Institute for Visualization and Interactive Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 302

Developing SurfaceSliding - A New Interaction for Smartphones

Phi Hung Dang

Course of Study:	Informatik
Examiner:	Jun.-Prof. Niels Henze
Supervisor:	Dipl.-Inf. Sven Mayer, Dipl.-Inf. Lars Lischke
Commenced:	18. Januar 2016
Completed:	19. Juli 2016
CR-Classification:	H.5.2

Einleitung

Basierend auf früheren Arbeiten und der Tatsache, dass etwa die Hälfte der Nutzer dazu neigen ihr Smartphone auf den Tisch zu legen, während sie sich zuhause oder im Büro befinden, haben wir uns dazu entschlossen eine neue Interaktionsmethode zu entwickeln, die es dem Nutzer erlaubt mit dem Smartphone zu interagieren ohne es dabei vom Tisch aufnehmen zu müssen. Stattdessen erfolgt die Interaktion, genannt *SurfaceSliding*, durch das physikalische Schieben des Smartphones über den Tisch. Nachdem wir die Trägheitssensoren von heutigen Smartphones in Hinblick auf ihre Brauchbarkeit für das Erfassen der Schiebebewegungen untersucht haben, entschieden wir uns letztendlich dafür, unsere Lösung auf einen optischen Fluss Algorithmus zu basieren, der mithilfe der Bilder der Frontkamera des Smartphones arbeitet. Am Ende unserer Arbeit implementierten wir einen Anwendungsfall in unser System und führten daraufhin eine Nutzerstudie durch, die die Brauchbarkeit und Effizienz von *SurfaceSliding* im Vergleich zu einer mehr konventionellen Interaktionsmethode untersucht. Obwohl unsere Ergebnisse zeigten, dass Leute beim Benutzen von *SurfaceSliding* mehr Zeit benötigten als auch mehr Fehler machten, teilten sie die Meinung, dass sie in *SurfaceSliding* eine zusätzliche Alternative sehen um mit Mobilgeräten zu interagieren.

Abstract

Motivated by previous work and the fact that about half of the users tend to place their phone on the table while being at home or at the office, we developed a new interaction method for smartphones which enables the interaction with the smartphone without actually picking it up. Instead, our new interaction method, called *SurfaceSliding*, focuses on the interaction through physically sliding the smartphone on the table. Investigating the suitability of the inertial sensors from nowadays smartphones in order to track the sliding movements at first, we eventually decided to implement an optical flow based solution using the images captured by the phone's front camera. After implementing one use case within our system, we conducted a comparable user study for investigating the usability and efficiency of *SurfaceSliding* when comparing it to a more traditional input method like tapping. Although our results showed higher task completion times and higher error rates when using *SurfaceSliding*, our participants shared the opinion of having *SurfaceSliding* as an additional alternate input method for mobile devices.

Contents

List of Figures

List of Tables

List of Acronyms

EF effort

ER error rate

FR frustration

KLM Keystroke-Level Model

MD mental demnad

PD physical demand

PF performance

SUS System Usability Scale

TCT task completion time

TD temporal demand

TLX Task Load Index

TW total workload

1 Introduction

Throughout the past years, smartphones and tablets have established themselves as a vital part of our everyday life. They have become one of the most popular devices for performing various tasks such as reading emails, browsing the Internet or playing mobile games. However most importantly, they help us to stay connected with our family and friends, our surroundings and also people from all around the world. As a result, a common aspect of our life focuses on keeping those devices in our reachable distance, e.g. trouser pockets, backpacks or on nearby tables. Wiese et al. [WSB13] have shown that about half of their participants tend to place their phone on the table while being at home or at the office. Thus, it is often required to physically pick up the phone in order to interact with it, e.g. declining a call or opening a chat message. We experienced this pick up as an unnecessary action since the user often only performs one or two touch actions or only wants to have a glimpse of the phone's screen.
In this work, we develop a new interaction technique, called *SurfaceSliding*, which focuses on enabling the interaction with a smartphone while saving the effort of actually picking it up. *SurfaceSliding* allows the user to interact with the phone by physically sliding it over a flat surface, e.g. on a table or a shelf. Inspired by other interaction technique and the fact that nowadays smartphones are equipped with numerous different sensors, we make use of the smartphone's inertial sensors in order to detect and track the various sliding actions performed by the user. Besides developing a working prototype, the thesis also elaborates and implements one use case which benefits from this input method. In the end, we analyze the implementation in regard to its reliability, usability and efficiency by conducting a comparable user study.

Structure

This work is structured as follows:

Kapitel 2 – Related Work focuses on previous work related to our topic and presents their relevance and affiliation.

Kapitel 3 – Designing SurfaceSliding deals with the general concept idea behind *SurfaceSliding*, the challenges we met during development and the results of our conducted focus group.

Kapitel 4 – Phone Dialer as a Use Case covers the implementation process of a selected use case within our system.

Kapitel 5 – Use Case Evaluation focuses on the conduction and the results of our user study.

Kapitel 6 – Discussion discusses the results of our conducted user study.

Kapitel 7 – Conclusion & Future Work summarizes the thesis' work and describes the limitations and potential possibilities of *SurfaceSliding*.

2 Related Work

In this chapter we present the status of previous works which are either related to our topic or help at getting insights concerning the interaction with nowadays mobile devices. Each work will be presented shortly and afterwards analyzed on its relevance and affiliation to our work of *SurfaceSliding*. The works we chose therefore focus on different topics like phone placement during the everyday, already existing on-table interactions, motion gestures for mobile interaction in general and also their advantages and disadvantages and at last the possibility of determining the phone's position using the approach of sensor-fusion utilizing the built-in sensors of nowadays smartphones.

2.1 Phone Placement

Given the general idea of *SurfaceSliding* it is required that the user's phone is placed on a table during the interaction. Wiese et al. [WSB13] investigated the question where people keep their phones throughout their everyday and the possible reasons therefore. They originally planned to investigate the possibility of enabling phones to infer about their current location in order to create a range of new interactions. Current location in this context does not mean their explicit geographical location by using GPS, but rather their placement within a room or in a comparable extent, whether the phone is in a bag, trouser pockets or backpacks. Given this information, phones would be able to interact differently with the user based on their current placement, e.g. placement-dependent notifications or the prevention of unintentionally calling somebody while the phone is in a trouser pocket, referred to as "pocket dialing". In order to test their system's accuracy, they used the data from the built-in accelerometer and an own built prototype with a capacitive grid and a multi spectral sensor. Using those three methods they were able to distinguish between different locations like pockets, bags or out in the hand to a specific degree. For example, by knowing the fact that the phone is currently located in the bag of the user, the phone could automatically lock its screen and thus prevent pocket dialing. Beside the implementation of their system and the conduction of their test study, they also collected data from 693 participants in form of

	Walk	Drive	Home (awake)	Office
Trousers	**50%**	26%	20%	24%
Purse	14%	13%	5%	7%
Bag	8%	5%	2%	7%
Jacket	9%	4%	2%	7%
Hand	14%	3%	11%	0%
Table	1%	<1%	**52%**	**49%**
Out In Car	<1%	**42%**	1%	1%
Other	4%	7%	7%	4%
Total	1083	1002	1039	135

Table 2.1: Where participants place their phone in different activities and locations. Respondents could indicate more than one place. The most popular place is bolded per column. Source: [WSB13]

questionnaires beforehand in order to identify typical placement locations for phones in different contexts throughout the day. Those results initially acted as motivation for our work since they show that some users actually tend to place their phone on the table during their everyday. Table 2.1 shows where participants reported they put their phone in four different contexts: walking, driving, home while awake and in the office. Interestingly for our work, about half of the responses stated that the phone is put on the table when being at home or in the office. Given these numbers, we started to develop an on-table interaction in order to make effective use of the fact that the phone is often placed on the table. *SurfaceSliding* focuses on the idea of having the phone lying on the table throughout the whole interaction in order to save the effort of actually picking it up to interact with it. Additionally, people, who are already used to having their phone placed on their table, do not need to change their personal behavior but rather only learn new actions or gestures in order to extend and improve the interaction bandwidth of their phone. In contrast, people, who normally do not place their phone on the table, will may find another reason in *SurfaceSliding* to do so.

2.2 On Table Interaction

There are numerous works which deal with extending the mobile interaction bandwidth through the usage of the built-in sensors [ML14; GWP12; KYR10]. However, during this section we only describe related work in the domain of on-table interaction and the research which has been done while the device was lying on the table while interacting.

SurfaceLink [Goe+14] describes the approach of having a multi device interaction by using the table itself as a functioning communication channel. The system of Goel et al. [Goe+14] allows multiple phones to communicate with each other by performing natural gestures on a shared surface. By using a combination of different sensors, inertial and acoustic microphones, SurfaceLink is able to detect a variety of gestures performed between the phones, for example dragging a finger on the table from one phone to another. Using this method, it is possible to detect the presence of devices or their relative arrangement on the same surface. They eventually managed to classify 24 gesture varieties with an average accuracy of 90.3 %.

Zhang et al. [Zha+15] developed a system, called BeyondTouch, which enables the interaction with the phone from the outside, for example by tapping or sliding on the outer case or the surface adjacent to it. The focus of their work based on the fact that most of the interaction with a smartphone is mostly limited to the front-facing touchscreen. Using the phone's sensors, BeyondTouch offers three additional possibilities to interact with it. Two of them proposed the idea of tapping and sliding your fingers on the back of the phone with either one or two hands. The third possibility described the same interaction technique as proposed by SurfaceLink [Goe+14]. By tapping and sliding on the surface adjacent to the phone, e.g. on the table, and by using the built-in sensors the user can perform different actions on the phone. However, in contrast to SurfaceLink, BeyondTouch's system is not specifically designed and not able to communicate between multiple devices. Their proposed on-table interaction should be used if the user is currently unable to directly interact with the phone. For example, Zhang et al. [Zha+15] described a scenario where the user's hands are wet and dirty and thus prevented the touching of the screen without actually polluting it. This scenario can also be transferred to *SurfaceSliding* since our system does not require the usage of the front touchscreen. The user can instead use any other part of the hand or arm in order to slide the phone into the respective direction. For the on-table interaction, their implementation used a combination of rule-based and machine learning approaches. In order to decide whether a specific event was triggered, they looked at the values of three different sensors: Microphone, Accelerometer and Gyroscope. Given these sensor values, they extracted feature points

and put them into a pre-trained model for classification in order to decide whether the user performed a single tap or a slide on the surface. In the end, BeyondTouch scored an average 93.74 % accuracy ($SD = 4.64$ %) for over 990 input events. Tap gestures were recognized with 95.68 % accuracy and slide gestures were recognized with 91.11 %.

In summary, both works offered the possibility of interacting with the smartphone without the conventional interaction method via touching the front touchscreen. One advantage of this method is its ease of use, especially the visibility of the whole front touchscreen. Usually, if the user is holding the smartphone with one hand or two hands, the user's thumb is often blocking the view to the front touchscreen to a specific degree. By using the same approach, we are able to utilize the whole screen of the smartphone without having the limitation that the user might not be able see a part of our screen.

Inspired by SurfaceLink and BeyondTouch we also took the approach of thinking outside the box by enabling the interaction with a smartphone without utilizing its front touchscreen. Both works showed us that it is possible to use the phone's sensor data and eventually get acceptable results concerning gesture recognition. Combining this knowledge and the fact that smartphones often lay on tables during the everyday, we eventually came up with the idea of *SurfaceSliding*.

Similar to SurfaceLink, Wozniak et al. [Woz+16] also aimed to enrich the interaction among multiple devices by developing a system, called RAMPARTS, which allows "users to collaborate on a sensemaking task using multiple mobile devices which use spatial-awareness to seamlessly share and identity relevant information". RAMPARTS offers the possibility of sharing and moving data among any number of mobile devices which are arbitrary placed on a shared horizontal surface. With the help of RAMPARTS, the user's task was to solve a given riddle by highlighting and moving digital notes between multiple tablets. By using directional slide gestures on the touchscreen of a tablet, the user could then transfer one note to another device while taking the devices' spatial arrangement and alignment in account. In order to keep track of the positional alignments, they used an infrared marker based motion tracking system which worked in six degree of freedom, thus also allowing the exchange of data on vertical levels. Their results showed that using RAMPARTS significantly decreased the user's task completion time when comparing to a paper-based system. Although their system is based on the usage of external sensors and devices, such as cameras and infrared markers, it shows possible areas of application for systems which deal with the positional tracking of mobile devices on flat surfaces. Having a solution like *SurfaceSliding*, which does not relay on external cameras or markers, could eventually improve and further extend the area of application of systems like RAMPARTS.

2.3 Motion Gestures for Mobile Interaction

Ruiz et al. [RLL11] performed a study in order to find possible end user motion gestures to invoke commands on a smartphone. Given a list of different tasks, which are often performed on smartphones, they found that consensus existed among their participants on parameters of movement and on mappings of motion gestures onto commands. Ruiz et al. therefore developed a taxonomy of motion gestures for mobile interaction. In order to explore user-defined gestures, they confronted their participants with a given list of tasks which should be triggered by performing self-defined motion gestures. The list of tasks included a variety of usual smartphone actions which were all common to the 20 participants beforehand. For example, answering a call, hanging-up a call, switching to the home screen, zooming in and out or selecting the next or the previous item on a list. For each task, gestures were collected and identical ones were grouped together. The group with the largest size was then chosen to be the representative gesture for that task. For example, in order to answer a call, the majority of participants performed a motion gesture where they would place the phone to their ear. Although, their findings are mostly based on three dimensional motion gestures, some of their discoveries can be transferred to the two dimensional case, which can eventually be used for the purpose of *SurfaceSliding*. Given the list of tasks, the most interesting ones are switching to the home screen and selecting the next or the previous item on a list. In order to switch back to the home screen, most of the participants performed a simple shake gesture while they were holding the phone in one hand. In order to select the next or the previous item on a horizontal list, the majority performed a flick to the side. “A flick is defined by a quick movement in a particular direction and returning to the starting position” [RLL11].

Returning to the home screen and selecting the next or the previous item on a list are both scenarios that represent applicable use cases for *SurfaceSliding* and additionally, both gestures can also be executed while the phone is lying on the table. Obviously, before implementing or supporting these gestures in *SurfaceSliding*, we have to clarify first whether the participants would perform these motion gestures on a table at all since a table can create a number of different circumstances. Nevertheless, the work of Ruiz et al. [RLL11] gave us a fundamental insight about the participants’ thoughts when performing different tasks on a smartphone by using only motion gestures. Given this knowledge, we get a rough idea of the possible gestures which can potentially be used in *SurfaceSliding*.

2.4 Motion Gestures - Advantages/Disadvantages

In contrast to most of the available apps on the play store or the conventional interaction method via touchscreen, *SurfaceSliding* relies solely on the input of motion gestures. In order to fully understand the concept of using only motion gestures for controlling and navigating, we looked into the aspect of how motion gestures differ themselves from actions like tapping or swiping on touchscreens. These differences include the question concerning the advantages and disadvantages of motion gestures and also the question whether motion gestures are more demanding towards the user.

Negulescu et al. [Neg+12] examined the relative cognitive load of motion gestures and compared them with surface taps and gestures in two specific scenarios where the user was distracted by another task. They were particularly interested in the costs and the benefits of motion gestures as an additional input method. They described motion gestures as attractive replacement for issuing commands on a smartphone. Negulescu et al. especially stated three advantages: First, motion gestures expand the input bandwidth of modern smartphones and can either serve as modifiers for surface gestures or are mapped to specific device commands. Second, they stated the high availability of motion gestures in comparison to normal touchscreen actions. Touchscreen actions often require a specific state the smartphone must be in, e.g. a specific application must be opened in order to function properly. However, motion gestures can be configured in a way that their commands, which are bounded to the gestures, are always available to the user regardless of the current state of the smartphone. Finally, motion gestures require less visual attention when comparing since the physical location of the smartphone can be sensed via the personal perception. As mentioned, Negulescu et al. [Neg+12] examined their study using two specific scenarios. The first scenario described the user interacting with the phone while walking. The second scenario described the user being situated in an environment with low cognitive load but where visual demand is at a peak, e.g. driving while stuck in traffic. In order to simulate the second scenario, they ask the participant to interact with the phone while it is held beneath a desk. It provides a scenario where the user has to interact eyes-free with the smartphone while sitting on a chair.
During the brainstorming stage of *SurfaceSliding* and similar to the work of Negulescu et al. [Neg+12], we also thought of possible scenarios where the user could possibly make use of our implementation. One initial scenario described a situation where the user was sitting and talking to another person while the phone was lying on the table. By performing a simple sliding gesture, eyes-free, the user would then be able to execute various tasks e.g. hang up an incoming call, set an alarm for the next five minutes or send a message. We were pleased to find out that previous work was

already done at evaluating the demand for motion gestures for this specific situation. Throughout their experiment, the participant had to perform various tasks, for example tap, swipe or move, while being in one of the two scenarios. For the motion gesture task, Move, the user had to perform four selected motion gestures, which were introduced by the consensus set of motion gestures described by Ruiz et al. [RLL11]. As an additional motion gesture input, they also added the Double-Flip delimiter for motion gestures which was proposed by Ruiz and Li [RL11]. For measuring the cognitive load of motion gestures, Negulescu et al. [Neg+12] measured the reaction time the participant needed in order to perform a given task. In the end, there were no significant effects among the three techniques, tap, slide or move, in respect to the participant's reaction time. However, it must be noted, that the participants performed significantly fewer commands during the Move task when comparing to the Swipe and Tap task. Given this fact, it shows that there are often trade-offs when using another input modality: While the throughput of commands is significantly lower for motion gestures, they yet enable the usage of eyes-free commands which can always be triggered by the user regardless of the current phone state.

