Technische Universität Berlin

Institut fü Softwaretechnik und Theoretische Informatik
Quality and Usability Lab

Fakultaet IV
Franklinstrasse 28-29
10587 Berlin

Master Thesis

Predicting tap locations on touch screens in the field using accelerometer and gyroscope sensor readings

Emanuel Schmitt

Matriculation Number: 333772

Examined by:
Prof. Dr.-Ing. Sebastian Möller
Prof. Dr.-Ing. Axel Kuüper

Supervised by:
Dr.-Ing. Jan-Niklas Antons

Thanks to all dem brothas

Hereby I declare that I wrote this thesis myself with the help of no more than the mentioned literature and auxiliary means.

Berlin, 01.01.2050

. . . .

(Signature [your name])

Abstract

This template is intended to give an introduction of how to write diploma and master thesis at the chair 'Architektur der Vermittlungsknoten' of the Technische Universitï¿½t Berlin. Please don't use the term 'Technical University' in your thesis because this is a proper name.

On the one hand this PDF should give a guidance to people who will soon start to write their thesis. The overall structure is explained by examples. On the other hand this text is provided as a collection of LaTeX files that can be used as a template for a new thesis. Feel free to edit the design.

It is highly recommended to write your thesis with LaTeX. I prefer to use Miktex in combination with TeXnicCenter (both freeware) but you can use any other LaTeX software as well. For managing the references I use the open-source tool jabref. For diagrams and graphs I tend to use MS Visio with PDF plugin. Images look much better when saved as vector images. For logos and 'external' images use JPG or PNG. In your thesis you should try to explain as much as possible with the help of images.

The abstract is the most important part of your thesis. Take your time to write it as good as possible. Abstract should have no more than one page. It is normal to rewrite the abstract again and again, so probaly you won't write the final abstract before the last week of due-date. Before submitting your thesis you should give at least the abstract, the introduction and the conclusion to a native english speaker. It is likely that almost no one will read your thesis as a whole but most people will read the abstract, the introduction and the conclusion.

Start with some introductionary lines, followed by some words why your topic is relevant and why your solution is needed concluding with 'what I have done'. Don't use too many buzzwords. The abstract may also be read by people who are not familiar with your topic.

Zusammenfassung

Contents

List of Figures

List of Tables

1 Introduction

This chapter should have about 4-8 pages and at least one image, describing your topic and your concept. Usually the introduction chapter is separated into subsections like 'motivation', 'objective', 'scope' and 'outline'.

1.1 Motivation

Start describing the situation as it is today or as it has been during the last years. 'Over the last few years there has been a tendency... In recent years...'. The introduction should make people aware of the problem that you are trying to solve with your concept, respectively implementation. Don't start with 'In my thesis I will implement X'.

1.2 Objective

1.3 Outline

The 'structure' or 'outline' section gives a brief introduction into the main chapters of your work. Write 2-5 lines about each chapter. Usually diploma thesis are separated into 6-8 main chapters.

This example thesis is separated into 7 chapters.

Chapter 2 is usually termed 'Related Work', 'State of the Art' or 'Fundamentals'. Here you will describe relevant technologies and standards related to your topic. What did other scientists propose regarding your topic? This chapter makes about 20-30 percent of the complete thesis.

Chapter 3 analyzes the requirements for your component. This chapter will have 5-10 pages.

Chapter 4 is usually termed 'Concept', 'Design' or 'Model'. Here you describe your approach, give a high-level description to the architectural structure and to the single components that your solution consists of. Use structured images and UML diagrams for explanation. This chapter will have a volume of 20-30 percent of your thesis.

Chapter 5 describes the implementation part of your work. Don't explain every code detail but emphasize important aspects of your implementation. This chapter will have a volume of 15-20 percent of your thesis.

Chapter 6 is usually termed 'Evaluation' or 'Validation'. How did you test it? In which environment? How does it scale? Measurements, tests, screenshots. This chapter will have a volume of 10-15 percent of your thesis.

Chapter 7 summarizes the thesis, describes the problems that occurred and gives an outlook about future work. Should have about 4-6 pages.

2 Fundamentals and Related Work

2.1 User Interaction Inference

Inferring user interactions through side channels has been of great interest to the academic world throughout history. As this thesis will cover a modern approach by recording sensory information provided by the Apple iPhone, an early and prominent example of spying on emanations reaches far back in time.

Back in 1943, researchers of the TEMPEST project, a subdivision of the NSA[1], were able to infer information from the infamous Bell Telephone model 131-B2, a teletype terminal which was used for encrypting wartime communication. Using an oscilloscope, researchers could capture leaking electromagnetic signals from the device and by carefully examining the peaks of the recorded signals, the plain message the device was currently processing could be reconstructed. This technique was later advanced and used in the the Vietnam war. Through similar electric emanation the US military could detect approaching Viet Cong trucks giving them an immense competitive advantage.

Ever since, various user interaction inference experiments have been conducted by researchers worldwide. However, in order to categorize these different approaches found in literature, we will divide these based on the emanation channel that was spied on. This categorization approach is more suitable since device model and user interaction strategies have frequently changed in time.

The literature denoted that user inference can be perused with the help of acoustic, optical, electro-magnetic and on sensory emanation. We will describe these in the following sections.

2.1.1 Acoustic Emanation

One frequently discussed method of obtaining leaked information involves the utilization of the acoustic channel. Many electronic devices deploy tiny mechanics that generate sounds as a byproduct during interactions or during operation. These distinct sounds can differ in their characteristics making them reconvertable to the original information currently being processed by the machine. In these scenarios the eavesdropper targets a microphone in near proximity of the target device in order to capture the audio signals

[1]National Security Agency

the device is exposing on which learning algorithms are then applied.

