31%	8.94%
			Gradient boosting	81.75%	7.76%
			Linear SVM	75.71%	6.55%

From the results of vector-based feature selection, it is obvious that as long as speed-related features are included, then the average accuracy of classification is always significantly higher (80.47% > 58.16%) than the feature set without speed-related features. At the same time,

acceleration- or angular-velocity-related features can be considered as artifacts for the classification task: first, because features based on these two vectors are performing at an extremely low accuracy rate (on average, 58.16%); second, because there is no significant accuracy improvement when either acceleration- or angular-velocity-related features are mixed with speed-related features (for example, LDA sits at 80.03% with speed-only features, but at 81.71% and 82.31% when mixing speed with acceleration- and angular-velocity features, respectively).

Feature-Category-Based Feature Selection

The feature categories presented in the feature extraction section were inspired by previous studies (Preece et al. 2009) (Fang, Yishui, and Wei 2016). However, some of the feature categories are either mathematical- or statistical-confounding factors for the project's classification problem. Thus, it is wise to analyze them separately in order to observe what feature categories contribute positively to the machine-learning model.

This study applied cross-validation on seven feature categories independently, and the results, including the top three classifiers that have the highest accuracy rates, are shown in Table 9. From the observations, it seems that through the application of median-related features, the accuracy rate can boom to 85.36% when naïve Bayes is used as the classifier. However, there is another observation that also uses naïve Bayes as the classifier but applies mean-related features; this reaches a similar recognition rate of 85.25%. These findings suggest that either median- or mean-related features contribute to the learning process. In addition, the accuracy rate of main-frequency-related ("main_freq") features are significantly lower than the accuracy of other feature categories; this could indicate that major frequency-related features are irrelevant to the success of the project's classification task.

Table 9. Performance of feature-category based feature selection

Feature-category	Selected features	Number of features	Classifiers	Accuracy_mean	Accuracy_std
Mean	"mean*ax","mean*ay","mean*az","mean*gx","mean*gy","mean*gz","mean*vx","mean*vy","mean*vz"	9	naïve Bayes	85.25%	6.53%
			Gradient boosting	84.00%	12.42%
			LDA	77.98%	11.57%
Std	"std*ax","std*ay","std*az","std*gx","std*gy","std*gz","std*vx","std*vy","std*vz"	9	Gradient boosting	56.51%	4.01%
			Decision tree	48.54%	6.84%
			Random forest	46.51%	6.40%
Min_max_gap	"min_max_gap*ax","min_max_gap*ay","min_max_gap*az","min_max_gap*gx","min_max_gap*gy","min_max_gap*gz","min_max_gap*vx","min_max_gap*vy","min_max_gap*vz"	9	Gradient boosting	58.05%	4.26%
			Decision tree	50.29%	5.56%
			Random forest	50.27%	4.88%
Median	"median*ax","median*ay","median*az","median*gx","median*gy","median*gz","median*vx","median*vy","median*vz"	9	naïve Bayes	85.36%	7.27%
			Gradient boosting	82.54%	10.69%
			Random forest	78.55%	7.82%
Energy	"energy*ax","energy*ay","energy*az","energy*gx","energy*gy","energy*gz","energy*vx","energy*vy","energy*vz"	9	Gradient boosting	83.00%	11.06%
			naïve Bayes	78.50%	6.23%
			Decision tree	77.93%	11.29%
Main_freq	"main_freq*ax","main_freq*ay","main_freq*az","main_freq*gx","main_freq*gy","main_freq*gz","main_freq*vx","main_freq*vy","main_freq*vz"	9	LDA	25.95%	1.60%
			Linear SVM	25.85%	0.72%
			Decision tree	25.49%	1.06%
Spectral_energy	"spectral_energy*ax","spectral_energy*ay","spectral_energy*az","spectral_energy*gx","spectral_energy*gy","spectral_energy*gz","spectral_energy*vx","spectral_energy*vy","spectral_energy*vz"	9	Gradient boosting	83.10%	11.42%
			Decision tree	78.42%	11.84%
			naïve Bayes	78.29%	7.19%

Combined Analysis

From the brief analysis of the previous two feature-selection methods, another hypothesis can be posited that if the feature set was downsized into a state where only speed and mean- or median-related features were included, then the accuracy of the specific classifier should be higher or remain at the same level. To verify this assumption, this study has created two feature

sets: one containing only the speed and mean-related feature, and the other only the speed and median-related feature. Cross-validation was then applied on these two feature sets. After generating the results, the author selected "gradient boosting," "LDA," and "naïve Bayes" as the benchmark classifiers for comparing since these are the most recurrent classifiers considered to be high performing, based on the previous results.