In conclusion, Negulescu et al. [Neg+12] expressed the opinion that "motion gestures represent a viable input alternative for situations where the user is distracted by another task and may require an eyes-free input method". Based on this, we are confident that *SurfaceSliding* will further extend the already given mobile interaction bandwidth and provide suitable possibilities for interactions that are characterized by being always-available and having the option for eyes-free input.

While motion gestures provide promising and innovative interaction methods for smartphones, designers should also take their social acceptability in account while using them in public places. Previous work by Rico and Brewster [RB10] showed the results of a study concerning the social acceptability of motion gestures performed on mobile devices with respect to the location and the surrounding audience. It showed that the user's willingness to perform certain motion gestures was significantly influenced by these two factors. Their work also provides a guideline containing design recommendations in order to develop motion gestures with a high social acceptability rate. First, participants felt more comfortable performing motion gestures which were more subtle and unobtrusive to their surroundings. Second, gestures that are similar to existing technologies, such as tapping and rhythm, were also considered to be more social acceptable since people are already known to these interfaces and do not see them as strange or weird. Third, they also recommend designing motion gestures which look or feel similar to everyday actions. Gestures, like foot tapping or shaking, were described as natural movements which people would normally do anyway while listening to music or shaking a juice bottle.

Based on the work by Rico and Brewster [RB10] we are sure that all motion gestures, which will be performed during the usage of *SurfaceSliding*, should be considered as socially accepted and deliver a comfortable way of usage since our interaction technique already fulfills two of the stated recommendations. First, *SurfaceSliding* offers the user the possibility to unobtrusively interact with the mobile device. For example, given our scenario mentioned earlier, the user is able to react to an incoming call while the interlocutor is unaware of the user's action. Second, the actual sliding gestures which are performed by the user can also be found in already existing technologies, e.g. the usage of an ordinary computer mouse. In both cases, the user is grabbing the device and sliding it over a given surface in order to interact with it. Since most people are already familiar with the concept of a computer mouse, we think that future users will quickly adapt to the interaction method of *SurfaceSliding*. In summary, we are certain that *SurfaceSliding* will not encounter any problems concerning the social acceptability. We plan on validating this hypothesis in the upcoming focus group and study.

2.5 Positioning Using Sensor Fusion

There are past works which deal with indoor positioning of mobile devices using the built-in sensors of nowadays smartphones [Li+13; WCV14].
Shala and Rodriguez [SR11] examined the accuracy of Android devices using the approach of sensor fusion for indoor positioning. Their work is dealing with estimating the phone's location inside a building or a place where the phone is unable to retrieve and use the GPS signal. After evaluating the individual sensors on their usefulness they decided to use different sensors simultaneously and combine them to raise the overall accuracy. Using the approach of sensor fusion, consisting of accelerometer, magnetometer, gyroscope and wireless adapter, they were able to pinpoint the phone's location with a average deviation between the estimated and real position of less than two meters. In order to calculate the momentarily position of the phone they integrated the accelerometer's data twice and combined them with the received signal strength of the surrounding Wi-Fi access points given by the wireless adapter. As a result it is impossible to recreate comparable results in an environment where no Wi-Fi access points are reachable at all. However, Shala and Rodriguez [SR11] showed that it is indeed possible to utilize the built-in sensors of Android devices in order to determine its momentarily position to a certain degree.

Inspired by their work, we also intend to take the various built-in sensors of an Android device and maximize their performance through sensor fusion. Nevertheless, it is worth

noting, that *SurfaceSliding* operates in a smaller environment, such as tables or shelves, and therefore requires a system with a higher accuracy and a lesser deviation.

2.6 Summary & Discussion

Wiese et al. [WSB13] started to investigate the question where people keep their phones throughout their everyday and the reasons therefore. About half of the responses stated that people put their phone on the table while being at home or at the office. Motivated by their work, we quickly had the idea of developing *SurfaceSliding*, an on-table interaction, in order to make effective use of the fact that the phone is often placed on the table. Literature has shown that work on developing on-table interactions for mobile devices has already be done. SurfaceLink [Goe+14], a system allowing the communication among multiple mobile devices by performing gestures on a commonly shared surface, and BeyondTouch [Zha+15], a system extending the interaction bandwidth of mobile devices by using built-in sensors to detect multiple gestures performed on the outside of the phone, are two examples for showing the possibility of having an implemented on-table interaction by using the sensors given from nowadays smartphones. Furthermore, we discussed the work of Wozniak et al. [Woz+16], RAMPARTS, which utilized an infrared marker based motion tracking system in order to keep track of the positioning and alignment of multiple devices on a horizontal surface. Given the general idea of *SurfaceSliding*, we also had to determine how the user has to move or slide the phone on the table in order to execute a specific task. Ruiz et al. [RLL11] explored this topic by performing studies in order to find possible end user motion gestures to invoke various commands on a mobile device. Their work gave us a fundamental insight about the user's thoughts when performing different tasks on a mobile device by using only motion gestures. In addition, Negulescu et al. [Neg+12] examined the cognitive demand of motion gestures on mobile devices and compared them with conventional surface taps and gestures. They showed that the usage of motion gestures often imply having certain trade-offs. While the throughput of commands was significantly lower for motion gestures, they allowed for eyes-free commands which are characterized by being always available regardless of the current phone state. Although motion gestures allow extending the interaction bandwidth of mobile devices, it is also important to look at their social acceptability. Rico and Brewster [RB10] advised to use motion gestures which are either subtle and unobtrusive to the audience or similar to gestures which are already in use in existing technologies. As mentioned, *SurfaceSliding* will make use of the built-in sensors of mobile devices in order to estimate the traveled distance through performing motion gestures on a flat surface. Shala and Rodriguez [SR11]

examined the accuracy of Android devices using the approach of sensor-fusion for indoor positioning. Using the accelerometer, magnetometer, gyroscope and wireless adapter they were able to pinpoint the phone's location with an average deviation of less than two meters. *SurfaceSliding* will also use the approach through sensor-fusion in order to determine the phone's location on the table, whereby it is worth noting that *SurfaceSliding* requires a system with a higher accuracy and a lesser deviation.

3 Designing SurfaceSliding

In this chapter we cover the design, the implementation and the limitations of our first prototype of *SurfaceSliding*. We particularly focus on the issues we ran into while developing and how we eventually solve them. Afterwards, we describe the focus group we conducted with our working prototype for investigating possible and applicable use cases for *SurfaceSliding*. We conclude this chapter with the results of the focus group.

3.1 Concept

Figure 3.1 shows the conceptional idea behind *SurfaceSliding*. By sliding the phone on a flat surface into different directions and by using the various features of nowadays smartphones, the user should be able to perform different input actions. The system should be able to detect the direction the phone was sliding to and also determine the exact position on the table, thus the distance the phone covered while sliding. The user would then be able to execute different actions on the phone by performing miscellaneous sliding movements. These sliding movements could be either simplistic, e.g. sliding the phone from left to right, or more complex like sliding the phone in a specific pattern, e.g. “drawing” an imaginary square on the table as illustrated in Figure 3.2.

3.2 Concept Realization

As mentioned in Section 2.6, our goal is to implement a system which is capable of determining the phone’s location while the user is sliding it into different directions on a flat surface. The phone should automatically track the movements performed by the user and recognize various patterns for interpreting the user’s input actions. Since *SurfaceSliding* is designed to be utilized within the everyday, we will only make use

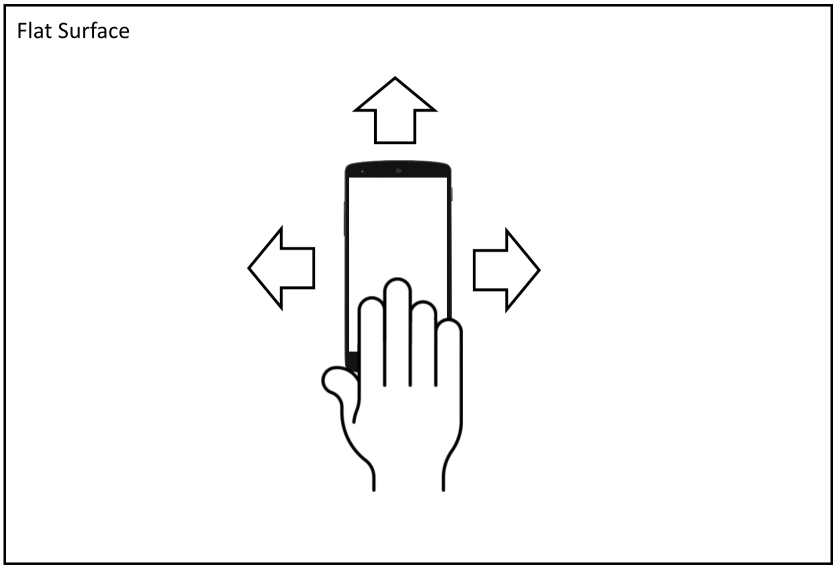


Figure 3.1: Drawing showing the conceptional idea behind *SurfaceSliding*. The black rectangle represents a flat surface, for example a table.[1]

of the functionalities our smartphone is featuring, e.g. data from the built-in sensors, and forgo the use of external sensors or other devices.

3.2.1 Apparatus

As apparatus we used a Google Nexus 5 running on CyanogenMod Version 12.1 which is based on Android 5.1.1. The Nexus 5 features a 5 inch display with a resolution of 1080×1920 pixels. It has a size of $137.84 \times 69.17 \times 8.59$ mm and weights 130 g. In addition, it also features a front-facing camera with 1.3 megapixels and also a high number of various built-in sensors, such as an accelerometer, a gyroscope or a proximity sensor. For development, we used Android Studio 1.4.1 and set the minimum required Android Version for our application to 4.0.3 (API Level 15).

[1] Source of hand image: `http://www.clker.com/clipart-hand-20.html` Released under creative commons CC0 (last access 2016-06-20)

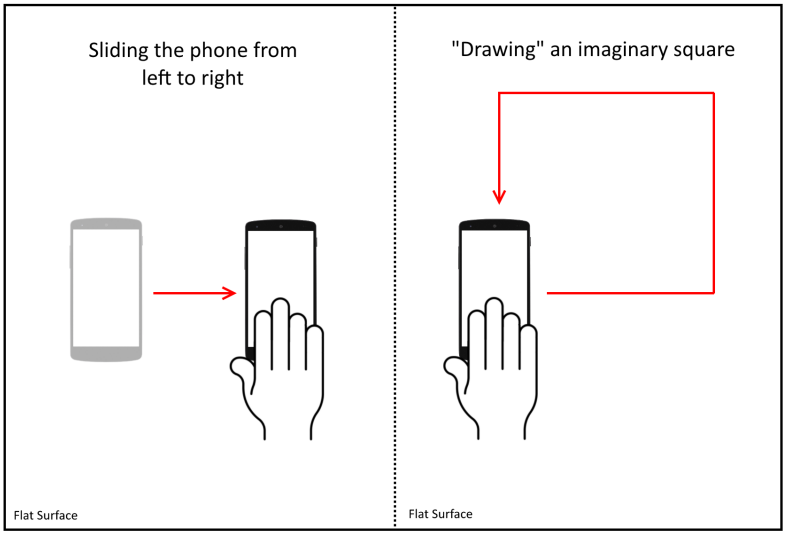


Figure 3.2: Two sliding movements illustrated using arrows as sliding directions. The black rectangle represents a flat surface, for example a table.

3.2.2 Sensor Approach

Our first approach for implementing a working prototype was solely based on the data given by the inertial sensors of our Nexus 5. Since *SurfaceSliding* mostly operates in ranges less than a half meter and therefore requires a high accuracy concerning the momentarily positioning we were unable to use the phone's location services, e.g. GPS. Given the inertial sensors of the Nexus 5, we implemented a system which was mostly based on the work of Seifert and Camacho [SC07]. In theory, by using the data of the phone's inertial accelerometer and the use of mathematical integration we are able to calculate the change of positioning while sliding. Using the fact that acceleration describes the rate of change of the velocity, it is possible to calculate the momentarily velocity by integrating the acceleration data of the given object. Of course this is only possible if all initial conditions are zero. This idea can then be also applied to the dependency between velocity and position. In other words, the momentarily position of an object can be obtained by double integrating the object's acceleration data (see Equation 3.1).

$$s = \int (v)\,\mathrm{d}t = \iint (a)\,dt\,dt \tag{3.1}$$

s position
v velocity
a acceleration

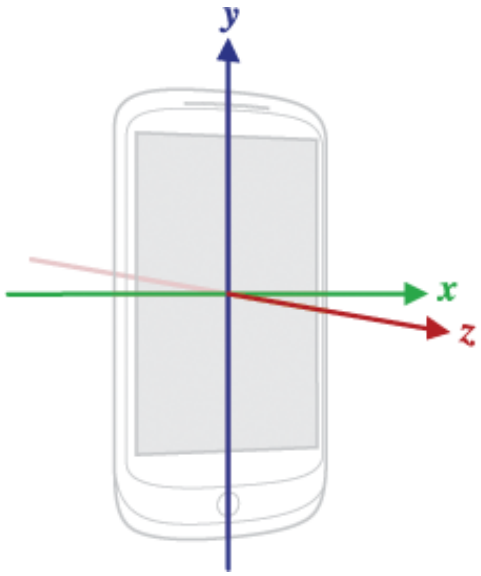


Figure 3.3: Coordinate system which is used by the Android sensor system.[2]

With regard to *SurfaceSliding* this means theoretically spoken that we only have to read the accelerometer sensor data and double integrate them to determine the phone's location on the table. The Nexus 5 provides two kinds of sensor for monitoring the momentarily acceleration of the phone: TYPE_ACCELEROMETER and TYPE_LINEAR_ACCELERATION. The difference between those two sensors lies in the way how they are implemented into the phone. The TYPE_ACCELEROMETER sensor is hardware-based, thus returning always raw accelerometer data, whereas the TYPE_LINEAR_ACCELERATION sensor uses a combination of different hardware-based sensors, e.g. gyroscope or magnetometer, in order to derive its data. Both sensors return a three-dimensional vector representing acceleration along each device axis (see Figure 3.3). However, in contrast to the TYPE_ACCELEROMETER sensor the TYPE_LINEAR_ACCELERATION sensor automatically subtracts the gravity part from the resulting vector. Figure 3.3 depicts that we have to consider the acceleration along the x- and y-axis since the phone is lying on the table throughout the whole interaction. Consequently, we used the TYPE_LINEAR_ACCELERATION sensor within our implementation since the influence of the gravity is negligibly small during the sliding movement. The units of measure for the three-dimensional vector are m/s^2.

[2]Image source: https://developer.android.com/guide/topics/sensors/sensors_overview.html (last access 2016-06-20)

Sensor Sampling Rate	Delay in ms
SENSOR_DELAY_NORMAL	200
SENSOR_DELAY_UI	60
SENSOR_DELAY_GAME	20
SENSOR_DELAY_FASTEST	0

Table 3.1: Table containing the default sampling rates for sensor callbacks within Android.[3]

The Android sensor system is using callbacks in order to send the sensor values to the actual application. The interval, at which the sensor values are sent, is controlled by the sampling rate. Table 3.1 shows the default sampling rates which can be specified during the initialization of each Android sensor. However, it should be noted that Android utilizes these sampling rates only as rough reference points. Without alternations, it is not guaranteed to receive sensor updates at regular time steps. Our application utilized the SENSOR_DELAY_FASTEST sampling since we aimed for the highest possible accuracy and therefore required as much sensor updates as possible. It is worth noting that a faster sampling rate imposes a higher load on the processor and therefore drains more battery power. Nevertheless, we used the fastest sampling rate since we intended to determine the boundaries in respect to accuracy of our implementation. In practice, we received new acceleration values after every $3 - 10$ms.