In 2010, Backes et al. examined the problem of acoustic emanations of dot matrix printers, which where, at that time, still commonly used in banks and medical offices. By using a simple consumer-grade microphone, the researchers were able to recover whole sentences the printer was printing based on a record of sound. Backes et al. pre-processed the audio samples in order to extracted frequency-domain features feeding a hidden markov model, a technique commly used in audio speech recognition. As a result, the recognition system was able to reconstruct individual characters based on the sound inputs. To demonstrate a potential attack, the researchers deployed the system in a medical office where they were able to obtain up to 72% of the sentences being printed on a medical subscription [5].

Being inspired by the findings concerning the dot matrix printer, Asonov and Agrawal investigated acoustic emanations produced by hitting keystrokes on a desktop and notebook keyboard. Following their hypothesis that each key has a macroscopic difference in the construction mechanics as well and a distinct reverberation caused by the position in the board, individual keystrokes were recorded [2]. Researchers then extracted frequency domain features from the audio signals and passed them into a neural network. In an experiment performing 300 keystrokes, 79% of the characters could be correctly recognized [2]. As this technique required substantial training before recognition, other studies have reached similar accuracies using an unsupervised approach [35] on the one hand and by using acoustic dictionaries [6] on the other.

2.1.2 Optical Emanation

Optical emanations and reflections also pose a potential source for information. Computer screens, for instance, mediate information through breams of light which are captured by our human eye. However, as these rays can reflect upon other surfaces they are vulnerable of being captured by an eavesdropper.

Kuhn developed a technique to eavesdrop on cathode-ray tube monitors at distance. The information displayed on the monitor can be reconstructed from its distorted or even diffusely reflected light by using easy to access compontents such as a photomultiplier tube and a fast enough computer to process the recordings [14]. A similar approach that comprises reflections was shown by Backes et al., however focusing on LCD displays rather than CRT monitors. In this experiment, the researchers caught reflections in various objects that are commonly to be found in close proximity to a computer screen. Such objects included eyeglasses, tea pots, spoons and even plastic bottles. This work was later extended to additionally capture screens based on the reflections on the human eye's cornea [4].

2.1.3 Electro-magnetic Emanation

As electric currents flow through the through computer components, they emit electromagnetic waves to their near surrounding. These electromagnetic radiations can be picked up as side channels using sensitive equipment for the purpose of reconstructing these signals in form of data. Side-band electromagnetic emanations are present in keyboards [33], computer screens [32, 15], printers [27] and computer interfaces, such as USB 2 [24] and the parallel port [31].

A prominent example of eavesdropping on electromagnetic emanations was detected by the researcher van Eck discovering that cathode ray tube monitors could be spied upon from a distance using general market equipment [32]. With the use of antennas van Eck could receive the signals emitted from the monitor's cable. Since these cables only transmit the video signal for visualization, the researcher could display the visual output of the target monitor revealing a full screen cast of the original image. This attack is referred to in literature as *Van Eck Phreaking*.

[33] used electromagnetic emanations of wireless and cable keyboard to recover keystrokes from a distance [].
- Smart Cards [28]
- Wireless keyboards [33]
- Serial port cables [31]
- CMOS [1]
- CRT radiation [32]

2.1.4 Motion Emanation

In the past decade modern devices are increasingly equipeed with highly responsive sensors, such as the gyroscope and accelerometer enabling the devices to sense rich interactions with their environment. As user interactions, such as typing the keyboard or tapping on touchscreens, require the user to apply a certain force while entering information, this motion can be captured by motion sensors in order to be used for as a side-channel attack reconstructing secret information [22, 25, 7].

Marquardt et al. conducted an experiment where an Apple iPhone equpped with an application that captures motion with it's accelerometer was placed next to a keyboard. Subjects then had to enter sentences on the keyboard while the application was monitoring the user's motion while typing. Furthermore, The researchers could then decode the accelerometer signals by measuring the relative physical position and distance between each vibration. The decoded characters were then matched based on a dictionary containing a frequency distribution of commonly used word. As a result, words were successfully obtained with an accuracy of up to 80% [20].

2.2 Eavesdropping on touch screen user interactions

As we have seen in the previous section, user interactions with peripheral devices, such as the keyboard or PIN pads, can be obtained by either the acoustic channel, by electromagnetical leakage or by capturing the motion of the user. However, with the rise in soft keyboard usage, the same discussed methods that extract information from keyboard do not apply to tap interactions on a touchscreen surface. Since a touchscreen does not embody fine mechanics producing sounds nor does it have emanating cables, the research community has developed nouvelle methods to spy on user inputs based on the smartphone embodied motion sensors.

The general idea behind the three approaches that are going to be discuss in the following is that a tap, or to be more precise the magnitude of the force of a tap, on a specific touch screen location creates an identifiable pattern on the motion sensors that can be sufficient to infer the initial tap location. This is particularly interesting since motion sensors are not considered as being privacy-sensitive and therefore lack access restrictions by the operating systems of the devices.