Table 10. Feature selection performance with benchmark classifiers

Description	Feature set	Number of features	Gradient boosting	LDA	naïve Bayes
Speed and mean	"mean*vx","mean*vy","mean*vz"	3	79.10%	72.34%	74.38%
All vectors and mean	mean*vx,"mean*vy","mean*vz","mean*ax","mean*ay","mean*az","mean*gx","mean*gy","mean*gz"	9	84.00%	77.98%	85.25%
Speed and median	"median*vx","median*vy","median*vz"	3	78.32%	72.34%	74.38%
All vectors and median	"median*vx","median*vy","median*vz","median*ax","median*ay","median*az","median*gx","median*gy","median*gz"	9	82.54%	77.88%	85.36%
Speed and all feature categories	mean*vx, "mean*vy","mean*vz","std*vx", "std*vy","std*vz","min_max_gap*vx", "min_max_gap*vy","min_max_gap*vz","median*vx", "median*vy","median*vz","energy*vx", "energy*vy","energy*vz", "main_freq*vx", "main_freq*vy","main_freq*vz","spectral_energy*vx", "spectral_energy*vy","spectral_energy*vz"	21	79.06%	80.03%	71.43%
All 63 features	All 63 features	63	83.65%	83.43%	79.42%

The compared results (Table 10) provide strong evidence for declining the aforementioned hypothesis. With a focus on features that are only related to speed and mean, the

performance of the classification actually decreased for all three benchmark classifiers, compared to the performance of the feature set with all vectors (speed, acceleration and angular-velocity). The same trend occurred on another feature set (speed and median related feature) since the recognition rates of benchmark classifiers boomed when features related to all vector and median were involved in the learning model.

Moreover, these results also provide evidence against the assumption addressed in 8.2.1, that acceleration- and angular-velocity-related features cannot contribute to the performance of the classification task because it is evident that the addition of more vector-based features can lead to a boom in performance (gradient boosting: from 79.10% to 84.00%; LDA: from 72.34% to 77.98%; naïve Bayes: from 74.38% to 85.25%).

Furthermore, this performance boom was not driven from the addition of the number of features: Table 10 shows that even with 21 features or all 63 features selected, gradient boosting and naïve Bayes performed worse than previous feature sets (all vectors and mean, all vectors and median) that only have 9 features.

CONCLUSION AND DISCUSSION

In this paper, a novel approach leveraging machine learning for estimating cursor position has been proposed. The new method can allow the user to remotely initialize and control the position of the cursor with the benefits of a low-cost, intuitive, and physically unconstrained experience. Through sound localization, the user can perform a coarse pointing to the desired interaction area by using a touch-down sound on the flat surface. In terms of cursor's movements control, this project proposed a four-direction mobile device's movement detection system for allowing the user to control the cursor by moving the mobile device. In total, 63 features and 10 classifiers were employed to construct the machine-learning models, and multiple feature-selection methods have been applied to find an optimized machine-learning model.

Four major conclusions should be addressed. First, this study proposed naïve Bayes, gradient boosting, and LDA as the reliable classifiers to build machine-learning models. Regardless of whether there is any feature-selection method involved, these three agents always have a higher recognition rate than others.

Second, from the result of the confusion matrix, it appears that there is almost no error prediction between "stand" classes (stand_on_x, stand_on_y) and "move" classes (move_right, move_left, move_up and move_down); this implies that the proposed machine-learning model performs well in distinguishing "move" from "stand". Most of the error predictions centered on classes inside "stand" and "move". For example, with all features selected, gradient boosting incorrectly classified 70 "stand_on _x" samples into "stand_on_y" samples and 116 "stand_on_y" samples into "stand_on_x" samples; these incorrect predictions capture 58.86% (186 out of 316) of the total number of incorrect predictions. Moreover, there were more error predictions in classifying two groups: "move_right" or "move_up" and "move_left" or

“move_down)”. This suggests that it is more difficult to differentiate “move right or move _up” and “move left or move down” than other combinations.

Third, only recursive feature elimination in the algorithm-based feature-selection method can reach a similar recognition rate compared to the non-feature-selection model. It reduces half of the dimensions by only using 31 features, but it is also a challenge to explain why these 31 selected features contributed to the learning results.