Before calculating the momentarily position of the phone, we needed to preprocess the linear acceleration data given by the sensor. First we started to calculate the real share of acceleration along each axis in relation to the phone's rotation. For example, starting from a position where the phone is lying on its back and the bottom of the phone is facing towards us we would rotate it by turning it by 90 degrees counter-clockwise. If we now slide the phone to the right, then all the acceleration gets measured along the negative y-axis. The phone would think that the user performed a downwards sliding movement even though the movement occurred along the original positive x-axis. To prevent this, we have to keep track of the phone's rotation and calculate the real acceleration along each axis based on the world coordinate system (see Figure 3.4). Therefore we used the TYPE_GAME_ROTATION_Vector sensor which is capable of defining the phone's rotation along all three axis, hence yaw, pitch and roll, in relation

[3]Source: https://developer.android.com/guide/topics/sensors/sensors_overview.html (last access 2016-06-20)

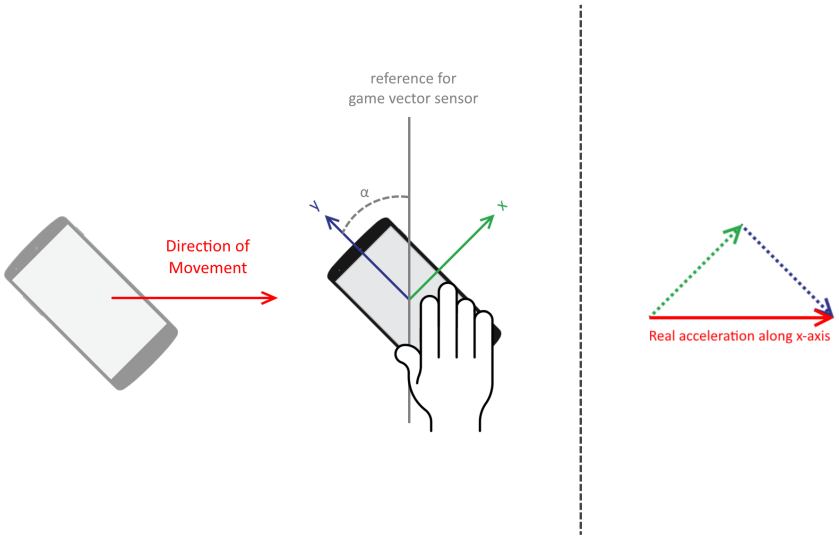


Figure 3.4: Illustration showing the real acceleration along the x-axis when sliding a tilted phone.

to a given reference point. In contrast to a normal rotation sensor, it is independent from any geomagnetic field. Having the degree of change along the z-axis we are then able to calculate the real acceleration along each axis by using simple trigonometric formulas. Since Android sensors are known to be prone to noise effects[4] it is advisable to implement a simple low-pass filter to reduce the overall noise reduction. Therefore, we implemented a low-pass filter using an alpha value of 0.1[5].

As already stated, the phone's position can be calculated by double-integrating the acceleration data given by the inertial sensors. After every sensor update we integrated the acceleration data by applying the trapezoidal rule in order to calculate the phone's new location on the table. The trapezoidal rule is a numerical method that approximates the value of a definite integral. Given the n-th acceleration value along one axis, we can then calculate the new integration value by using the following equation:

[4] https://developer.android.com/guide/topics/sensors/sensors_motion.html (last access 2016-06-20)

[5] https://www.built.io/blog/2013/05/applying-low-pass-filter-to-android-sensors-readings/ (last access 2016-06-20)

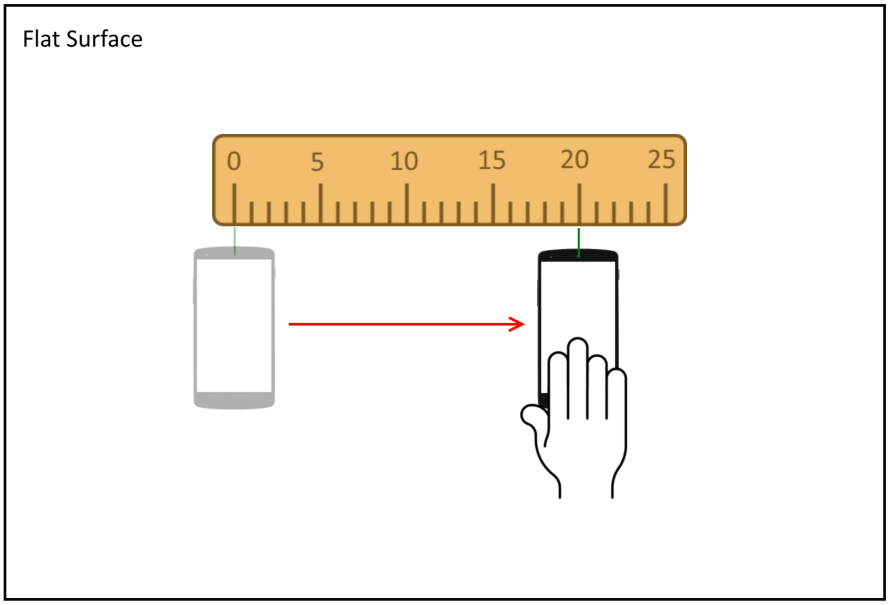


Figure 3.5: Drawing showing the setup for our first prototype test. We added a green marker to the smartphone in order to track the distance with the help of a ruler.

$$\int_0^n f(t)\,\mathrm{d}t = \int_0^{n-1} f(t)\,\mathrm{d}t + \frac{acc_{n-1} + acc_n}{2} * \Delta t \tag{3.2}$$

acc_n n-th acceleration value given by sensor along one axis

Δt time passed since (n-1)-th value acceleration was delivered

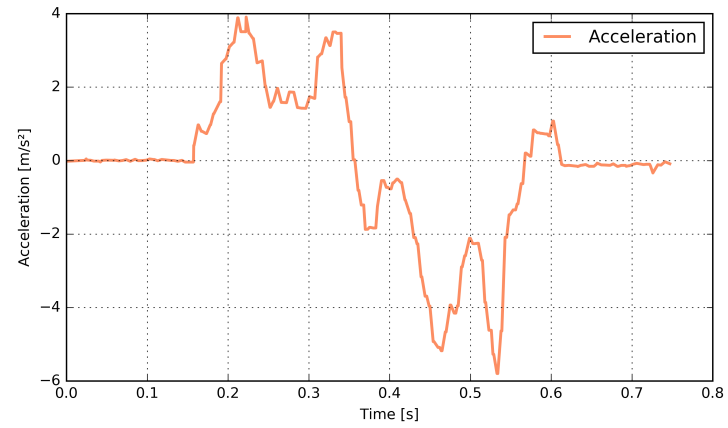
Since we have to consider the movements along two axis, this calculation has to be done for each axis separately two times after every sensor update.

For our system it was important to implement a calibration function in order to ensure an overall correctness by declaring multiple reference points: First, the game rotation sensor which was responsible for tracking the phone's rotation on the table required a reference point to which the resulting degrees are referring to. Second, the linear acceleration sensor always has an offset which needed to be removed beforehand[4]. During calibration, we read the momentarily linear acceleration values and declared

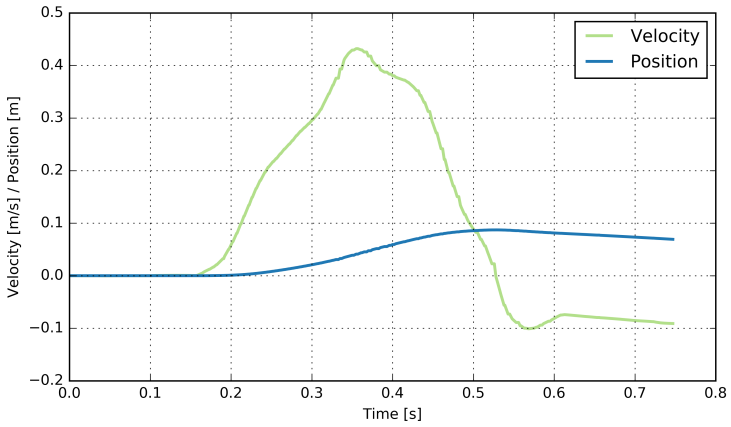
	Left-to-Right			Up-to-Down		
Sliding Distance	**M**	**SD**	**ACC**	**M**	**SD**	**ACC**
5 cm	1.26	.83	74.90	1.00	.55	80.08
10 cm	2.07	.72	79.32	2.80	1.19	71.99
15 cm	3.12	.96	79.21	3.94	.85	73.71
20 cm	6.39	.88	68.04	6.60	1.20	67.12

Table 3.2: Results of our first prototype test using only the inertial sensors. We performed ten repetitions per sliding distance. Mean (M) and standard deviation (SD) are given in cm, accuracy (ACC) in percentage.

them as offsets for future sensor readings. At last, our calculations regarding the momentarily position of the phone is also always in relation to a given starting origin point. Therefore it is also required to declare a reference point to which the changes of position are referring to. In our system, calibration was done by clicking on a button while the phone was lying still on the table.
Having a first working prototype, we investigated our prototype's accuracy concerning the calculation of the phone's location by relying solely on the inertial sensors' data. At first, we only investigated the prototype's accuracy while performing a unidirectional sliding movement, e.g. only from left to right. Besides the sliding's direction, we also varied the distance the phone is slid for, namely 5 cm, 10 cm, 15 cm and 20 cm. Figure 3.5 shows the setup of our first prototype test. We used a regular ruler and sticked a marker to the phone in order to keep track of the distance we slid the phone for. The phone was slid 10 times for each distance and for each direction respectively. We calibrated the system before every slide movement.
Table 3.2 shows the results of our first prototype test. Unfortunately, looking at the accuracy values, it is clear, that our first prototype did not fulfill our expectations at all. One reason for this observation can be partly blamed on the performance of the Nexus 5's inertial sensor. These sensors, which can also be found in other nowadays smartphones, are mostly cheaper ones which are not designed to deliver a high accuracy in such an environment or dimension. However, the sensor's performance was only one part of the issue. Looking at the curve progression of the velocity and position data it became apparent where the main part of the issue lies.
Figure 3.6 shows an example of the curve progression of acceleration, velocity and position when performing a sliding movement from left to right with a traveled distance of ten centimeters. The curve's progression towards the end of the movement shows that the acceleration sensor of the phone is especially prone to drift effects. We only slid the phone from left to right. We did not slide it back to the left at all. But still,



(a)



(b)

Figure 3.6: Figure 3.6a shows the curve progression of the acceleration data while Figure 3.6b shows the thereof integrated velocity and position of the phone.

according to the calculated velocity and position values, the phone was slightly slid back to the left after t = 0.525s (see Figure 3.9). The phone's inertial sensor did not realize the sliding movement's stop in time and continued to deliver false acceleration data resulting in false velocity and position values. This malfunction accumulates along the whole sliding movement and caused a drift effect in the end concerning the phone's position on the table. In order to counteract the emerging drift effects we added an additional source of data which should be able to signalize whenever the phone gets slid.

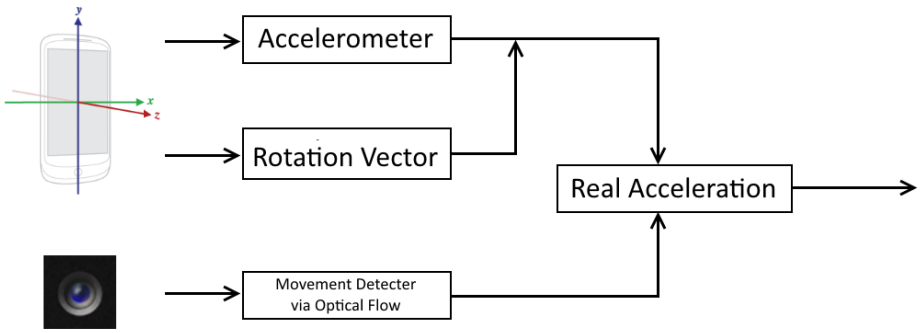


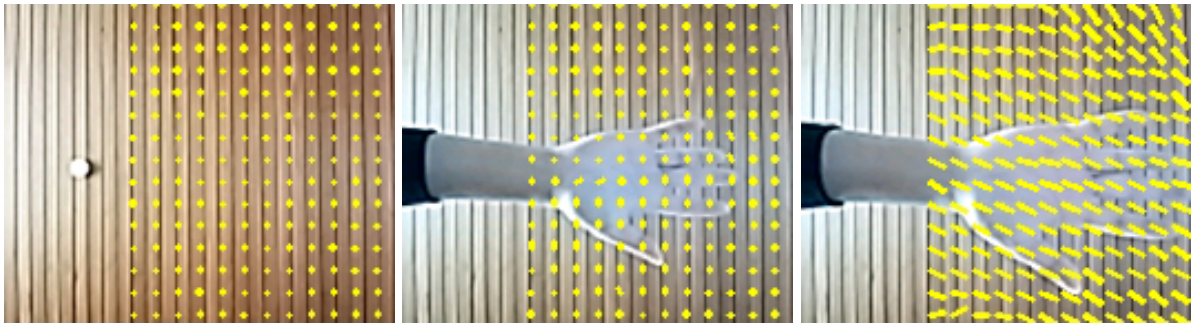
Figure 3.7: Diagram showing our sensor fusion approach utilizing the phone's front camera.

3.2.3 Sensor Fusion Approach

Our approach for counteracting the drift effects consisted of the idea of performing sensor fusion using the built-in features of the phone. Besides the inertial sensors, we decided to use the front camera of the smartphone as a sort of movement detector during the sliding movements. Since the phone is lying on its back throughout the whole interaction, it was guaranteed that the front camera's view gets not obstructed while sliding the phone on the table. Provided by the images coming from the front camera of our Nexus 5, we used an optical flow based algorithm for signalizing the phone's movement. Having a hopefully more reliable movement detector, we would then whenever no movement occurs set the acceleration data artificially to zero for signalizing the system that the phone is indeed not sliding anymore. Figure 3.7 shows the concept behind our sensor fusion approach.

In order to access the phone's front camera and to implement our optical flow based algorithm we used the OpenCV library[6] which was originally developed by Intel's research center. Being a cross-plattform library we implemented our system by using the 3.1 Version of the OpenCV Android Library.
The OpenCV Android Library offers the possibility of acquiring the images taken from the phone's camera and also provides a high amount of different algorithms and functions used within the field of real time computer vision. For implementing our movement detector we utilized OpenCV's own implemented optical flow algorithm,

[6] http://opencv.org/platforms/android.html (last access 2016-06-20)



(a) **(b)** **(c)**

Figure 3.8: Images showing the optical flow of the movement. Figure 3.8a shows no movement. Figure 3.8b shows small movement. Figure 3.8c shows high movement.

namely *calcOpticalFlowFarneback*[7]. Our implementation set the following parameters: pyr_scale = 0.5, levels = 3, winsize = 30, iterations = 3, poly_n = 5, poly_sigma = 1.1 and flags = 0. By passing two consecutive frames, the method returns a computed flow image which contains the optical flow value for each pixel separately. The images we captured had the minimum supported resolution of 176×144px in order to maximize the overall performance of our system. Each optical flow value includes the magnitude and the angle/direction of the movement. Figure 3.8 shows the optical flow for a limited number of pixels. We only used this grid of pixels in order to calculate the mean optical flow in regard to magnitude and angle/direction. As soon as the mean magnitude exceeded a specific threshold value, which we defined beforehand, our movement detector signalized the system that a movement was detected and then it now should start reading the inertial sensor values. In contrast, as long as the mean magnitude fell under the threshold number, we artificially set the acceleration values to zero in hope of getting rid of the mentioned drift effects.