The first paper regarding this security threat was published by Cai and Chen. In their proof-of-concept study they created an Android[2] application which displays a 10-digit PIN-pad. During interactions with the PIN-pad, the accelerometer signals were monitored and used for later data analysis. Having observed that a tap movement affects the rotation angle of the screen, the researchers handcrafted features based on the path of the *pitch*[3] and the *roll*[4] angles of the accelerometer. These were intersected to find a dominating edge on where the tap had presumably taken place. By using a probability density function for a Gaussian distribution the researchers were able to achieve an average accuracy of 70% for interred PIN-pad digits. The training set size involved 449 pin strokes [7]. Even though *Touchlogger* was a promising first step, due to it's low granularity of only 10 distinguishable large screen areas, it remained unclear if the attack can be carried over to a full software keyboard. Furthermore, since the inference was performed on only a single smartphone model, the question is left open whether other smartphones or tablet computers are similarly vulnerable.

In order to show the feasibility for a full software keyboard, Owusu et al. performed a second attempt to the problem by creating *ACCessory*. *ACCessory* is an Android application with functionalities similar to the previously mentioned *Touchlogger*. However, the application significantly differ in it's tap area granularity providing two separate modes for tap inputs: *area mode* and *character mode*. *area mode* consists of tap areas arranged in a 60-cell grid, whereas a QWERTY keyboard within landscape orientation was displayed in *character mode*. Having extracted features mainly from the time-

[2]Android operating system for smartphones by Google Inc.
[3]The pitch-angle corresponds to the x-axis of the accelerometer.
[4]The roll angle corresponds to the y-axis of the accelerometer.

domain, a classification using the Random Forests algorithm reached an accuracy of 24.5% for the 60-cell grid. Here, the corresponding dataset consisted of 1300 keystrokes collected from 4 participants. As the *area mode* experiment focused on recognizing individual keystrokes, the *character mode* experiment focused on cracking passwords. By combining the keystrokes into a sequence and assuming recognition errors in individual characters, the researchers could create a ranked list of candidate passwords by running a maximum likelihood search for the most probable classification errors for an obtained password. Here, 6 out of 99 password could be inferred under 4.5 median trials given that one trails refers to traversing down one item of the candidate list. Furthermore, the majority of 59 out of 99 passwords could be inferred within 2^{15} median trials. As general result, even though the overall accuracy of the learning system scored low, the researchers could significantly reduce the search space for reconstructing a password indicating that accelerometer readings can in deed yield confidential information.

The most comprehensive study to date regarding the topic was conducted by Miluzzo et al. and differs from *ACCessory* and *Touchlogger* in many important ways. While both previous studies are both evaluated on the Android smartphones, *TapPrints* investigates the tap inference on both iOS and Android operating systems including tablets and smartphones alike. Another important point of differentiation is the used learning system. In order to raise the level of entropy, *TapPrints* combines readings from the accelerometer and gyroscope for a more sophisticated feature extraction. Here, time-domain and frequency-domain features, as well as the correlation and angles between individual sensor components are considered. For classification purposes, the researchers use an ensemble method combining decision tress, support vector machines, k-nearest neighbors and multinomial logistic regression in a winner-takes-it-all[5] voting fashion. The dataset collected in this experiments contains over 40.000 individual taps collected from 10 different users. In addition, the researchers also requested user to use different input modalities while typing including the usage of the index finger and thumb.

The *TapPrints* undertaking consists of two separate experiments: The first is a icon tapping experiment where icons are arranged in a 20 cell grid and the second one being a letter tapping experiment involving the standard software keyboard offered by the operating system. In the first experiment, an average accuracy of 78% was achieved for the iPhone whereas 67% of icon taps could be correctly inferred on the Android device. In the letter tapping experiment users were asked to enter pangram[6] sentences on the OS soft-keyboard. Results showed that on both iPhone and Android an average of 43% of the letters could be correctly classified. Even though the average accuracy for individual letters were not particularly high, Miluzzo et al. could show that when pangrams were repeatedly entered, a majority vote could be applies to individual character recognitions allowing to recover the whole pangram in approx. 15 trials. To conclude, *TapPrints*

[5] This implies that all classifiers classify separately and the classifier with the highest prediction score wins.

[6] A pangram is a sentence using every letter of a given alphabet at least once.

could demonstrate that motion sensor can be used to obtain passwords on multiple platforms and formats and with different input modalities.

Since the data used in *TapPrints* and in the other related work was collected in a controlled environment [25, 7, 22], it is not possible to tell if the feasibility of tap inference will also apply to data collected in the field environment. It is plausible that the environment the user is currently in, has an effect on the recorded taps. If we imagine a user sitting in a moving vehicle, the motion sensor will include the vibrations of the motor. Therefore, this work will aim at collecting taps and corresponding motion sensor data from both a laboratory and a field environment to test how prediction accuracies compare to both environments. For this purpose, an iOS application with different tap area grids, as we have seen in *ACCessory* and *Touchlogger*, will be evaluated in the upcoming sections.

2.3 Machine Learning

As we will be using machine learning techniques for the later classification of sensory data, the following chapter will give a brief overview of the fundamental concepts evolving around statistical learning.

2.3.1 Overview and Definition

Ever since computers were invented, there has been a desire to enable them to learn [30]. This desire has grown into the field of machine learning which seeks to answer questions on how to build build systems that automatically improve with experience, and what the fundamental laws of learning processes are. Today, state-of-the-art ML covers a large set of methods and algorithms designed to accomplish tasks where conventional hard-coded routines have brought insufficient results. From speech recognition to email spam detection or recommendation systems, ML methods find broad usage in a variety of problem domains.