Fourth, by applying manual-selected features, the findings demonstrate that in vectors (acceleration, angular-velocity, and speed), speed-related features are more relevant to the classification task than the other two vectors. Furthermore, in feature categories, mean- and median-related features contribute more to the learning process than other feature categories. However, by limiting the features to speed and mean-related features (mean*vx, mean*vy, and mean*vz), the performance drops significantly, based on the three benchmark classifiers (see Figure 10). It seems that to have a high performance, the learning model should add back some features. Based on the observation in Figure 10, gradient boosting and naïve Bayes have a prominent recognition rate increase when other vector-related features are added back. However, when all feature-category-related features are added back (third row in Figure 10), there is no performance boom. These findings suggest that the presence of all vector-related features is more important than the presence of all feature-category-related features in terms of performance, and, moreover, it is wise to only use speed-related features to perform the classification task since other vectors also contribute to the learning process.

REFERENCES

Ali, Mehreen and Tero Aittokallio. 2019. “Machine Learning and Feature Selection for Drug Response Prediction in Precision Oncology Applications.” *Biophysical Reviews* 11(1):31–39.

Bao, Ling and Stephen S. Intille. 2004. “Activity Recognition from User-Annotated Acceleration Data.” Pp. 1–17 in *Pervasive Computing*. Vol. 3001, edited by A. Ferscha and F. Mattern. Berlin, Heidelberg: Springer Berlin Heidelberg.

Backyard Brains. “Experiment: How Fast Your Brain Reacts To Stimuli.” Retrieved January 24, 2019a (https://backyardbrains.com/experiments/reactiontime)

Bluma, Avrim L. 1997. “Selection of Relevant Features and Examples in Machine Learning.” *Artificial Intelligence* 97:245–271.

Bohan, Michael, Shelby G. Thompson, and Peter J. Samuelson. 2003. “Kinematic Analysis Of Mouse Cursor Positioning As A Function Of Movement Scale And Joint Set.” *Proceedings of the 8th Annual International Conference on Industrial Engineering – Theory, Applications and Practice.*

Boring, Sebastian, Marko Jurmu, and Andreas Butz. 2009. “Scroll, Tilt or Move It: Using Mobile Phones to Continuously Control Pointers on Large Public Displays.” P. 161 in *Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group on Design: Open 24/7 - OZCHI ’09*. Melbourne, Australia: ACM Press.

Calmes, L. (2019). Binaural sound source localization - Software. [online] Laurentcalmes.lu. Available at: http://www.laurentcalmes.lu/soundloc_software.html [Accessed 28 Jul. 2019].

Chen, Kuang-Hsuan, Jing-Jung Yang, and Fu-Shan Jaw. 2016. “Accelerometer-Based Fall Detection Using Feature Extraction and Support Vector Machine Algorithms.” *Instrumentation Science & Technology* 44(4):333–42.

Deborah J. Rumsey. “How to Interpret a Correlation Coefficient r.” *Dummies*. Retrieved March 28, 2019b (https://www.dummies.com/education/math/statistics/how-to-interpret-a-correlation-coefficient-r/).

Fang, L., S. Yishui, and C. Wei. 2016. “Up and down Buses Activity Recognition Using Smartphone Accelerometer.” Pp. 761–65 in *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*.

Ferrero, Renato, Filippo Gandino, Bartolomeo Montrucchio, Maurizio Rebaudengo, Alejandro Velasco, and Imane Benkhelifa. 2015. “On Gait Recognition with Smartphone Accelerometer.” Pp. 368–73 in *2015 4th Mediterranean Conference on Embedded Computing (MECO)*. Budva, Montenegro: IEEE.

Ikematsu, Kaori and Itiro Siio. 2015. “Memory Stones: An Intuitive Information Transfer Technique between Multi-Touch Computers.” Pp. 3–8 in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications - HotMobile ’15*. Santa Fe, New Mexico, USA: ACM Press.

Lee, Kangjae and Mei-Po Kwan. 2018. “Physical Activity Classification in Free-Living Conditions Using Smartphone Accelerometer Data and Exploration of Predicted Results.” *Computers, Environment and Urban Systems* 67:124–31.

Liu, Zhen-Tao, Min Wu, Wei-Hua Cao, Jun-Wei Mao, Jian-Ping Xu, and Guan-Zheng Tan. 2018. “Speech Emotion Recognition Based on Feature Selection and Extreme Learning Machine Decision Tree.” *Neurocomputing* 273:271–80.

Marquardt, Nicolai, Till Ballendat, Sebastian Boring, Saul Greenberg, and Ken Hinckley. 2012. “Gradual Engagement: Facilitating Information Exchange between Digital Devices as a Function of Proximity.” P. 31 in *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces - ITS ’12*. Cambridge, Massachusetts, USA: ACM Press.