We repeated the initial test of our first prototype but with the addition of utilizing the movement detector now. Unfortunately, we had to realize that the implementation of an additional movement detector, in form of a camera based algorithm, did not solve our main concern regarding the emerge of drift effects. Looking at the sensor values and the times when no movement was signalized, we quickly realized that our movement detector was too slow in comparison to the actual sensor readings (see Figure 3.9). Main reason for this low performance rate was the fact that our

[7] http://docs.opencv.org/trunk/dc/d6b/group__video__track.html#ga5d10ebbd59fe09c5f650289ec0ece5af (last access 2016-06-20)

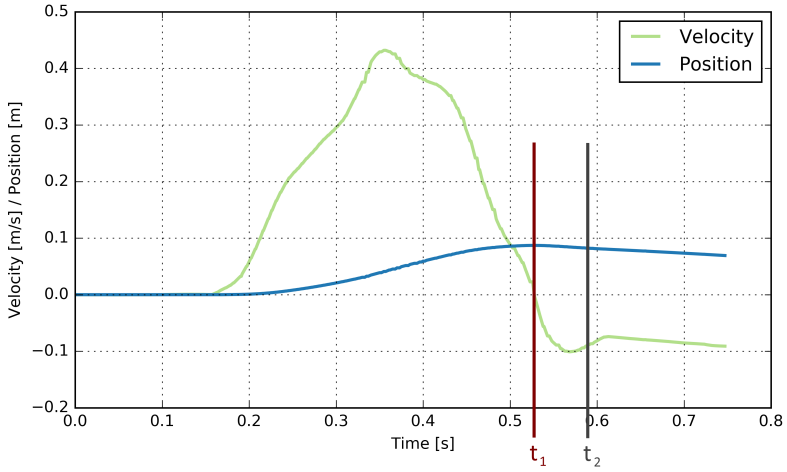


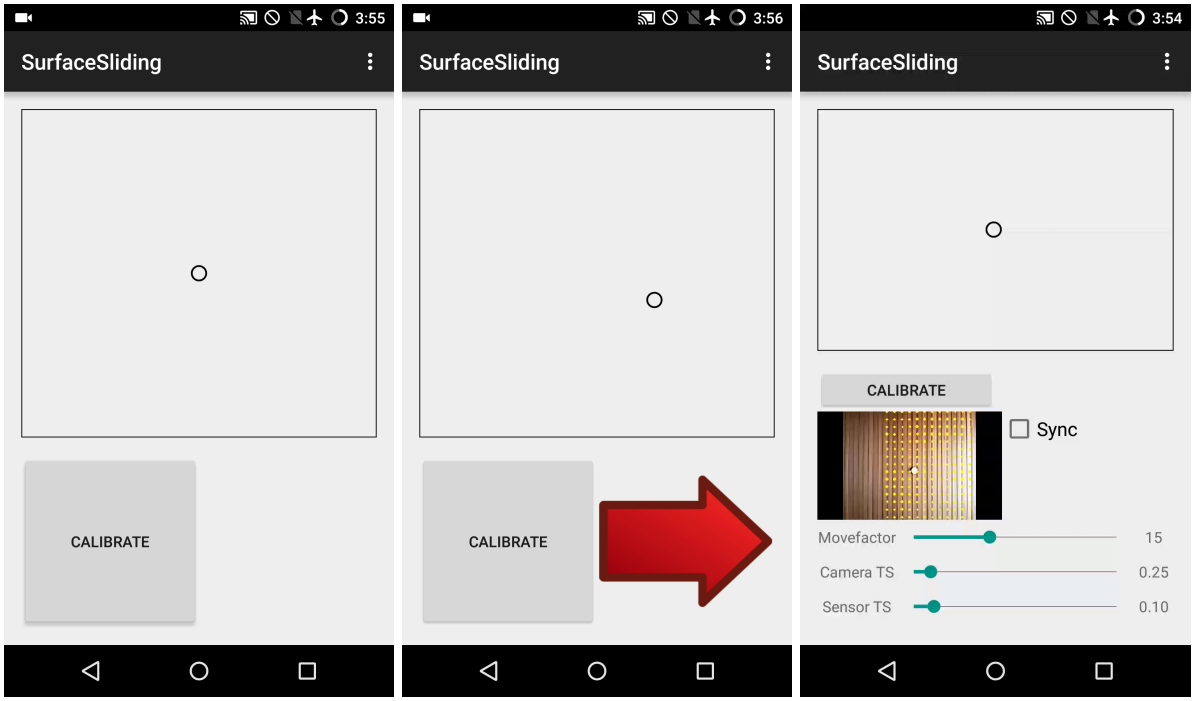
Figure 3.9: Showing the movement detector performance issue in comparison to the actual sensor readings. t_1 shows the time when the actual sliding movement has ended. t_2 shows the time when the movement detector reported that the sliding movement has ended.

movement detector algorithm was implemented as a frame based solution. This means that the signal for an occurring movement gets only updated whenever a new image was delivered by the front camera. Using OpenCV for Android, the front camera of our Nexus 5 only archived an average frame rate of 10-15 frames per seconds (fps). As a result, our movement detector signalized an occurring movement change just every 66 ms at best. In comparison to the sampling rate of the inertial sensors (see Section 3.2.2), it is clear, that such a performance rate is far too slow and unfortunately hardly usable.
Eventually, we had to drop the idea of using an additional movement detector and had to look for an alternative solution approach.

3.2.4 Camera Approach

Until now, we mainly relied on the inertial sensor's data for tracking the phone's movement on the table. As we continued to look for another solution approaches, we came to the conclusion that it would be the best if we drop the idea of using the inertial sensor's data as main source of information and relay on another data source instead. Main reason for this decision was the fact that nowadays built-in smartphone sensors do not offer the performance and especially not the accuracy we were originally looking for.

Eventually, we had the idea of switching the roles of the two main contributors during our sensor fusion approach. We would therefore use the optical flow data for positioning and the inertial sensors as an additional movement detector. Like mentioned earlier, the optical flow data of an image includes not only the average magnitude but also the angle/direction the movement was performed at. Having this information, we are able to reduce the phone's sliding movement to the optical flow data.
Needless to say, using an image based algorithm for positioning also brings both advantages and disadvantages. As noted in Section 3.2.3, the front camera of our Nexus 5 did only offer an effective frame rate of 10-15 fps. In contrast to our previous approach the frame rate did not affect our currently prototype's performance significantly since our optical flow algorithm already calculated the difference between two frames and therefore returned the right distance the phone was slid for.
However, using an image based algorithm also means that our prototype is highly prone to potential side effects occurring in the front camera view point. Our prototype would not function properly if the optical flow algorithm is unable to locate any characteristics or differences within the image since it utilizes them while processing, e.g. while being in a complete dark room the algorithm would detect no movement at all. This also means that our prototype's performance depends highly on the images the front camera is taken. In most cases, the front camera should mostly capture the room's ceiling since the phone should be horizontally placed on the table. Having a monotone ceiling without any characteristics at all, e.g. no light sources or hardly any contours, would then consequently result in bad performance and accuracy.
Furthermore, the algorithm's performance would also be influenced whenever a movement would occur within the front camera's view while the phone was lying still on the table. For example, if we waved our hand over the phone's front camera, the system would falsely assume that a movement occurred and therefore caused false behavior. Similar to our previous approach, we added an additional movement detector which utilizes the phone's accelerometer sensor for detecting occurring sliding movements in order to counteract these side effects. Whenever the acceleration values exceeded a certain threshold value, the system would then start to process the images from the front camera and calculated the overall optical flow. Unlike before, by using the inertial sensors as movement detector this time, it is guaranteed that the detection of movement occurs faster than the positioning since the sensor's sampling rate is higher than the frame rate of the phone's front camera. In addition, while sliding the phone, we noticed that our head was often captured by the phone's front camera. As a result, we alternated the algorithm to only process the upper two-thirds of the image in order to avoid any possible side effects from moving the head while sliding the phone (see Figure 3.8).



(a) (b) (c)

Figure 3.10: Screenshots of our optical flow based prototype. Figure 3.10b indicates a sliding movement to the right. Figure 3.10c shows the debugging settings.

Keeping these points in mind, we adapted the functioning of our prototype (see Figure 3.10). The little circle in the upper square illustrates the momentarily position of the phone. By sliding the phone on the table, the circle would start moving accordingly to the user's sliding movements. We were able to alternate the movement speed of the circle by defining a speed factor beforehand. On top of that, we added an imageview in the bottom-right corner showing the momentarily direction the phone is slid to, e.g. up, down, left or right (see Figure 3.10b). If there is no movement within the camera's field of view, the imageview is displaying a blank image. Furthermore, the user had the possibility to align the circle back to the center by clicking on the calibrate button. We intentionally hid the image preview of the front camera since the user's opinion could be otherwise influenced by knowing the technical aspect and functioning of our prototype. However, it is possible for the user to display it along other debug options by selecting the respective option in the options menu (see Figure 3.10c).
Overall, we were pleasantly surprised by our prototype's performance and accuracy. Having a visual indicator for showing the momentarily position of the phone is especially helpful for understanding the general concept of our sliding interaction idea.

Since our focus was more on having a working prototype showing our general concept idea rather than having a perfect accurate system, we were confident that the status of our prototype was sufficient for the upcoming task of finding applicable use cases for it.

3.3 Use Case Exploration

After implementing the core functionality of our prototype, we conducted focus groups in order to find a number of use cases for our system. Focus groups are ideal for our purposes since these provide a great insight into the user's motivation, desires and needs in a quickly and cheaply way. By having several participants together, they are more able to generate different ideas than they can come up with on their own. Furthermore, we want to ensure that the users experience the prototype at first-hand since our system can be heavily improved by how well the users respond and think about it. The design of our focus group is based on the work of Gvero et al. [Gve13].

For our focus groups we recruited eight participants through the university's mailing list of whom seven were male. The participants were aged between 23 and 28 ($M = 25.7$, $SD = 1.6$) and either worked as PhD students or were bachelor or master students. We split the eight participants into two equal sessions resulting in two focus groups with four participants respectively. Both focus groups were performed on the same day and took on average 60 minutes. For analyzing the focus groups, we audio and video recorded both sessions after obtaining the consents of all participants. All of the participants owned an Android smartphone for several years and were therefore used to the concept of handling a modern smartphone.

After a short introduction to ourselves we asked the participants to introduce themselves to the others by telling their name, age and current occupation. We also asked them about the smartphone they were currently using in order to create a small connection among the participants to raise their comfortableness level. The purpose of this introduction was to set the tone for the following discussion and to break the ice for the participants. Before conducting the actual discussion, we asked all participants whether they were all already familiar with the concept and the idea of a focus group. Since every participant took part in another focus group before, we skipped the explanation and continued with the actual discussion. To this point, we did not tell them the actual purpose of this focus group since the participants should concentrate on the current questions rather the actual end goal which could eventually influence their answer. Our questions were chosen in a way, that we slowly approached the topic concerning possible use cases for our system. Using this approach, the participants

Figure 3.11: Picture showing the conduction of one of our focus group session.

would gradually explore the actual end goal which should help to keep them more focused and motivated for the momentarily discussion.

As an introductory question we asked the participants to list a number of places where they keep their phone at during their daily routine. We handed little cards out and told them to write down about three or four places they could think of. This question served as initial ice breaker and should motivate all participants to involve themselves with the following discussion. After the participants finished writing, we collected all cards and sorted them into groups based on their location. We read every single group out aloud and asked the participants to tell possible reasons why they keep their phone at that place. Of the eight participants, every one of them wrote down that they keep their phone in one of their trouser pocket. After asking why, they stated reasons like easy transportation, quick access and the possibility of feeling the vibration of the phone if a notification or call comes in. During the focus group the majority of participants continually stated how important it is for them to use their phone as a notification center which notifies them about things like received messages, emails or incoming calls. The idea of having a personal notification center during the day is definitely a point we considered throughout the following process. Similar to trouser pockets, all participants also listed table as place of storage. As main reason, the participants told us that they feel much more comfortable if their phone is lying out on the table

rather in their trouser pocket while they are sitting. Three of the eight participants reduced this behavior to their phone's proportion since their phones feature a rather large display which could eventually lead to a bulky feeling in their pocket while sitting. Furthermore, two participants stated that it is easier and more convenient to see whether they received a notification or not since they do not have to explicitly take their phone out of their trouser pocket. This is especially relevant if the phone is running in silent mode where the users can only see by the LED status whether they received a notification or not. Other but less mentioned locations were nightstand, backpack and shelf. We mentioned that Wiese et al. [WSB13] also investigated this matter in 2013. We told the participants that Wiese et al. were interested in the possibility of creating new interactions for phones based on their current location. Before sharing the actual result of the study we asked the participants to name us the most common place they think they put their phone at during the last 24hrs. Only two participants reported that tables were the most common location. In contrast, five participants picked front pocket as their most common spot. The last participant shared with us that he already heard about the paper and therefore decided to abstain from voting since he was also interested in the answers of his fellow participants and did not want to influence their decision. Although the number of our participants were quite low in comparison to the survey of Wiese et al. [WSB13] with 693 participants, we also share the fact that table and front pocket were the most popular answers on this question.

We then proceeded by sharing the results of Wiese et al. [WSB13] which show the distribution of different locations in respect of the currently performed activity: walking, driving, home (while awake) and in the office (see Table 2.1). We asked the participants whether they agree or disagree with the overall distribution. If someone disagreed we demanded a short explanation why they thought so. Participant P1 stated that he rather hold his phone in his hands than in his front pocket while walking since his phone was too big to fit in. After a short discussion involving the different numbers, we directed the attention to the fact that about half of the responses, while being home (while awake) or at the office, indicated that the phone is lying out on the table. The majority of our participants stated that they were able to identify themselves with it. As a result, we led the focus to the idea that being able to interact with the phone while it is lying out on the table could eventually be beneficial since this would actually save the effort of picking it up. Based on the responses of the participants we were sure that everyone agreed with our suggestion. Henceforth we laid the focus of the discussion on the actual on-table interaction.

As an introduction to this topic, we asked the participants what would come up their mind when someone mentions the term "on-table interaction". Additionally, the participants should also talk about how an on-table interaction could look like

or whether they already know about some interaction techniques beforehand. One of the first things which popped up during both sessions focused on the usage of the front camera in order to detect simple swipe gestures with your hand which are performed above the phone. By identifying these movements the user could perform common tasks like scrolling down a list or manually turning a page within a document. Participant P3 stated the idea of actually flipping their phone upside down to perform certain tasks like rejecting incoming calls or turning off the alarm. The three other participants liked the idea since the gesture itself can be seen as metaphor of actually rejecting an incoming event. Furthermore, such gestures do not demand great cognitive perception or fine motor skills from the user. Another approach involves the usage of the table itself. Participant P4 proposed the idea of actually knocking with the hand on the table whereby the phone would eventually register the resulting vibrations. The user could vary the interaction by using different rhythms patterns or magnitudes. At last, participant P6 mentioned that the phone could function as a computer mouse replacement by moving the phone in the two dimensional space which is given by the table. The participant's approach basically described the general idea of *SurfaceSliding*.
After collecting all ideas of the participants we proceeded by demonstrating our prototype to them. Before handing the phone over we gave them a brief overview about the general usage of the prototype, e.g. lay the phone on the table, press calibrate and eventually move the phone. While testing the prototype, the participants made some remarks: Participant P5 asked about the possibility of rotating the phone anti-/ and clockwise. Besides, participants P4 and P6 noted that the physical camera could eventually suffer from the overall interaction since most of the phones, including our prototype, lay on the camera when placed on their backs. Many participants tried to quickly push the phone in one direction in order to check whether the prototype would be able to register such movements or not. Unfortunately our prototype was not able to do so. Back in Section 2.4 we mentioned the problematic concerning the possible social unacceptability of *SurfaceSliding* due to its sliding movements. However, all of our participants stated that they never felt uncomfortable while using *SurfaceSliding*.
After the participants became familiar with our prototype we asked them whether they could think of possible use cases for it. Overall, the participants' ideas can be divided into three categories: Multi-Device Interaction, Table Involvement, Smartphone Actions Simulation.
Two ideas were focused on the interaction between multiple devices, especially the combination of a phone and a computer. Like already proposed in a earlier step of the focus group, participant P6 could imagine to use the prototype as a computer mouse replacement. A second idea revolved around the usage of the phone as an additional input device in order to remotely control or trigger specific events on a second device like the personal computer. The user would therefore have to predefine