In order to understand what the principle of machine learning is, we will start with a definition by Samuel [30]:

> *Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed.*

In this definition, special emphasis is to be put on the last part of this definition. A computer is only then able to learn when he can perform a task without being explicitly instructed. Thus, in order to learn, the computer must somehow be able to instruct itself without the influence of an outer . As this definition lacks a more detailed view on what computer learning is, we will dive into a definition by Tom Mitchell [23]:

> *A learning system is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

The example that Mitchell notes, is one from the games of checkers [23]. In this case checkers is the task T that the computer is aiming to learn. In order for the computer to learn, information on previously played matches is required. Since the computer does not know how to evaluate is a particular match was either good or bad, we set the performance to be defined based on how many matches were actually won. If a computer program can raise the amount of games won *(performance measure P)* with the help of the experience from previously placed matches *(experience E)* then it can learn to play checkers *(task T)*.

To break this down into a more practical perspective, the challenge lies in finding an appropriate model in order to learn from data, which is the most common format to represent past experience. By learning, the computer adjusts parameters on the model based on the data that we feed the system with. Once the model has been adjusted, it can perform tasks with new incoming experience.

2.3.2 Categorization of Methods

As machine learning algorithms and methods differ from their approach to learning and underlying concepts, it is common practice to separate these into the following categories [9, 21] : Supervised learning, unsupervised learning, reinforcement learning and evolutionary learning. In the following sections I will briefly outline these.

Supervised learning, which is also named learning from example, is presumably the most prominent category of ML algorithms. The algorithm is given a training set of examples $\{x_0, \ldots, x_n\}$, which are also known as *features* and the correct target values $\{y_0, \ldots, y_n\}$ mapped to each set of features, which is the answer that the algorithm should produce. The algorithm then generalizes based on the training set in order to respond with sensible outputs on all possible input values. Outputs, if they are discrete labels, correspond to a classification task whereas outputs on a continuous scale refer to a regression task (see [21]).

An example for supervised learning is the classification of malignant or benign tumors as seen in cancer diagnosis. Let's assume we have a dataset with different properties of a tumor, such as the size or the color of the cells. These properties form our features x. Each set of features is mapped to an output label y stating if the tumor is malignant or benign. The first step is to use the pairs (x, y) of the training set to teach the algorithm the correct mapping of the problem space. As x is linked to the output y in the training set, learning the conjunction of these two values is done under supervision since the output label y is given. Once learned, the algorithm is generalized to map unseen inputs to the correct output label.

Practical applications are for example digit and handwriting recognition [16], spam filtering [10] for e-mails or network anomaly detection [17].

Presumably the most widely known machine learning techniques belong to this category, such as Support Vector Machines (SVMs), Artificial Neural Networks, Bayesian Statistics, Random Forests and Decision Trees [9].

Unsupervised learning is the task of learning structures from input values that are not explicitly labeled. In comparison to supervised learning, where correct output values are provided for each input, unsupervised algorithms learn to identify similarities in the input data and can therefore group these [9]. These grouping problems are referred to as *clustering*. The underlying idea here, is that humans learn by not explicitly being told what the right answer should be [21]. If a human sees different species of snakes, for instance, he or she is able to identify them all as snakes. Hence, the human is aware that there are differences in each specific type of snake without specifically knowing a correct label.

A prominent example where unsupervised learning is heavily used, is in recommender systems for online retail shops. Amazon.com, for instance, uses a technique called *collaborative filtering*, which measures similarity in customers based that they have previously bought [18]. Having identified similar customers utilizing the cosine similarity, the algorithm can then recommend items that similar users have bought. This technique is also used for music recommendations [26] or social network recommendations [13].

The field of unsupervised learning is closely related to density estimation in statistics, as with the density of inputs, we are able to group them. The K-means algorithm is the most prominent in this field [21].

Reinforcement learning falls in between supervised and unsupervised learning methods. Whereas supervised learning tries to bridge the gap between input and corresponding output values and unsupervised methods detect groupings in incoming data, reinforcement learning is based on learning with a *critic* [21]. The algorithm tries different solution strategies to a problem and is told weather or not the answer provided was correct. An important fact here, is that the algorithm is not told how to correct itself. This practice of "trying-out" is based on the concept of *trail-and-error learning* which is known as the *Law-of-effect* [21]. A good example is a child that tries to stand up and learn walking. The child tries out many different strategies for staying upright and receives feedback from the field based on how long it can stand without falling down again. The method that previously worked best is then repeated in order to find the optimal solution resulting in the child learning to walk [21].

In more mathematical terms, the reinforcement learning problem is formalized with an agent and his environment. The environment in which the agent is set provides a set of *states* on which the agent can perform *actions* to maximize a certain *reward*. By performing actions the state changes and a new reward is calculated. The reward then tells the agent if the action was a good choice. Goal of the algorithm is to maximize the reward [21].

Reinforcement learning is a practical computational tool for constructing autonomous systems that improve themselves with experience. These applications have ranged from robotics, to industrial manufacturing, to combinatorial search problems such as computer game playing [12]. Prominent methods of this category are Q-learning, Monte Carlo methods and Hidden Markov Models [21].

Evolutionary learning is inspired by strongly inspired by nature. As biological evolution improves the survival of a species, the strategy of adaptation to improve survival rates and the chance of offspring has inspired researchers to craft genetic algorithms (GA) [21].