Nancel, Mathieu, Olivier Chapuis, Emmanuel Pietriga, Xing-Dong Yang, Irani Pourang, and Michel Beaudouin-Lafon. 2013. “High-Precision Pointing on Large Wall Displays Using Small Handheld Devices.” Pp. 831–40 in *CHI ’13: SIGCHI Conference on Human Factors and Computing Systems*, edited by ACM. Paris, France.

Pew Research Center “Demographics of Mobile Device Ownership and Adoption in the United States.” Retrieved January 5, 2019 (http://www.pewinternet.org/fact-sheet/mobile/).

Paay, Jeni, Dimitrios Raptis, Jesper Kjeldskov, Mikael B. Skov, Eric V. Ruder, and Bjarke M. Lauridsen. 2017. “Investigating Cross-Device Interaction between a Handheld Device and a Large Display.” Pp. 6608–19 in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI ’17*. Denver, Colorado, USA: ACM Press.

Preece*, S. J., J. Y. Goulermas, L. P. J. Kenney, and D. Howard. 2009. “A Comparison of Feature Extraction Methods for the Classification of Dynamic Activities From Accelerometer Data.” *IEEE Transactions on Biomedical Engineering* 56(3):871–79.

Preece, S. J., J. Y. Goulermas, L. P. J. Kenney, and D. Howard. 2009. “A Comparison of Feature Extraction Methods for the Classification of Dynamic Activities From Accelerometer Data.” *IEEE Transactions on Biomedical Engineering* 56(3):871–79.

Rakhman, Arkham Zahri, Lukito Edi Nugroho, Widyawan, and Kurnianingsih. 2014. “Fall Detection System Using Accelerometer and Gyroscope Based on Smartphone.” Pp. 99–104 in *2014 The 1st International Conference on Information Technology, Computer, and Electrical Engineering*. Semarang, Indonesia: IEEE.

Rekimoto, Jun. 2004. “SyncTap: Synchronous User Operation for Spontaneous Network Connection.” *Personal and Ubiquitous Computing* 8(2):126–34.

Sarabadani Tafreshi, Amir E., Andrea Soro, and Gerhard Tröster. 2018. “Automatic, Gestural, Voice, Positional, or Cross-Device Interaction? Comparing Interaction Methods to Indicate Topics of Interest to Public Displays.” *Frontiers in ICT* 5.

Schmidt, Dominik, Julian Seifert, Enrico Rukzio, and Hans Gellersen. 2012. “A Cross-Device Interaction Style for Mobiles and Surfaces.” P. 318 in *Proceedings of the Designing Interactive Systems Conference on - DIS ’12*. Newcastle Upon Tyne, United Kingdom: ACM Press.

Seifert, Julian, Andreas Bayer, and Enrico Rukzio. 2013. “PointerPhone: Using Mobile Phones for Direct Pointing Interactions with Remote Displays.” Pp. 18–35 in *Human-Computer Interaction – INTERACT 2013*. Vol. 8119, edited by P. Kotzé, G. Marsden, G. Lindgaard, J. Wesson, and M. Winckler. Berlin, Heidelberg: Springer Berlin Heidelberg.

Strohmeier, Paul. 2015. “DisplayPointers: Seamless Cross-Device Interactions.” Pp. 1–8 in *Proceedings of the 12th International Conference on Advances in Computer Entertainment Technology - ACE ’15*. Iskandar, Malaysia: ACM Press.

Welch, P. 1967. “The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging over Short, Modified Periodograms.” *IEEE Transactions on Audio and Electroacoustics* 15(2):70–73.

Yan, Ke and David Zhang. 2015. “Feature Selection and Analysis on Correlated Gas Sensor Data with Recursive Feature Elimination.” *Sensors and Actuators B: Chemical* 212:353–63.

Yuan, H., C. Maple, C. Chen, and T. Watson. 2018. “Cross-Device Tracking through Identification of User Typing Behaviours.” *Electronics Letters* 54(15):957–59.

von Zadow, Ulrich, Wolfgang Büschel, Ricardo Langner, and Raimund Interactive Media Lab Dachselt. 2014. “SleeD: Using a Sleeve Display to Interact with Touch-Sensitive Display Walls.” Pp. 129–38 in *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces - ITS ’14*. Dresden, Germany: ACM Press.

Zhang T., Wang J., Xu L., Liu P. 2006 Fall Detection by Wearable Sensor and One-Class SVM Algorithm. In: Huang DS., Li K., Irwin G.W. (eds) *Intelligent Computing in Signal Processing and Pattern Recognition.* Lecture Notes in Control and Information Sciences, vol 345. Springer, Berlin, Heidelberg