sliding patterns, like moving your phone quickly up and down, which would be linked to certain actions on the computer, like skipping the current song or changing the volume level.
The second category deals with use cases which try to include the table as a physical unit within the interaction. One suggestion was to divide the table surface into multiple areas which would represent different functionalities. Depending on the current location of the phone, the phone would act in different ways. For example, the user could define the top-right area of the table as a special area. Whenever the phone would be pushed into this area, the phone would automatically switch into silent mode or perform different actions. Another idea focused on the concept of automatically locking and unlocking the phone by either pushing or dragging. Every time the user drags the phone to him, it would automatically unlock itself and turn the screen on. The benefit of this concept would be the quick access and insight whether the user received any notification without actually pressing any button or performing certain touch events to unlock his phone. By moving or pushing the phone in any direction away from the user, the phone would then automatically lock itself again. Consequently, in order to implement this functionality it is required that the phone knows where the user is currently sitting at. Participant P2 and P7 imagined themselves at using the prototype to explore photos or maps by moving the phone on the table. Like navigating in a photo or map by using a finger, the user would then slide the phone into the direction the user intends to explore. Additionally, by using the sliding technique instead, the screen of the phone would not be obscured by the user's finger and enable a greater field of vision to the user.
Besides these two categories, most of the ideas from the participants focused on the executions of different phone actions which would be instead triggered by using the sliding technique of the prototype. Typical phone actions include the acceptance or rejection of an incoming phone call, the adjusting of the phone's volume level, scrolling through a given list or the dismissal of notifications. For example, if the phone is lying on the table and a call is incoming, the user could have the option to slide the phone into a specific direction in order to appropriately react to the call. A slide to the right could represent the action of accepting the call, whereas a slide to the left would mean that the user is rejecting the call and the phone would automatically send the caller a message that the user is at the moment occupied and is unable to accept the call. Based on this example, the different directions, the phone is slid to, could be bound to different actions in order to appropriately react to incoming events. Aside from reacting to events like phone calls, participant P2 stated the usage of the prototype for scrolling through a given list or page, like a tweet history of a Twitter user or the timeline within the Facebook application. The user would slide the phone either up or down in order to scroll in the respective direction. Since the free space on most tables is limited to a specific degree, it would be impractical to directly translate the sliding

movement into the actual scrolling. Instead, we need a solution which focuses rather on relative than absolute movement. The participant proposed the idea of declaring an origin point of the phone and using it to relatively decide in which direction and magnitude the user intends to scroll to. By sliding the phone away from the origin point, the phone would calculate its momentarily distance to the origin point and then decide in which direction and how fast the scrolling will take place in. Given a opened page in the phone's Internet browser, by slowly sliding the phone down from the origin point, the phone would automatically try to scroll down to the bottom of the page. The more the distance to the origin point, the faster the scrolling is executed. In order to stop the scrolling, the user would have to slide the phone back to its origin point. By using this approach, we only need limited space in order to perform actions which normally require more space. Besides scrolling, this method can also be transferred to all applications which allow to raise or lower a certain feature, e.g. in- and decreasing the phone's volume level.
The remaining ideas from the participants were proposals which either share already mentioned concepts or went beyond the limits of the original idea of *SurfaceSliding*. For example, participant P8 suggested to include the third dimension in the interaction by actually picking up the phone and holding it above the table and simulate the slide movement by dragging the phone in different directions.
As we noticed that the participants ran slowly out of possible ideas we started to direct the conversation towards the end. We recapitulated all mentioned ideas and thanked the participants for their attendance at the focus group.

3.4 Summary & Discussion

Having the idea of designing an interaction where the user is sliding the smartphone on the table in order to perform different actions, we started to implement our first prototype. Our first approach focused on only utilizing the inertial sensors of nowadays smartphones, e.g. accelerometer and rotation sensors, since we looked for an everyday solution which should not require any external sensors or devices. Inspired by the work of Seifert and Camacho [SC07], we double integrated the acceleration data from the sensors in hope of getting the momentarily position of the phone. Unfortunately, we quickly realized that this approach was highly prone to emerging drift effects concerning the accuracy of the acceleration sensors. Hoping for better results, we adapted our prototype by adding an additional movement detector in form of an optical flow based algorithm using the images captured by the front camera of the smartphone. However, limited by the frame rate of the Nexus 5's front camera we were forced to drop the idea and look for an alternative approach. Switching the roles of

the inertial sensors and our optical flow based algorithm we were pleasantly surprised by the performance and accuracy of our prototype. We were confident that the status of our prototype was sufficient for our needs and therefore continued to conduct a focus group with the aim of finding possible and applicable use cases for our system. During the focus group, we were particularly interested in the participant's general smartphone usage and their opinion about controlling their smartphone through on-table interactions. After introducing them to our prototype, we asked the participants whether they could think of any possible use cases in which they could eventually benefit from using our sliding technique.

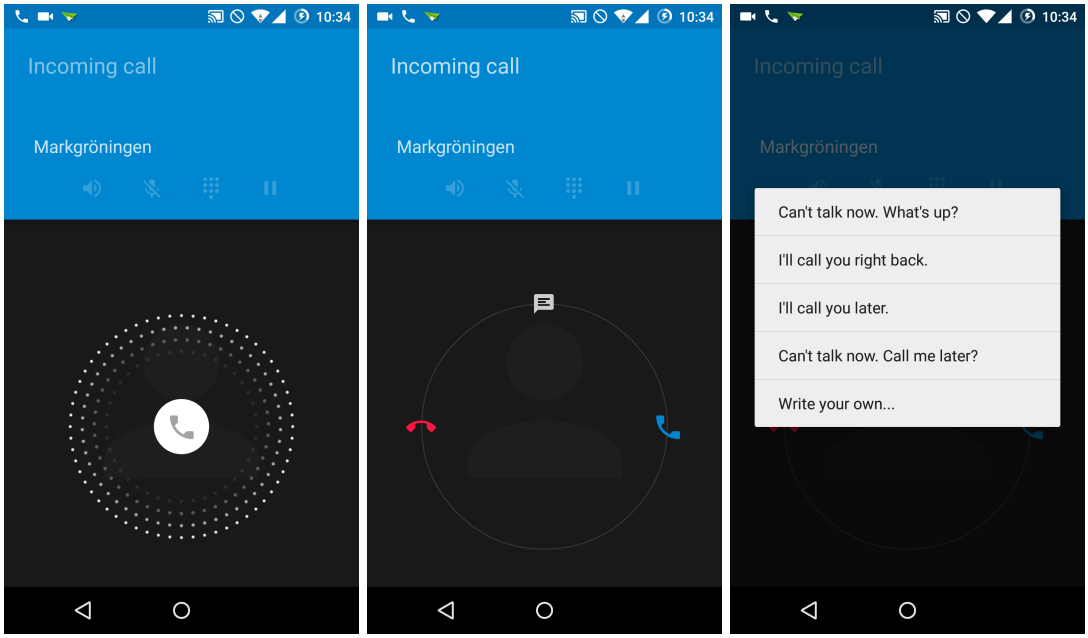
4 Phone Dialer as a Use Case

After the conduction of our focus group, we eventually picked one of the use cases proposed by our participants. In this chapter we analyze the use case's applicability concerning its usage within the upcoming user study and describe the development process of implementing the incoming call use case within our system. We mainly focus on the challenges we experienced and the decisions we eventually made in order to solve them. The chapter covers the application's logic and also its graphical user interface structure which was inspired by the official call activity used in Android Lollipop.

4.1 Incoming Call Use Case

Based on the results and the answers from our participants within the focus group, we needed to select one applicable use case which should be implemented and later on further investigated by conducting a user study. We were looking for a use case which people should encounter in their daily routine and furthermore offer the possibility of collecting objective data in order to assess and evaluate our input method. Additionally, the use case should feature the usage of at least one more other input method so that we are capable of performing a comparative user study using different input methods. Given these constraints we decided to implement the use case in which the user had to respond to an incoming call by using *SurfaceSliding*. In detail, by sliding the phone into different directions, e.g. up, down, left or right, the user is able to react to the incoming call in different ways, e.g. accepting, declining or sending a text message. Figure 4.1a shows the screen of our Nexus 5 device when a call is incoming. As a short notice, this screen gets only displayed when the phone was locked and the screen turned off beforehand. It is part of the official Android Dialer App [1] which is used on all Android devices which are running on Android Lollipop (5.0) Stock Version. The idea is to allow the user to navigate within this activity by using *SurfaceSliding*. As can

[1] https://android.googlesource.com/platform/packages/apps/Dialer/ (last access 2016-06-20)



(a) **(b)** **(c)**

Figure 4.1: Activity displayed on our Nexus 5 when a call is incoming. We removed the displayed telephone number for data security.

be seen in Figure 4.1b, a user normally has three options in order to respond to the call: Declining, accepting or selecting a predefined text message to be sent back to the caller (see Figure 4.1c). For example, if a user wants to decline the call, he performs a single drag and drop action by touching the gray icon in the middle and dragging it over the red decline icon on the left. From now on, we will call this the tapping method. Like mentioned in the focus group, in order to perform the same action using *SurfaceSliding*, the user would have to physically slide the phone to the left while it is laid down on the table. This is called the sliding method. During the upcoming study, the user's task would be to appropriately respond to an incoming call by using the two input methods mentioned above. The user would then receive a specific objective, such as responding to an incoming call by sending a SMS back containing the text "I'll call you right back". By having a specific objective for the user, we are able to collect and measure objective data concerning the success rate of the user. First, by tracking the time the user needed in order to respond to the call and second, the amount of times the user successfully performed the task, e.g. selecting the right answer option we demanded.

Based on the collected data we would then be able to objectively compare both input methods concerning the usability and effectiveness for responding to an incoming call.

In addition, the tracking of the user's task completion time allows us to get an overall insight about the average time a user needs in order to perform various actions, e.g. how long does it take to select the right message within a list by using the sliding input method. Being a new interaction method and inspired by the Keystroke-Level Model (KLM) by Card et al. [CMN80], we intended to develop a similar model consisting of the various actions the user can perform by using *SurfaceSliding*.

4.2 Concept

As mentioned in Section 4.1, the task of our participants in the upcoming user study is to respond to an incoming call by using two different input methods, namely the tapping and the sliding method. Since we intended to conduct the user study under realistic conditions we were looking for an approach which should resemble the Android's official incoming call activity for the most part. Inspired by the Android's Dialer app[2] for Lollipop (Version 5.0), we reimplemented the incoming call activity and enabled the interaction through sliding as well as through traditional tapping.

4.3 Implementation

As we started our implementation from scratch we were obligated to imitate the functioning and the design of the official Android application.
Focused on the graphical user interface at first, we screen captured our Nexus 5 while receiving a call in order to collect all required graphical resources. During implementing, we took special care to ensure that our implementation features not only icons or backgrounds but also all transitions and animations which are occurring on the official Android application. Figure 4.2 shows a comparison between the official and our implemented version of the incoming call activity. In contrast to the official version, our implementation features no functionality concerning the four icons located at the top-third of the activity, e.g. speakerphone mode or microphone muting. Apart from that, the user had the same response possibilities as within the official application, e.g. accepting, declining the call or sending a predefined text message.

After completing the graphical user interface of our application, we had to ensure that the user is able to navigate within the activity by using both input methods.

[2] https://android.googlesource.com/platform/packages/apps/Dialer/ (last access 2016-06-20)

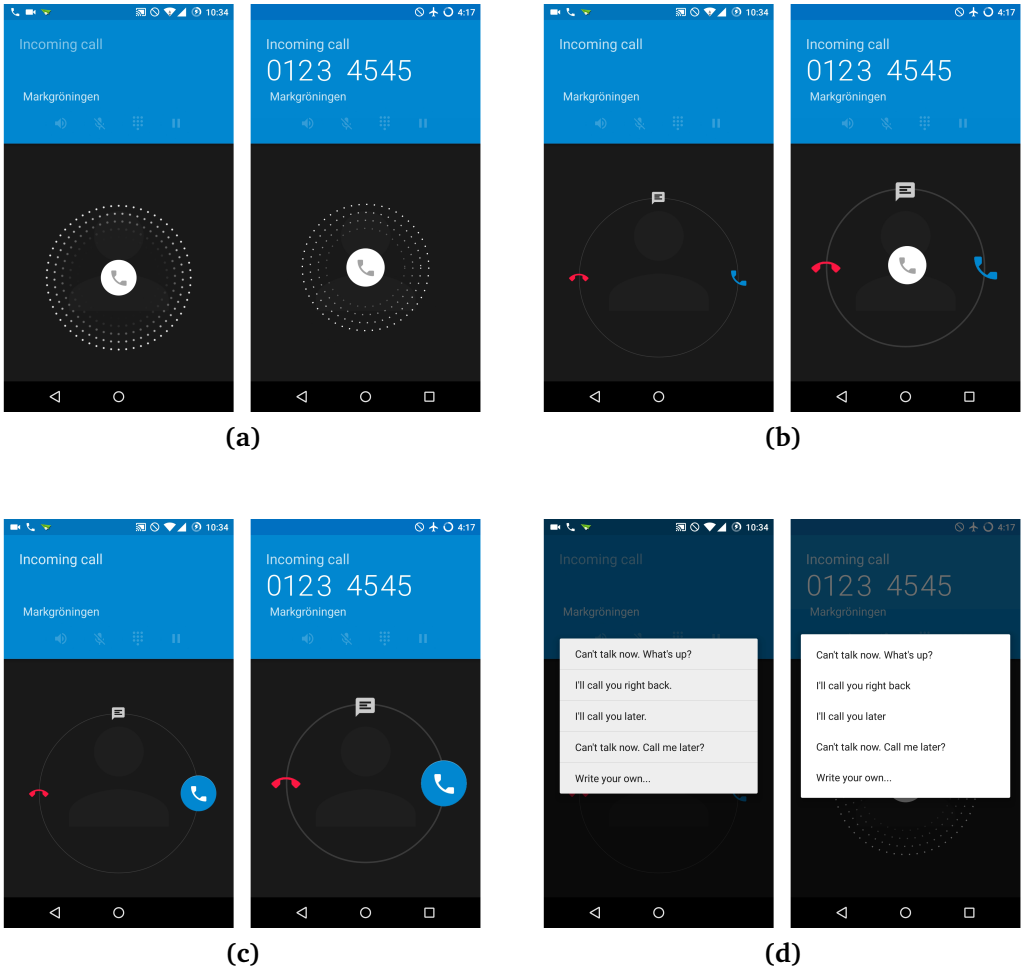


Figure 4.2: Comparison between the official (left) and our implementation (right) of the incoming call activity.

As for tapping, our implementation features the same navigation design as in the official Android Version. While receiving a call, the user starts by holding the finger over the middle gray icon and then continues to select the appropriate response option by dragging the finger over the respective icon and lifting the finger back up. The checkup of the user's selection was done by implementing simple collision detections between the icons. If the user selects the text message option, a listview will pop-up which includes five different predefined text messages which can normally be changed within the Android's settings application. Our implementation only includes the five text messages which are set by default.

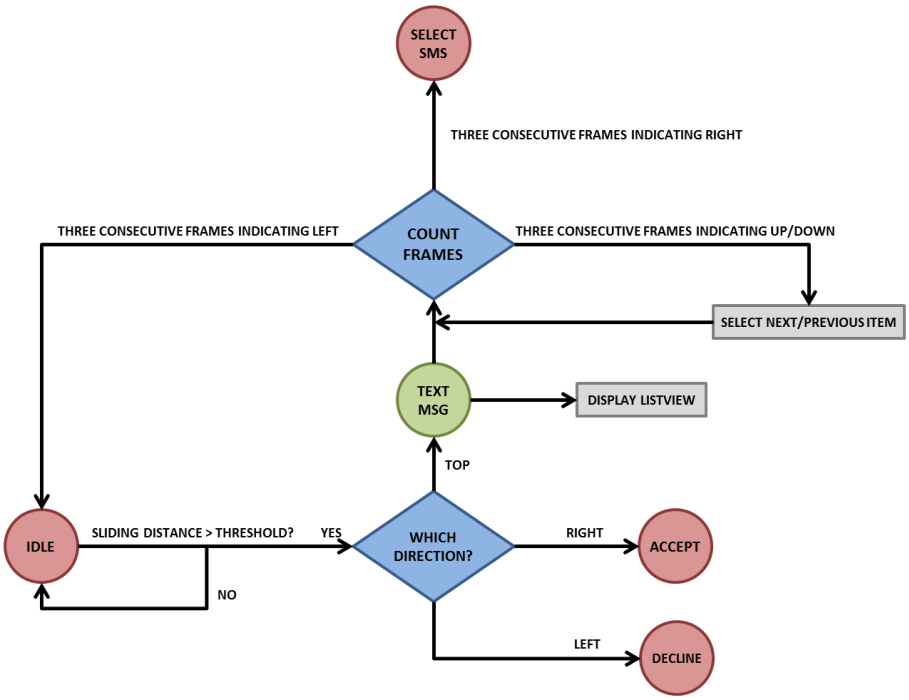


Figure 4.3: State machine used for interpreting the sliding data.