Genetic algorithms are a family of adaptive search procedures which have derived their name from the fact that they are based on models of genetic change in a population of individuals. These models have their foundation in three basic ideas: (1) Each evolutionary state of a population can be evaluated on a *fitness* scale. This is done since biological evolution has a natural bias towards animals that are "fitter" than others. These animals tend to live longer, are more attractive and generate healthier and happier offspring, an idea which was originated in Charles Darwin's "The Origin of Species". (2) Each population can be mated to generate offspring using a *mating operator*. (3) The third component are *genetic operator*, such as *crossover* and *mutation*, which determine how the offspring solution is composed of the genetic material of the parents [8].

Evolutionary learning is often considered when other methods fail to find a reasonable answer. Algorithms find applications in search and mathematical optimization, but also in arts and simulation [21].

In this section we have seen several different problems that we can solve with the help of algorithmic learning. For our use case, as we want to predict the locations on smartphone screens using sensory data. As this is a supervised learning problem, we will cover one supervised approach in more detail in the following section: Artificial neural networks.

2.3.3 Artificial Neural Networks

2.3.4 Regularization

2.3.5 Optimization

2.4 Data Aquisition in the field

- topics on data aquisition in the field

3 Implementation

For labeled data acquisition a system was required that is able to function in a laboratory environment, as well as recieve the data coming from the field study. For this purpose TapSensing was created. TapSensing is an iOS application that collects touch events including their sensory information to then send them to a backend server application. In the following sections we will outline the different components of TapSensing.

3.1 System Architecture

The TapSensing application comprises to two main components, one being the mobile application and the other being the server-side application. The mobile client is responsible for generating the sensor and tap information by providing a user interface for the user to tap. For the data to be stored in a centralized manner, the backend provides endpoints as a gateway to the database.

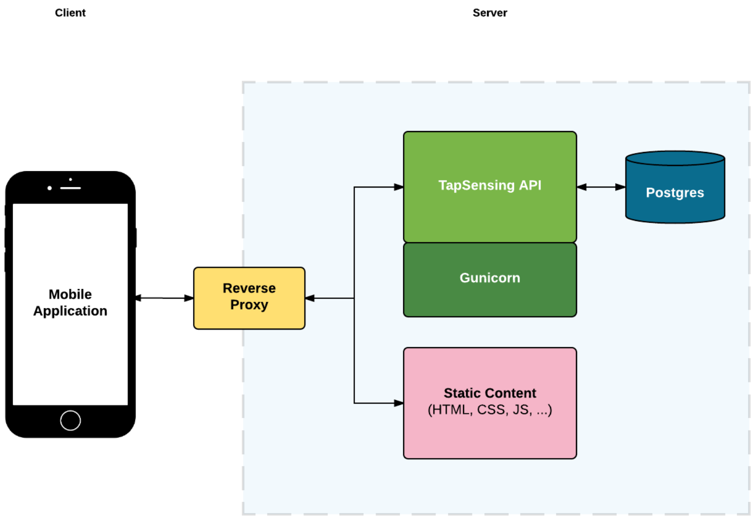

Figure 3.1: TapSensing architecture overview.

For the network requests containing the tap information which come from the mobile device to reach the backend, it must first pass through a reverse proxy. As reverse proxy we have chosen NGINX due to it's easy configurability. In this case, NGINX forwards requests to the TapSensing application and serves static files.

The TapSensing backend is written upon the Python Django Framework[1] which is being executed upon the gunicorn application server. Django uses a so-called ORM to perform transactions with the Database, which in our case is a PostgreSQL database.

3.2 Mobile Application

3.2.1 Interface

Tap Input

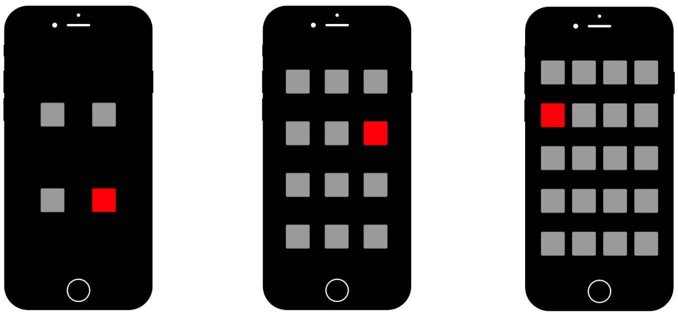

Figure 3.2: Grid sizes of Grid.

For the user to be able to enter the taps into the device the app provides a screen with buttons aligned in grip structures.

The positions of the buttons are calculated based on the configuration that is set. Here, the amount of buttons in height and width can be adjusted.

The source code of the of the question view is to be found in `GridViewController.swift`.

Questions

To obtain more information on the taps provided in the tap input view, the application provides views for the user to answer several questions concerning input modalities, body posture and mood. After the tap input screen, the questions are displayed providing multiple answer choices. In addition, the application provides a pictogram for each answer option.
Each view is generically set based on the questions and answer possibilities. Once the user taps on an icon the view transitions to the next question view.

The source code of the of the question view is to be found in `QuestionViewController.swift`.

[1] https://www.djangoproject.com/

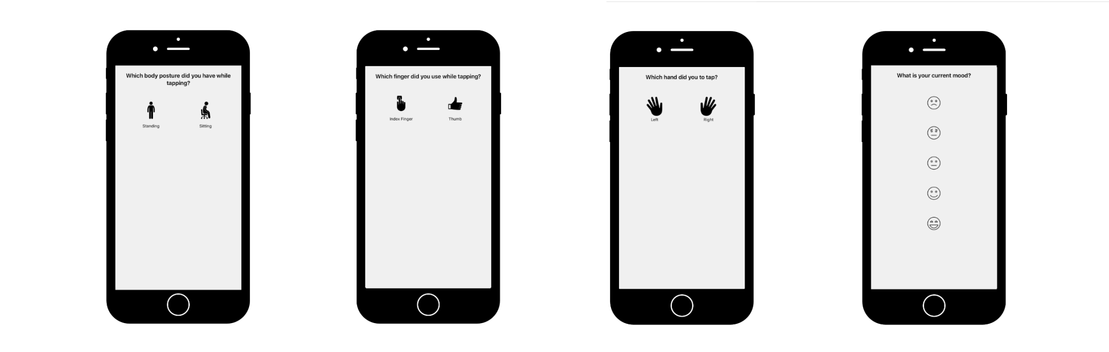

Figure 3.3: The figure displays question views with icons as answer possibilities.

Upload

After all taps and additional information is gathered, we show a view that the application is uploading the data to the server-side application.

3.2.2 Smartphone Sensors

Modern smartphones come with a variety of different sensors offering valuable services to it's users and enhancing many applications. The newest Apple iPhone to date, the iPhone 7, has a fingerprint sensor, a barometer, a three-axis gyroscope and accelerometer (MEMS), a proximity sensor and an ambient light sensor attached to it's main-board [11]. As we are going to predict finger taps on the iPhone screen, the only sensors that are effected by the force of the tap are the gyroscope and the accelerometer. Therefore, these will be outlined in the following sections.

Accelerometer

The accelerometer is a sensor module that measures the acceleration it encounters by either movement or gravity [19]. However, the acceleration caused by movement, the so-called inertial acceleration and the gravitational acceleration can not be distinguished by the sensor. This is due to Einstein's equivalence principal stating that the effects of gravity on an object are indistinguishable from the acceleration of the object's reference frame [29].

When the position of the device is fixed, as for example when it is placed on a table the accelerometer values would yield $a = \{a_x = 0, a_y = 0, a_z = -1\}$. This feature make it suitable for detecting device screen rotations. As the device flips from landscape to portrait orientation, the gravity is sensed by a different set of accelerometer axis [19].

The values of the acceleration are quantified in the SI unit metres per second per second (m/s^2). However, in engineering the acceleration is typically expressed in terms of the standard gravity (g).

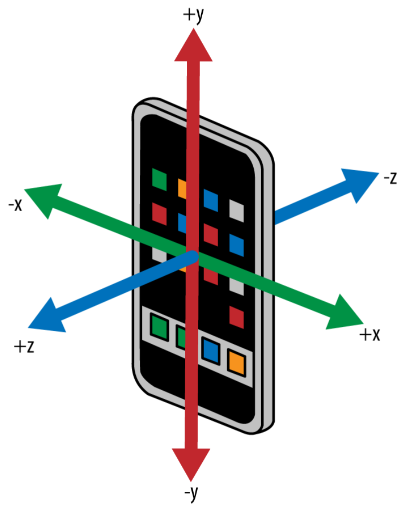

Figure 3.4: Apple iPhone with the corresponding axes of the accelerometer.

Gyroscope

As the accelerometer is suitable for detecting orientations, it lacks the ability to detect spin or more precise rotation movements. These spin movements are detected by the gyroscope sensor which is responsible for detecting and maintaining orientation [19].

A mechanical gyroscope typically composes of a spinning wheel which is set within three so-called gimbals. These gimbals enable the spinning wheel to be set in any orientation. Although the orientation does not remain fixed when the device is rotated, it changes in response to an external torque much lesser and in a different direction than it would be without the large angular momentum associated with it. Each gimbals translates to one of the three gyroscope outputs, namely *pitch*, *roll* and *jaw* (See [34]).

The gyroscope sensor within the MEMS[2], the chip deployed in the iPhone, is between 1 to 100 micrometers of size. When the gyroscope is rotated, a small resonating mass is shifted as the angular velocity changes. This movement is converted into very low-current electrical signals that can be amplified and read by a host system.

The values of the gyroscope are quantified in as rotations per seconds (RPS) or as degrees per second (deg $/s$).

[2]Microelectromechanical systems

Accessing sensor values

In order to access gyroscope and accelerometer Apple provides a high level API[3] for accessing the device's sensors: `Core Motion`. Core Motion reports motion and environmental related data from sensors including accelerometers, gyroscopes, pedometers, magnetometers, and barometers in easy to use manner.

Sensor values can either be accessed as proceeded version including aggregations of the values and a raw version. For TapSensing, we make sure to record the raw values to avoid any form of bias. The update interval can be configures at ranges from 10Hz - 100Hz. Higher update-rates are possible but are not ensured to be processed in real-time by the device. For TapSensing, the update rate is configured with the highest (safe) value possible. This ensures that tap patterns are captured with an accurate resolution to make a later classification easier. The figure below is a code snipping depicting how sensor values are retrieved in the TapSensing application.

Figure 3.5: Swift code snippet displaying how to access sensor values with Core Motion.