Compared to tapping, there were numerous things we had to keep in mind while implementing our sliding input method into the application. First, we had to define how our application is dealing with the continuously incoming data coming from the inertial sensors and the optical flow based algorithm. Figure 4.3 shows the general concept idea of our implementation concerning the evaluation of the sliding data. We used a state machine in order to interpret the sliding accordingly. In the beginning of each incoming call, the user's task is to select one of the three initial response options by sliding the phone into the respective direction. Being in the IDLE state, we would constantly look at the overall traveled distance of the phone. As soon as it exceeded a predefined threshold value, the application would check for the direction the phone was most slid to by comparing the traveled distances along both axes. If the user slid the phone to the left, then the application would decline the incoming call. Sliding the phone to the right would consequently accept the call. Performing an upward sliding movement, would trigger the text message option and therefore display the listview which includes the predefined text messages. The selection of the respective text message is also controlled through the sliding of the phone. While selecting the text message and in contrast to the IDLE state, the system is not checking for the traveled distance but rather relies on a frame based solution approach. We

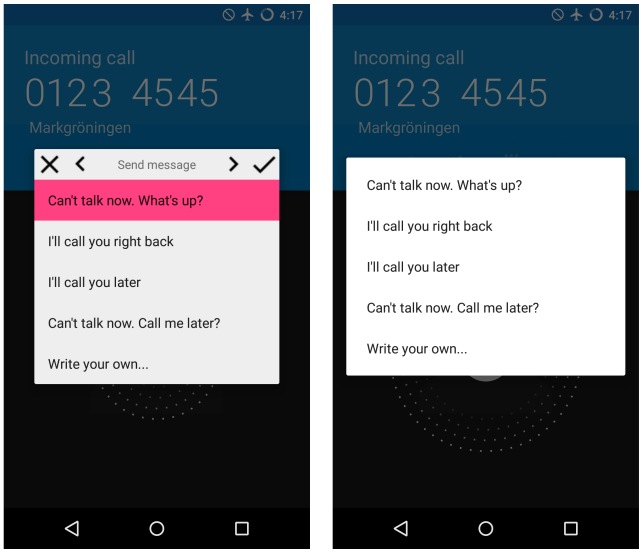


Figure 4.4: Design of the listview used in the sliding scenario (left) in comparison to the tapping scenario (right).

mentioned in Section 3.2.4 that the optical flow based algorithm does not only return the magnitude but also the angle/direction the sliding was performed at. Being in the TEXT MSG state, the system is checking frame for frame in which direction the phone is slid to. As soon as the optical flow based algorithm reports three consecutive frames with the same direction, e.g. right-right-right, the system would then perform an action which is bound to the respective direction. The three possible actions are as follows: Up/Down - Used for navigating through the listview; Right - Send the selected message to the caller; Left - Close the listview and return back to the initial selection. In order to help the user and to visualize the momentarily selection of the text message we adapted the appearance of the listview (see Figure 4.4). The first listview item is selected at default since the user already has to slide the phone upwards in order to open it. Thus we minimize the required space for the sliding interaction as the user can now utilize the space again for selecting the respective option by sliding back downwards.

4.4 Time Measurement

As stated in Section 4.1, we intend to develop a KLM-similar model which should focus on predicting how long it takes users to accomplish different tasks using the sliding input method. For example, interesting questions could be either how long

	Definition	
Time Event	**Tapping**	**Sliding**
START	Time when call is coming in	Time when call is coming in
FIRSTMOVE	Time when user is touching the gray icon for the first time	Time when optical flow based algorithm is reposting movement for the first time
ITEMHOVER	Time when user it hovering over an response option (accept, decline, text message)	not available
SELECTION	Time when system is registering the selection of one response option (accept, decline, text message)	Time when system is registering the selection of one response option (accept, decline, text message)
SMS_SELECT	Time when user is clicking on one of the listview items	Time when user is sliding the phone to the right while listview is open
SMS_UP	not available	Time when user is sliding the phone upwards while listview is open
SMS_DOWN	not available	Time when user is sliding the phone downwards while listview is open
SMS_BACK	Time when user is clicking outside of the listview	Time when user is sliding the phone to the left while listview is open

Table 4.1: Definition of every recorded time event. All time stamps were formatted in hh:mm:ss.SSS.

does it take to select the correct item within a list or how long does it take to slide the phone five centimeters to the right using *SurfaceSliding*. Due to this, it is required to measure the time at different points while the user is in process of responding to the call. Table 4.1 shows and defines all events at which we decided to track the time. Our application automatically saves all events including a timestamp to a .csv file located on the internal storage of the phone. Having these results, we are then able to define our own model concerning the required time the user needed in order to complete different tasks.

4.5 Summary & Discussion

As the participants proposed several interesting use cases, we eventually decided to pick the scenario in which the user is able to respond to an incoming call by sliding

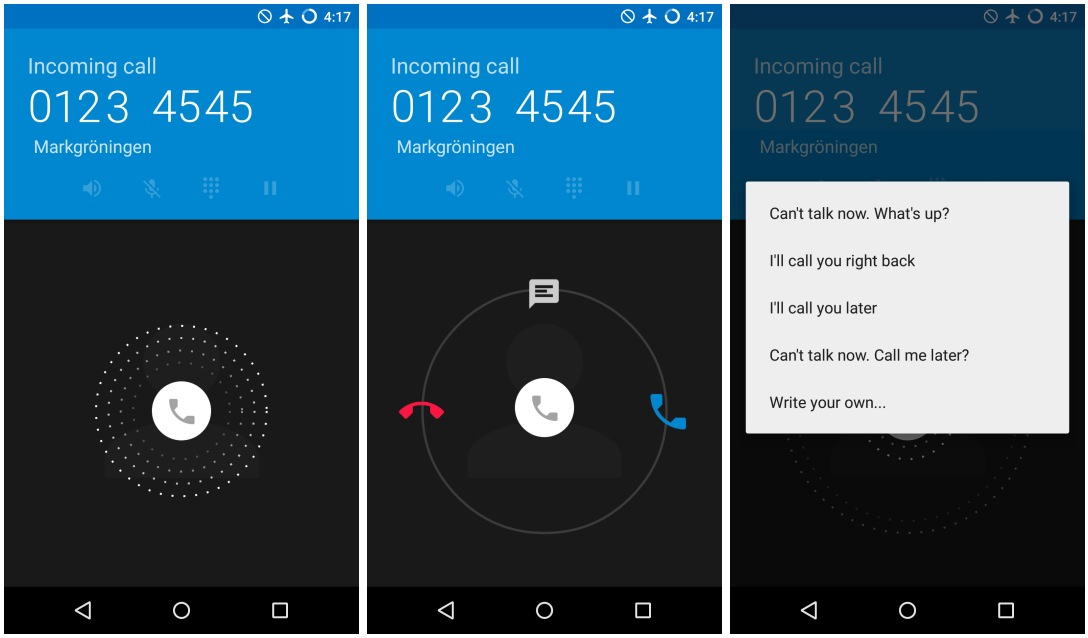
the phone into different directions. Therefore we discussed the use case's suitability concerning the upcoming user study and the implementation of the incoming call use case within our system. Inspired by and based on the Android's Dialer app for Lollipop, we reimplemented the incoming call activity and enabled the interaction through traditional tapping as well as through using *SurfaceSliding*. Having a new interaction method, we adapted and extended the application's interaction logic, e.g. by introducing the possibility of navigating through a listview by using *SurfaceSliding*. Furthermore, we added the functionality of automatically tracking the time at different points since we intend to develop a KLM-similar model later on. We continue our work by conducting a user study utilizing the incoming call use case as task in order to compare our interaction method to a traditional input method concerning its usability and efficiency.

5 Use Case Evaluation

In this section, we address the design and the procedure of our conducted study. As described in Chapter 4, the task was to response to an incoming call by using the traditional tapping as well as our implemented sliding method. During the study we measured the task completion time (TCT), the error rate (ER), the total workload (TW) of the usability determined by the raw NASA Task Load Index (TLX) [HS88] and a subjective rating score provided by the System Usability Scale (SUS) questionnaire (SUS Score) [Bro+96]. In addition, we also asked our participants about their general opinion concerning our sliding method by performing a semi-constructed interview at the end of each session.
We conducted this study in order to get a first impression of outsiders concerning our implemented system and to resolve the question whether *SurfaceSliding* could be seen as a useful interaction method. The study addresses different aspects of *SurfaceSliding*, for example its likability, usability and efficiency.

5.1 Hypothesis

Based on our experience during the implementation phase and given the two input methods we expect a noticeable difference in terms of TCT and ER. TCT and ER should be significantly lower while using the tap method since it represents an interaction method which the majority of our participants should already be familiar with. Even if the participant was not familiar with the Android OS itself, the participant should at least feel comfortable at using the finger in order to navigate within the activity. Concerning the outcome of the NASA-TLX questionnaire [HS88], we expect a higher total workload while using our sliding method. Especially the physical demand should be significantly higher since it is naturally more strenuous to physically move the phone instead of tapping and sliding on the touchscreen. Since *SurfaceSliding* aims to be an alternative interaction method in order to perform various actions on the phone we expect similar SUS scores [Bro+96] among the tapping and the sliding method.



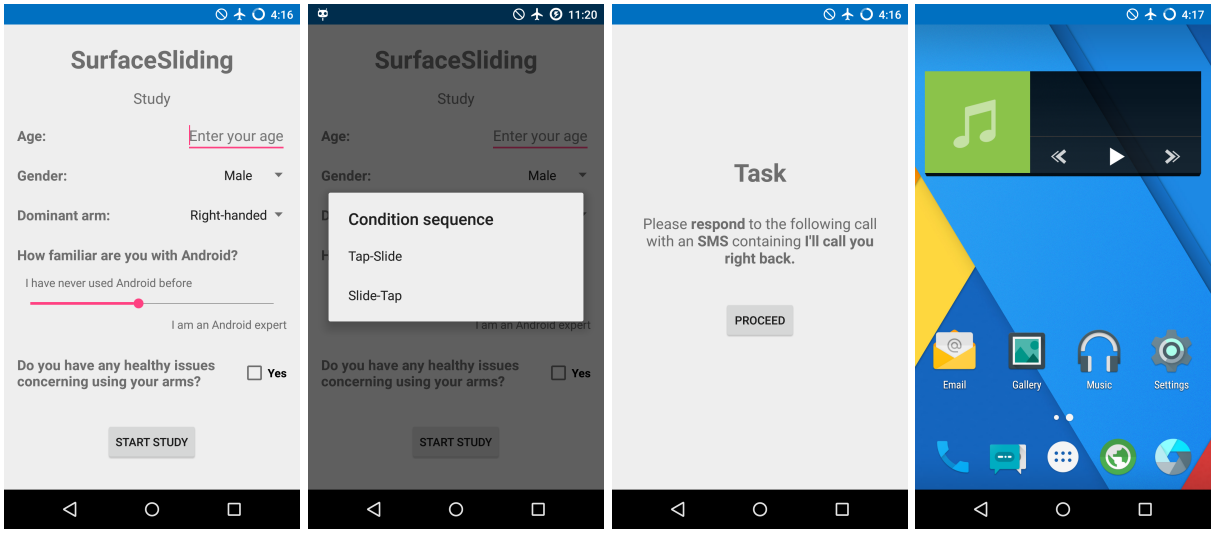
(a) (b) (c)

Figure 5.1: Incoming call activity showing all seven possible response possibilities.

5.2 Study Design

For our study we used a repeated measures design, using the input method as the independent variable and TCT, ER, TW and Usability-Score as the dependent variables. Figure 5.3 shows the overall structure of our study tool.
At the beginning of our study tool, the participants were prompted to fill in various personal information which is relevant to the later results. We asked for the participant's age, gender, dominant arm, familiarity with the Android OS and whether they had movement restrictions with their arms. For answering the question concerning the Android familiarity, we used a slider component which ranged from 0 (I have never used Android before) to 9 (I am an Android expert) using 1 step intervals.
After filling in all personal information and before the actual task began, we performed a test to determine whether the participants physically pick up the phone in order to respond to an incoming call or rather interact with it while the phone is resting on the table. For this beginning analysis we only used the tap input method since most users should already be familiar with the general concept behind it in how to use it. The structure of this beginning task resembles mostly the actual task later on but differed in certain aspects: While the participants waited for the incoming call we told them to read a book since we aimed to create a controlled environment and common

situation in which the participant should not actively wait for the call. The study tool will automatically start simulating an incoming call after 20 to 30 seconds. During this time, the study tool displays a screenshot of the phone's homescreen (see Figure 5.2d). The incoming call will signalize itself by playing the phone's default set ringtone, in our case *Orion*, and by displaying the incoming call activity (see Figure 5.1a). For the actual task later on, we removed the functionality of playing the ringtone for signalizing the participants since they will not be occupied alongside but instead lay their full focus on the phone and task. Similar to the actual task, the participant also had to repeat this task several times and was also instructed to respond to the incoming call in a specific way. In contrast, we removed the option for accepting the call during this beginning phase since accepting the call could eventually influence their decision whether they pick up the phone or not. The study observer kept track of the number of times the participant physically picked up the phone. By having those numbers, we get a general impression of how often people actually picking up their phone in order to interact with it and also are able to roughly estimate the usefulness of an input method while the phone is resting on the table. Additionally, participants were also getting to know the system in how to precisely respond to an incoming call. Especially participants who were not familiar with the Android OS could quickly adapt to the way of how Android handles incoming calls. We made sure that after every incoming call the participants would place the phone back on the table. In the following, we refer to this test as *pickup-test*.
The participants then continued to perform the actual task by using both input methods consecutively. At the end of every session we conducted a semi-constructed interview with the purpose of achieving the participant's general opinion towards the sliding method. We asked them whether they like or dislike our interaction method and how they would compare it to a more traditional input method like tapping. With the help of the semi-constructed interview we were also looking for possible areas of improvement in regard of raising the usability of our system.
The order of input methods can be selected at the beginning of each study. The study tool does not automatically select the next order; as such it is necessary to keep an external list on the number of times a specific order has been used to ensure uniform distribution. Participants may abort the study or ask questions at any given time. Any progress which has been made so far is getting logged. The study tool is able to detect participants who aborted the study.



(a) (b) (c) (d)

Figure 5.2: Activities within our study tool. Figure 5.2a shows the form for filling in all relevant personal information. Figure 5.2b shows the selection of the input methods' order. Figure 5.2c shows the task description during the pickup-test. Figure 5.2d shows the displayed screen while waiting for a call.

5.3 Task

The main task of our participants focuses primarily on responding to an incoming call by using either the traditional tap or our sliding input method. As described in Chapter 4 we are using our recreated implementation of the incoming call activity instead of the standard one which is provided by the Android Lollipop System by default. Every participant had to respond to 42 incoming calls by using both input methods consecutively. Before every incoming call, our study tool instructed the participants the way in how they should respond to it. Overall, we had seven different options in order to respond to a call. Five alternatives for every text message option, one for accepting and one for declining the call. After the participants responded to the number of incoming calls by using one input method, we let them answer the NASA-TLX as well as the SUS questionnaire in regard to their just experienced input method. Afterwards they repeated the same task with the exception that they now had to use the other input method respectively.