Interoperability -JSON **Scalability** -Machine

3.3 Backend application

3.3.1 General requirements

Security

Consistency

3.3.2

3.3.3 Data model

3.3.4 User management & Security

3.3.5 Ensuring consistency

[3] An Application Program Interface is a set of rules and subroutines provided by an application system for the developer to use. Here is a link to the Core Motion API Documentation: `https://developer.apple.com/documentation/coremotion/`

4 Inferring Tap Locations

4.1 Data Preprocessing

4.2 Convolution Neural Network

4.3 Recurrent Neural Network

5 Method

5.1 Overview

5.2 Hypothesis

5.3 Experimental Setup

5.3.1 Subjects

5.3.2 Devices

5.3.3 Experiment Settings

Lab

Field

5.4 Analysis

6 Results

6.1 Hypothesis

6.2 Discussion

7 Conclusion

7.1 Further Outlook

List of Acronyms

3GPP	3rd Generation Partnership Project
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
AS	Application Server
CSCF	Call Session Control Function
CSS	Cascading Stylesheets
DHTML	Dynamic HTML
DOM	Document Object Model
FOKUS	Fraunhofer Institut fuer offene Kommunikationssysteme
GUI	Graphical User Interface
GPS	Global Positioning System
GSM	Global System for Mobile Communication
HTML	Hypertext Markup Language
HSS	Home Subscriber Server
HTTP	Hypertext Transfer Protocol
I-CSCF	Interrogating-Call Session Control Function
IETF	Internet Engineering Task Force
IM	Instant Messaging
IMS	IP Multimedia Subsystem
IP	Internet Protocol
J2ME	Java Micro Edition
JDK	Java Developer Kit
JRE	Java Runtime Environment
JSON	JavaScript Object Notation
JSR	Java Specification Request
JVM	Java Virtual Machine
NGN	Next Generation Network
OMA	Open Mobile Alliance
P-CSCF	Proxy-Call Session Control Function
PDA	Personal Digital Assistant
PEEM	Policy Evaluation, Enforcement and Management
QoS	Quality of Service
S-CSCF	Serving-Call Session Control Function
SDK	Software Developer Kit
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SMS	Short Message Service

SMSC	Short Message Service Center
SOAP	Simple Object Access Protocol
SWF	Shockwave Flash
SWT	Standard Widget Toolkit
TCP	Transmission Control Protocol
Telco API	Telecommunication API
TLS	Transport Layer Security
UMTS	Universal Mobile Telecommunication System
URI	Uniform Resource Identifier
VoIP	Voice over Internet Protocol
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
XCAP	XML Configuration Access Protocol
XDMS	XML Document Management Server
XML	Extensible Markup Language

Bibliography

[1] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. *The EM Side—Channel(s)*, pages 29–45. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. ISBN 978-3-540-36400-9. doi: 10.1007/3-540-36400-5_4. URL `https://doi.org/10.1007/3-540-36400-5_4`.

[2] D. Asonov and R. Agrawal. Keyboard acoustic emanations. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pages 3–11, May 2004. doi: 10.1109/SECPRI.2004.1301311.

[3] M. Backes, M. DÃ¼rmuth, and D. Unruh. Compromising reflections-or-how to read lcd monitors around the corner. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 158–169, May 2008. doi: 10.1109/SP.2008.25.

[4] M. Backes, T. Chen, M. Duermuth, H. P. A. Lensch, and M. Welk. Tempest in a teapot: Compromising reflections revisited. In *2009 30th IEEE Symposium on Security and Privacy*, pages 315–327, May 2009. doi: 10.1109/SP.2009.20.

[5] Michael Backes, Markus Dürmuth, Sebastian Gerling, Manfred Pinkal, and Caroline Sporleder. Acoustic side-channel attacks on printers. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, pages 20–20, Berkeley, CA, USA, 2010. USENIX Association. ISBN 888-7-6666-5555-4. URL `http://dl.acm.org/citation.cfm?id=1929820.1929847`.

[6] Yigael Berger, Avishai Wool, and Arie Yeredor. Dictionary attacks using keyboard acoustic emanations. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 245–254, New York, NY, USA, 2006. ACM. ISBN 1-59593-518-5. doi: 10.1145/1180405.1180436. URL `http://doi.acm.org/10.1145/1180405.1180436`.

[7] Liang Cai and Hao Chen. Touchlogger: Inferring keystrokes on touch screen from smartphone motion. In *Proceedings of the 6th USENIX Conference on Hot Topics in Security*, HotSec'11, pages 9–9, Berkeley, CA, USA, 2011. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=2028040.2028049`.

[8] Kenneth De Jong. Learning with genetic algorithms: An overview. *Machine learning*, 3(2):121–138, 1988.

[9] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000. ISBN 0471056693.

[10] Thiago S Guzella and Walmir M Caminhas. A review of machine learning approaches to spam filtering. *Expert Systems with Applications*, 36(7):10206–10222, 2009.

[11] Apple Inc. iphone 7 - technical specifications. URL `https://www.apple.com/iphone-7/specs/`.

[12] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[13] Henry Kautz, Bart Selman, and Mehul Shah. Referral web: combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, 1997.

[14] M. G. Kuhn. Optical time-domain eavesdropping risks of crt displays. In *Proceedings 2002 IEEE Symposium on Security and Privacy*, pages 3–18, 2002. doi: 10.1109/SECPRI.2002.1004358.

[15] Markus G Kuhn. Electromagnetic eavesdropping risks of flat-panel displays. In *Privacy Enhancing Technologies*, volume 3424, pages 88–107. Springer, 2004.

[16] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[17] Kyumin Lee, James Caverlee, and Steve Webb. Uncovering social spammers: social honeypots+ machine learning. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 435–442. ACM, 2010.

[18] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

[19] Ming Liu. A study of mobile sensing using smartphones. *International Journal of Distributed Sensor Networks*, 9(3):272916, 2013. doi: 10.1155/2013/272916. URL `http://dx.doi.org/10.1155/2013/272916`.

[20] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. (sp)iphone: Decoding vibrations from nearby keyboards using mobile phone accelerometers. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 551–562, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0948-6. doi: 10.1145/2046707.2046771. URL `http://doi.acm.org/10.1145/2046707.2046771`.

[21] Stephen Marsland. *Machine Learning: An Algorithmic Perspective.* Chapman & Hall/CRC, 1st edition, 2009. ISBN 1420067184, 9781420067187.

[22] Emiliano Miluzzo, Alexander Varshavsky, Suhrid Balakrishnan, and Romit Roy Choudhury. Tapprints: Your finger taps have fingerprints. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, pages 323–336, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1301-8. doi: 10.1145/2307636.2307666. URL http://doi.acm.org/10.1145/2307636.2307666.

[23] Tom Michael Mitchell. *The discipline of machine learning*, volume 3. Carnegie Mellon University, School of Computer Science, Machine Learning Department, 2006.

[24] Leszek Nowosielski and Marian Wnuk. Compromising emanations from usb 2 interface. In *PIERS Proceedings*, 2014.

[25] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. Accessory: Password inference using accelerometers on smartphones. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, HotMobile '12, pages 9:1–9:6, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1207-3. doi: 10.1145/2162081.2162095. URL http://doi.acm.org/10.1145/2162081.2162095.

[26] Javier Pérez-Marcos and Vivian López Batista. Recommender system based on collaborative filtering for spotify's users. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 214–220. Springer, 2017.

[27] Rafal Przesmycki. Measurement and analysis of compromising emanation for laser printer. In *PIERS Proceedings*, pages 2661–2665, 2014.

[28] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *Proceedings of the International Conference on Research in Smart Cards: Smart Card Programming and Security*, E-SMART '01, pages 200–210, London, UK, UK, 2001. Springer-Verlag. ISBN 3-540-42610-8. URL http://dl.acm.org/citation.cfm?id=646803.705980.

[29] W. T. Read. Handbook of engineering fundamentals (eshbach, ovid w., ed.). *Journal of Chemical Education*, 14(1):49, 1937. doi: 10.1021/ed014p49.2. URL http://dx.doi.org/10.1021/ed014p49.2.

[30] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 44(1.2):206–226, 2000.

[31] Peter Smulders. The threat of information theft by reception of electromagnetic radiation from rs232 cables. 9:53–58, 02 1990.

[32] Wim van Eck. Electromagnetic radiation from video display units: An eavesdropping risk? *Comput. Secur.*, 4(4):269–286, December 1985. ISSN 0167-4048. doi: 10.1016/0167-4048(85)90046-X. URL http://dx.doi.org/10.1016/0167-4048(85)90046-X.

[33] Martin Vuagnoux and Sylvain Pasini. Compromising electromagnetic emanations of wired and wireless keyboards. In *Proceedings of the 18th Conference on USENIX Security Symposium*, SSYM'09, pages 1–16, Berkeley, CA, USA, 2009. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=1855768.1855769`.

[34] Wikipedia. Gyroscope — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Gyroscope&oldid=793494393`, 2017. [Online; accessed 09-August-2017].

[35] Li Zhuang, Feng Zhou, and J. D. Tygar. Keyboard acoustic emanations revisited. *ACM Trans. Inf. Syst. Secur.*, 13(1):3:1–3:26, November 2009. ISSN 1094-9224. doi: 10.1145/1609956.1609959. URL `http://doi.acm.org/10.1145/1609956.1609959`.

Annex

```xml
<?xml version="1.0" encoding="UTF-8"?>
<widget>
        <debug>off</debug>
        <window name="myWindow" title="Hello Widget" visible="true">
                <height>120</height>
                <width>320</width>
                <image src="Resources/orangebg.png">
                        <name>orangebg</name>
                        <hOffset>0</hOffset>
                        <vOffset>0</vOffset>
               </image>
                <text>
                         <name>myText</name>
                         <data>Hello Widget</data>
                         <color>#000000</color>
                         <size>20</size>
                         <vOffset>50</vOffset>
                         <hOffset>120</hOffset>
                </text>
        </window>
</widget>
```

Listing 1: Sourcecode Listing

```
INVITE sip:bob@network.org SIP/2.0
Via: SIP/2.0/UDP 100.101.102.103:5060;branch=z9hG4bKmp17a
Max—Forwards: 70
To: Bob <sip:bob@network.org>
From: Alice <sip:alice@ims—network.org>;tag=42
Call—ID: 10@100.101.102.103
CSeq: 1 INVITE
Subject: How are you?
Contact: <sip:xyz@network.org>
Content—Type: application/sdp
Content—Length: 159
v=0
o=alice 2890844526 2890844526 IN IP4 100.101.102.103
s=Phone Call
t=0 0
c=IN IP4 100.101.102.103
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000

SIP/2.0 200 OK
Via: SIP/2.0/UDP proxy.network.org:5060;branch=z9hG4bK83842.1
;received=100.101.102.105
Via: SIP/2.0/UDP 100.101.102.103:5060;branch=z9hG4bKmp17a
To: Bob <sip:bob@network.org>;tag=314159
From: Alice <sip:alice@network.org>;tag=42
Call—ID: 10@100.101.102.103
CSeq: 1 INVITE
Contact: <sip:foo@network.org>
Content—Type: application/sdp
Content—Length: 159
v=0
o=bob 2890844526 2890844526 IN IP4 200.201.202.203
s=Phone Call
c=IN IP4 200.201.202.203
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Listing 2: SIP request and response packet[**?**]