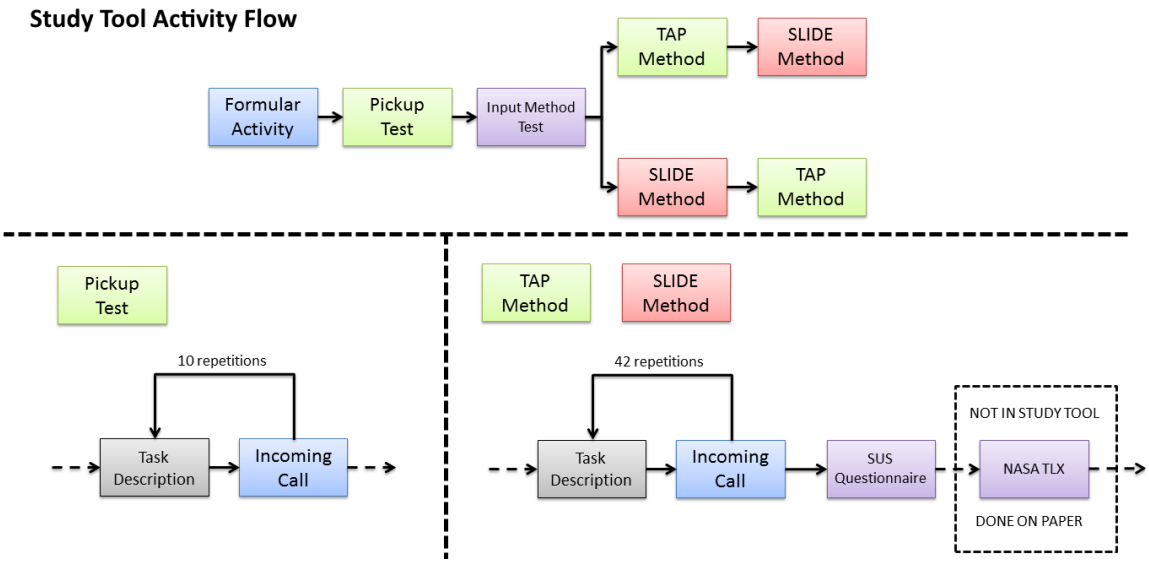


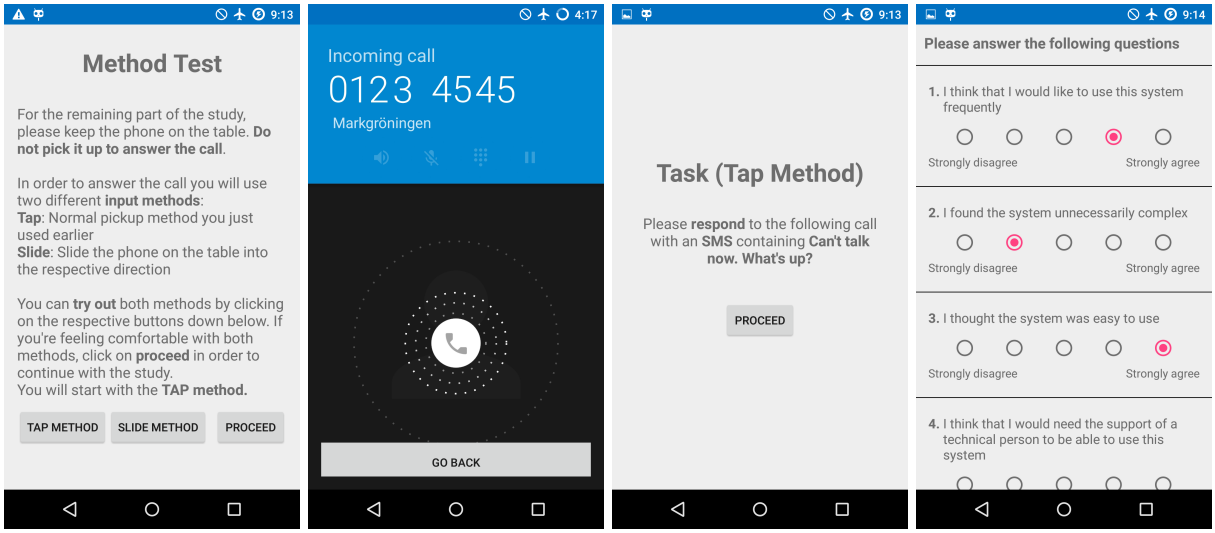
Figure 5.3: Diagram showing the workflow of our study tool. The pickup-test and both input method tasks are structured in detail down below.

5.4 Procedure

The whole study took place on three days, while having four to seven participants per day. On average, every session took 30 minutes. In order to perform the study our participants did not need any additional tools except the smartphone which we provided. Every participant performed the study at the same desk which is located in our office. Throughout the study we made sure that the participant sat on a chair and that the sound level of the surrounding environment was kept to a minimum.
At the beginning of the study, we shortly introduced us to the participant and offered a chair to take a seat. We handed in a consent form which consisted of information about data collecting, risks and benefits, the participant's right, additional contact information and whether we were allowed to take images, to record videos or record audio while the participant was performing the study. We pointed out that they are not forced to give us those recording rights and that it is indeed their privilege to say no. It was important to us that the participants felt comfortable throughout the study and that they perform this study on a voluntary basis and were not obligated to do things they were not agreed with. We told them that they could ask questions or even abort the study at any given time and also thanked them beforehand for their helpfulness. While the participant was reading the consent form we prepared the smartphone for

Figure 5.4: A participant performing the study using the slide input method.

the upcoming tasks by selecting the desired order of input methods. By clicking on the subheading study located at the top of the welcome screen (see Figure 5.2a) a window will pop up which allows the selection of the order the input methods will be used later on (see Figure 5.2b). We switched the order of input methods after every participant by using Latin Square. After we received the filled in consent form from the participants we handed the smartphone over and instructed them to fill in all relevant personal information, such as age, gender, etc. (see Figure 5.2a).
We then started to explain the participants their task during the upcoming pickup-test. By clicking on the button START STUDY, we showed them the description of their first initial task (see Figure 5.2c). We made sure that they read it and told them that it is now their task to respond to an incoming call by fulfilling the description the study tool was telling them. We further explained that by clicking on the button PROCEED, a timer will be started which will trigger the incoming call in the end. We handed them a book over and instructed them to start reading as soon as they started the timer. Since it was our intention to find out how often participants pick up their phone when a call is incoming, we had to ensure that the phone is placed on the table before the participants started reading the book. In order to do so, it was sufficient to make a short subliminal remark such as “You can safely put the phone aside while you’re reading since it takes some time till the call comes in”. At last, we announced that this procedure will occur several times and that they should just repeat following the instructions we gave them. During the pickup-test it was the study observer’s task to manually count how often the participants picked up the phone in order to interact with it.



(a) **(b)** **(c)** **(d)**

Figure 5.5: More activities of the study tool.

After ten incoming calls the pickup-test was completed and the study tool automatically displayed the introduction activity for the second part of the study (see Figure 5.5a). As soon as the participants finished reading the text displayed on the screen, we gave them a short summary of what the second part of the study consists of. We told them that their task basically remains the same with the exception that they now have two different input methods they should consecutively make use of and that the call, they should response to, comes faster in than before. Besides the tap method, which they were already familiar with due to the pickup-test, we introduced them to our sliding method by explaining them the general idea behind it. Instead of tapping on the touchscreen by using their fingers, they should slide the phone on the table into the direction where the respective option is located. By clicking on the two respective buttons on the bottom left corner the participants were able to try out both input methods (see Figure 5.5b). We therefore simulated incoming calls within a loop, resulting in an unlimited number of calls the participants could respond to using one of the input method. The participants were allowed to try out both methods as long as they wanted and could return to the previous activity by clicking on the button GO BACK. During this test phase, our study tool automatically counted how often a participant tried out selecting a response option per input method. When the participants felt comfortable with both input methods we noted once again that they firstly start responding to incoming calls by using just one input method and then change it only to the other as soon as we told them. The input method, with which the participants should begin with, is noted at the end of introduction

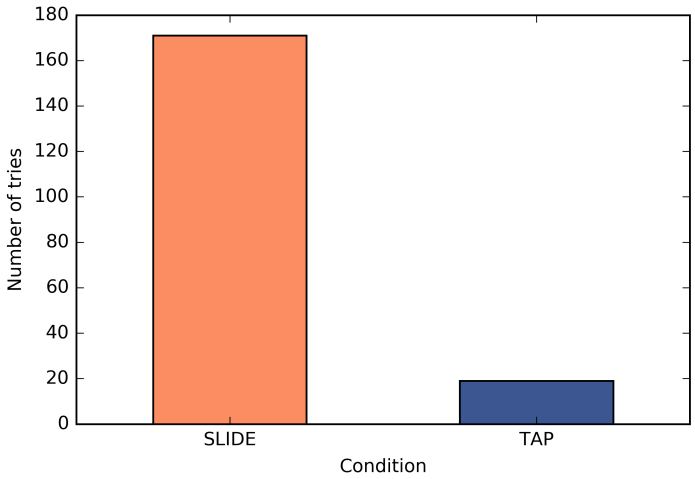


Figure 5.6: Diagram showing the total amount of times participants responded to an incoming call during testing per input method.

activity (see Figure 5.5a). As a form of help, we added a remark to all following task descriptions in order to display the currently used input method (see Figure 5.5c). Every participant had to respond to 42 incoming calls per input method. In order to minimize the possibility of having two identical task descriptions consecutively, we divided the 42 incoming calls into 6 equally sized groups consisting of all 7 possible response possibilities and shuffled the order of every group separately.

After responding to all calls using one input method, the participant was prompted to answer two questionnaires concerning the condition they just experienced. Developed by John Brooke [Bro+96], we first asked our participants to fill out the SUS questionnaire. The SUS questionnaire consists of ten different questions, using the Likert scale, concerning the effectiveness, efficiency and satisfaction of a system. By answering the questions, we receive a single score on a scale of 0 (worst) to 100 (best). Figure 5.5d shows our implementation of the SUS questionnaire within the study tool. Second, to rate the perceived workload by filling in the official NASA-TLX questionnaire [HS88]. We decided to use the raw version of NASA-TLX since we did not expect any noticeable

	Slide		**Tap**	
	Total	**M**	**Total**	**M**
Number of tries	171	8.55	19	2.11

Table 5.1: Number of responded calls the participants tried out per input method.

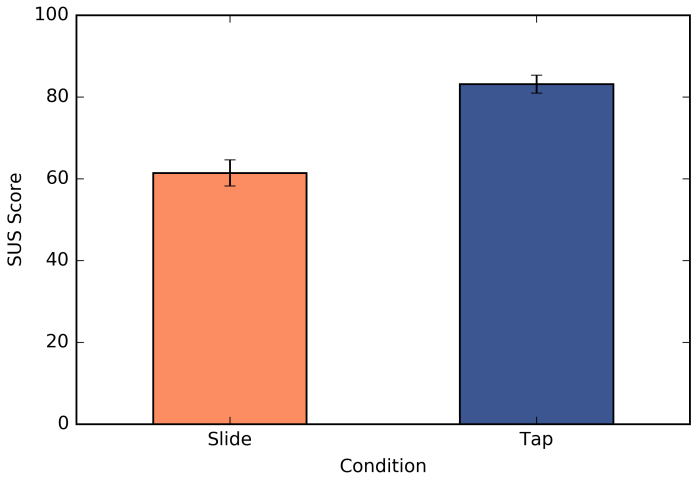


Figure 5.7: Diagram showing the resulted SUS score per input method including standard error.

changes regarding the weighting of the six dimensions influencing the total workload (TW). The TW consists of mental demnad (MD), physical demand (PD), temporal demand (TD), performance (PF), overall effort (EF) and frustration (FR). Each dimension uses a scale which operates with 21 steps and ranges from low to high respectively. Since it is recommended to let the participants to perform this questionnaire using the pen and paper version[1], we did not implement the questionnaire into our study tool. Besides subjective data, objective data was collected by automatically tracking TCT and the number of wrong selections, referred to as ER, for each incoming call. We did not track the time the participant needed for filling in the two questionnaires. After the participants completed both questionnaires we instructed them to use the other input method from now on in order to respond to the call.
The study tool ended after the participant filled in both questionnaires for the second

	Slide		**Tap**	
	M	**SD**	**M**	**SD**
SUS score	61.4	12.34	83.1	8.45

Table 5.2: System Usability Scale (SUS) score per input method.

[1] http://humansystems.arc.nasa.gov/groups/TLX/downloads/TLXScale.pdf (last access 2016-06-20)

	Slide		Tap	
Workload	**M**	**SD**	**M**	**SD**
MD	43.33	24.74	28.33	19.29
PD	47.67	23.23	21.00	17.34
TP	48.00	22.93	37.00	21.97
PF	48.00	22.12	32.67	17.59
EF	44.67	21.33	25.67	16.11
FR	42.33	23.16	25.33	17.56

Table 5.3: Raw NASA-TLX workloads per input method. Mental Demand (MD), Physical Demand (PD), Temporal Demand (TD), Performance (PF), Effort (EF) and Frustration (FR).

time but now regarding the other input method. Additionally, we noted any singularities and comments the participant expressed while performing the study.
At the end of the study we conducted a semi-constructed interview with the participant in regard to their smartphone usage and their opinion about both input methods. We started the interview by asking for their currently owned smartphone, especially whether they use an iOS, Android or another OS powered smartphone. If the participant was only in possession of a regular cellphone instead of a smartphone, we continued asking the same questions since it does not influence their answers for the following interview, but was noted nonetheless. As a follow-up question, we asked the participants to estimate how often they find themselves in such a situation as constructed in the study. In particular, how often they place their smartphone on the table and how often they get called during their everyday. All these questions serve as basis in order to get to know the participant since it is essential to be aware of the context their answers are given in. In addition, by having simple questions at the beginning of the interview, we had the possibility to set the tone for the remaining conversation and raise the participant's comfortableness level. Subsequently, we started asking questions concerning the actual study. The participants were asked to compare both input methods they experienced throughout the study, for example by asking which one of the input methods felt more pleasant to them and why they would think so. Furthermore, we asked for their opinion about the advantages and disadvantages of one input method when comparing it to the other. At last, we asked them whether they could come up with other possible use cases which utilize the sliding interaction. Every question was constructed to be open-ended since we did not want to limit the participant's imaginariness. As soon as we noticed that the participant struggled to

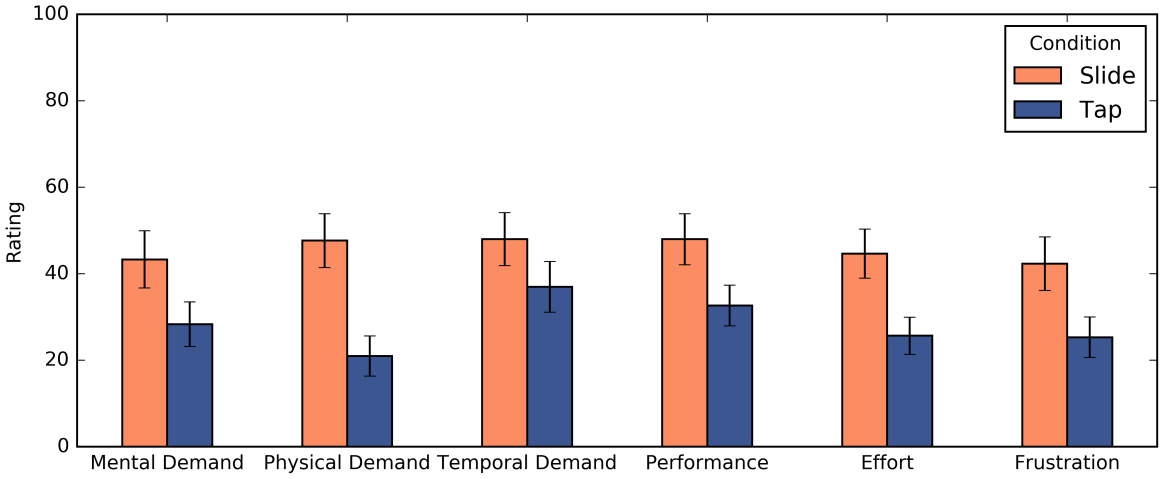


Figure 5.8: Diagram showing the results concerning the NASA-TLX questionnaire including standard error.

answer one question we made some answer proposals and asked for the participants' agreement or disagreement and why they would think so.

5.5 Participants

We recruited 16 participants of whom 6 were female and 10 were male. The participants were aged between 18 to 25 ($M = 21.5$, $SD = 1.9$). Out of 16, only one participant is using the left arm as the dominant one. We had 11 owners of an Android smartphone, 3 with iPhones and 1 with a Windows Phone. One participant did not possess any smartphone before and was using a regular cellphone instead. Concerning our rating scale regarding the participants' familiarity with the Android OS we received a mean score of 5.3 and a standard deviation of 2.1. None of the participants had movement restrictions with their arm. All of our participants were recruited through the university's mailing list and were native German speakers. We did not demand any requirements for participating in our study.

	Slide		Tap	
Task	**M**	**SD**	**M**	**SD**
Accept	1.30	1.93	.45	.67
Decline	1.46	2.50	.44	.42
SMS 1	5.93	4.35	1.82	.66
SMS 2	3.88	3.45	1.87	.69
SMS 3	5.23	3.13	2.29	.87
SMS 4	5.38	3.33	2.18	.83
SMS 5	4.94	3.25	1.48	.48

Table 5.4: Task completion time in (s) per input method and per task. SMS X stands for the x-th listview item.

5.6 Results

During the conduction of our user study, we collected data from the users concerning the usability, reliability and efficiency of *SurfaceSliding*.
During the session, the participant's first task was to complete the pickup-test. From a total of 160 incoming calls, which had to be rejected, the phone was only picked up from the table 53 times by our participants. Out of the 16, nine participants let the phone rested on the table throughout the whole pickup-test. In contrast, only 3 participants picked up the phone every time when a call was incoming. The remaining 4 participants varied their pickup behavior during the test process. Participant P1 and P14 started picking up the phone at the beginning but decided to stop mid-process. In the interview at the end, both participants stated that they eventually saw the actual pick up action as unnecessary and cumbersome since the response action they had to perform only consisted of one or two touches and therefore decided to let the phone rested on the table for the remaining calls. On the contrary, participant P10 let the phone rested on the table during the two initial calls but then picked up the phone for the remaining eight calls. The participant could not explain his behavior as he said that he acted intuitional and did not put much importance in it. However, after drawing the attention to this matter, he also stated like the other two participants that he would now most likely also let the phone rested while performing only micro transactions, e.g. one or two touches. Unusually, participant P13 only picked up the phone during the first and the fourth call. He explained that he only picked up the phone during the first call since he was unfamiliar with the structure of the activity and therefore needed to take a closer look. Concerning his second pickup, he stated that he was confused by the task description since it told him to write an own written

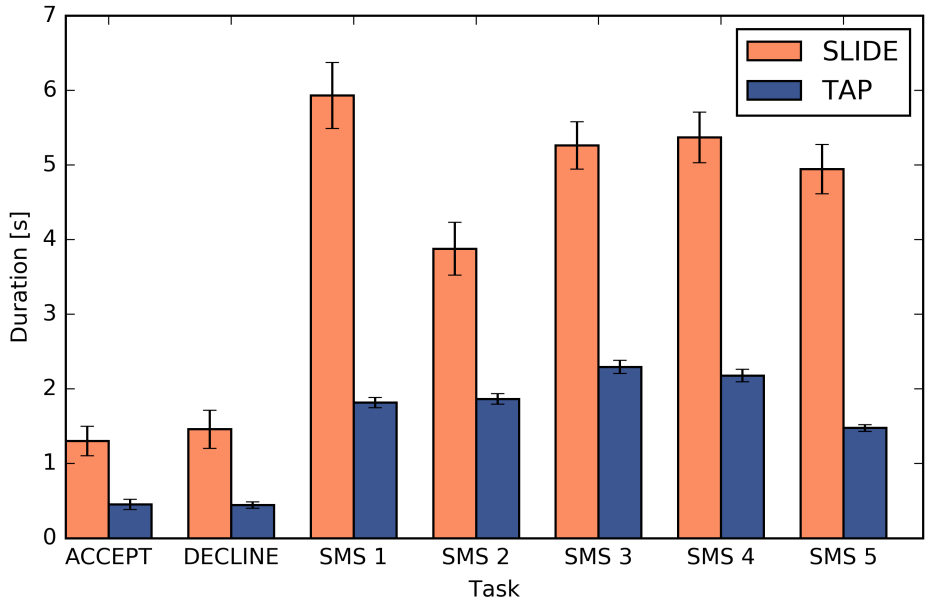


Figure 5.9: Diagram showing the average amount of time participants needed per task type including standard error. The time tracking started at the event FIRSTMOVE (see Section 4.4).

SMS and he initially thought that he actually had to write a SMS using the touchscreen keyboard and therefore decided to pick up the phone.

Following the pickup-test, all participants had the opportunity to try out both input methods. Table 5.1 and Figure 5.6 show the results for this test phase. In total, the sliding input method was tried out 171 times by the participants. When comparing, the sliding input method was tested nine times more often than the traditional tapping, as the participants only tested it 19 times in total. In order to measure the efficiency of both input methods, we measured the TCT and ER of the participants while performing the incoming call task. Table 5.4, Figure 5.9, Table 5.5 and Figure 5.10 show the results concerning TCT and ER per input method and per task. Furthermore, we also measured the subjective opinion of our participants concerning our system by using two questionnaires. First, we measured the input method's usability by asking our participants to fill in the SUS questionnaire. Table 5.2 and Figure 5.7 show the results of the SUS questionnaire per input method. The sliding input method scored a SUS score of 61.4 while the tapping input method received a score of 83.1. Second, to rate the participants' perceived workload by filling in the raw NASA-TLX questionnaire. Table 5.3 and Figure 5.8 show the results for the individual NASA-TLX workloads

	Slide			Tap		
Task	**Total**	**M**	**SD**	**Total**	**M**	**SD**
Accept	17	.173	.38	6	.061	.24
Decline	5	.051	.22	7	.071	.26
SMS 1	23	.235	.42	2	.020	.14
SMS 2	19	.194	.40	5	.051	.22
SMS 3	28	.286	.45	5	.051	.22
SMS 4	24	.245	.43	7	.071	.26
SMS 5	21	.214	.41	5	.051	.22

Table 5.5: Error rate per input method and per task. SMS X stands for the x-th listview item. Total represents the total number of errors throughout the study.

per input method. The content and results of our semi-constructed interview will be addressed later on.

5.7 Remarks

Our study tool is designed to work on all Android smartphones which have an Android Version higher than 4.0.3 (Ice Cream Sandwich, API Level 15) and a functional front camera. As apparatus we used the same Nexus 5 as we developed on in Chapter 3 and in Chapter 4. For general usage, we decided to use English as the main language for all instructions within the study tool. As mentioned, our study tool automatically tracks the time the participant needed in order to select the appropriate answer and the number of times a wrong selection was made. In addition, the study tool also saves all user relevant data and the participants' answers concerning the SUS questionnaires. All data will be saved as .csv files on the internal storage of the phone located at `\SurfaceSliding\Output`.

5.7.1 Ceiling and lighting conditions

As stated in Section 5.4, every participant performed the study while sitting at a table within our office. This is important since our sliding input method strongly relies on the optical flow based on the images delivered by the phone's front camera. Due to this, we had to make sure that throughout the whole study all participants share the same ceiling in order to guarantee equal conditions from participant to participant.

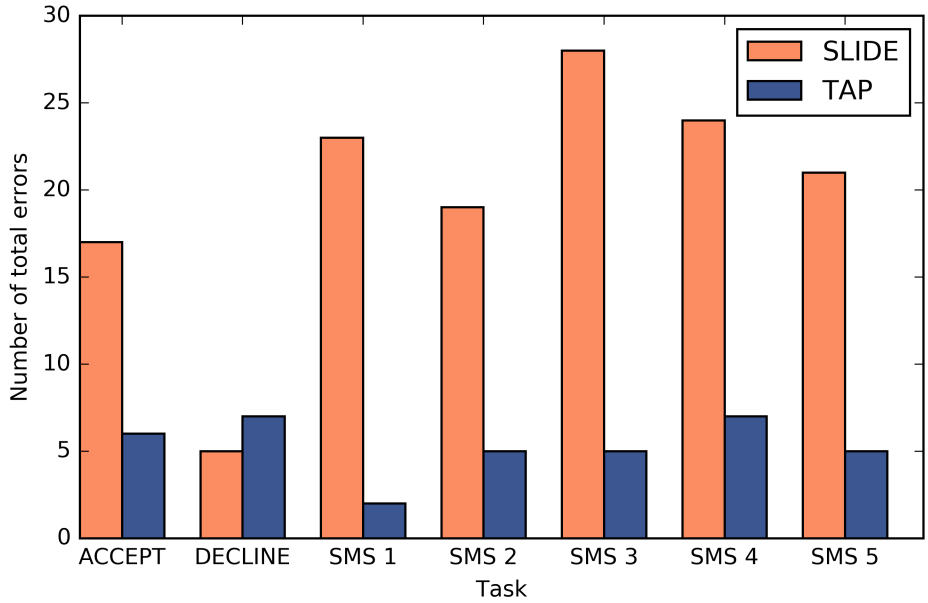


Figure 5.10: Diagram showing the total amount of errors participants made per input method.

Figure 5.11 shows an image of the ceiling in our office, taken with the Nexus 5's front camera while it was placed on the table. The ceiling is made from concrete and originally only featured a single light strip. We stuck two AR markers to the ceiling in order to improve the precision of our optical flow based algorithm. Since the desk is directly aligned to a wall, the image also features the side wall of our office including the blackboard we hang there. The lighting conditions within our office room remain same through all three days and we did not turn on the light strip on the ceiling.

5.7.2 Remarks concerning sliding input method

Besides the ceiling, we also had to make sure that every single task, a participant completed by using the sliding input method, held the same start conditions. In order to archive this, we drew a marker directly on the table by outlining the phone's proportions with a thin pen. During the study as we introduced the participants to our sliding input method we told them that they should always place the phone back to the marker whenever they finished a task. By doing this, we ensured that the actual sliding always originated from the same starting position and thus guaranteed the same start conditions throughout all sliding tasks.

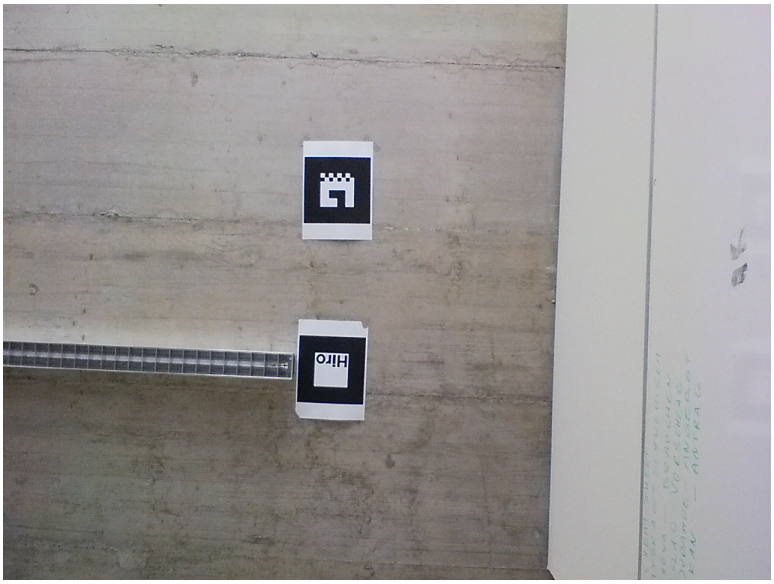


Figure 5.11: Image taken by the phone's front camera while lying on the table.

Additionally during the introduction to our sliding input method, we made it clear to the participants that we are testing our system and by no means the participants themselves. We told them that in cases of wrong behavior, e.g. wrong selections due to misinterpretation of optical flow data, they should simply ignore them and instead continue with the next task.

5.8 Summary

This chapter focused on the structure and design of our user study concerning the use case of responding to an incoming call using two different input methods. Out of seven different response possibilities, the participant's task was to select the right one by using either tapping or sliding. We developed an Android based study tool in order to conduct the study and automatically track all relevant variables including the task completion time, error rate and the participant's rated SUS score. On top of that, we also let our participants fill in the NASA-TLX questionnaire in order to get their subjective opinion about the overall task workload. Furthermore, we described the general procedure of our user study which consisted of three tests. First, we looked into the participant's pickup behavior whenever they receive an incoming call while the phone was lying on the table. Interestingly only a third of our participants decided to pick up the phone which showed that the majority of people rather tend to let the

phone rest on the table whenever they had to react to an incoming call event which only requires the performing of one or two touch actions. The two remaining tests consisted of the response of an incoming call by using both input methods consecutively for navigating and selecting the respective option. We let our participants to test out both input methods beforehand in order to let the participants get to know with the overall system and especially the interaction through sliding. The chapter also presents the results which we collected with the help of our 16 participants. Their meaning and their interpretation will be discussed later on. Looking at the results later on, we expect better objective ratings using the tap input method but a high attractiveness for the sliding input method in regard to its usability and applicability. Additionally, we also addressed different remarks we experienced throughout the study which should be considered for future iterations.

6 Discussion

As stated in Section 2.1, our initial motivation relied on the fact, that a considerable amount of people tend to lay their phone on the table while they are at home or the office. We further extended these findings within the user study by checking the users' behavior via a simple pickup-test. The number of actual pickups and the statements from the participants showed that there is indeed a demand for having an on-table interaction which eventually saves the user's effort for picking the phone up when only performing micro transactions. Participant P1 and P14 especially stated that the actual pickup action is rather unnecessary and cumbersome in their eyes when only performing one or two touches on the front touchscreen.
Another aspect focused on the learnability of *SurfaceSliding*. After the initial pickup-test, the participants had the opportunity to try out and get familiar with both input methods which should be used during the remaining session. As mentioned earlier in Section 5.4, the participants could try out both input methods as long as well as how often they wanted. Obviously, it should be clear that the participants were already familiar with the concept of the traditional tapping method since the majority of our participants owned a smartphone for themselves. In addition, since the participants already performed the pickup-test up to that point, most of them only were interested in getting to know the sliding input method. Given the high number of sliding tests, it is safe to say that it is indeed necessary to teach the users how they should slide the phone in order to perform a specific action, e.g. how to navigate through a listview by using only sliding movements. Interestingly, although four participants tested the sliding input method only four times or less, we did not find any significantly difference concerning their task completion time or error rate. Yet, it should be noted that even if these four participants did not test the sliding input method as much often as the others they still received our introduction concerning the usage of the sliding input method for navigating within the incoming call activity. Participant P7 even stated in the interview that it is definitely necessary to introduce the future users to the new sliding input method, for example by offering some sort of tutorial within the Android system which could be based on the introduction of ours. In addition, by having no significant differences between the four participants and the others concerning their TCT and ER, it shows that our sliding input method is easily learnable and does not require a long training period. This conclusion is supported by the responses the

participants gave us after asking them about the advantages of *SurfaceSliding* as they mentioned its easy learnability.
We also stated in the previous chapter that we intended to compare both input methods concerning their usability by using the System Usability Scale. Our sliding input method only scored a SUS score of 61.4 while the tapping input method received a score of 83.1. This also means that our system falls under the average since the official average SUS score is 68[1]. However, we do not think that a SUS score of 61.4 consequently equates to a bad system since we have to reconsider several things. First, the implementation of a use case which is familiar and known to all participants includes the problem that the majority of participants already knows how the original approach is working and implemented. It is always difficult to introduce the users to a new interaction method which basically tries to mimic the logic behind an already existing interaction and expect an above average reaction or judgment by the users. It often takes a remarkably long time or even multiple iterations of a system in order to convince the user base of using a new interaction method instead of the old and conventional one. *SurfaceSliding* should be seen as the first representative concerning the possibility of interacting with the phone by performing sliding movements on a flat surface. Therefore, we are confident that the user base's opinion about such an interaction method should change positively in the near future or following iterations. The momentarily SUS score can therefore be used as a general guideline for future comparisons.
Furthermore, we also measured the objective quality of our system in form of TCT and ER. Looking at the values in Section 5.6 it is clear, that the sliding input method is by no doubt inferior when comparing to the traditional tapping method. As mentioned in Section 5.1, we already expected such results. First, since *SurfaceSliding* presented a new interaction method for all of our participants, it definitely hindered their overall performance since all of our participants did not have any experience beforehand and were unfamiliar with its proper usage. Second, looking at the distances the users had to cover by using each input method, also favors the traditional tapping method. Although we did not record or measure the distances the phone was slid for, it was obvious to outsiders that sliding covered a distance many times over the distance the participants' index finger moved while using the traditional tapping method. This becomes even clearer when comparing the processes of both input methods for the selection of a text message within a listview. While using tapping, the participant only had to perform one on-screen sliding movement and one clicking action. On the other hand, using sliding, the participant had to perform one physical sliding movement to

[1] https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html (last access 2016-06-20)

the top and in worst case additional four downwards and one sliding movement to the right. This insight can also be observed when comparing the task completion time differences between sending a text message and declining/accepting an incoming call in respect to the particular input method. While it took 933 ms more at average when using *SurfaceSliding* in order to decline or accept a call, the average participant needed 2795 ms more time when instructed to send a response message when comparing with the traditional tapping method. This fact also bugged more than the half of our participants. The majority stated that they definitely felt slower in comparison to tapping when they were instructed to send a text message back. In their opinion, the main reason was the fact that the selection of the correct text message took way too long for them. When sliding the phone either up- or downwards, it often occurred for them that the momentarily selection skipped one or two listview items at once and thus resulted in missing the right selection. As a result, the majority of our participants expressed the opinion that using sliding for selecting the right item within a listview was rather cumbersome than innovative or helpful. Yet, they liked the concept of sliding the phone either to the left or to the right in order to perform micro transactions which only have two possible answer options. When asking why, they said that they did not experience any noticeable differences concerning their time they needed when performing a simple sliding movement to the side. All of them agreed to our proposal when we proposed that *SurfaceSliding* should rather be used for micro transactions which do not require a high amount of the user's attention and that we should stick with a sliding input method which should only consider the rough direction of the sliding once, for example detecting whether the phone was sliding up-/downwards or to the left/right, instead of continuously checking the momentarily position of the phone. We were quite surprised by this statement since it was originally our initial goal to implement a system which is capable of precisely determining the momentarily phone's position on the table. Nevertheless, we did not take this for granted since this is only referring to the incoming call use case. A change within the participants' opinions could easily be experienced when implementing another use case.
Another major concern focuses on the fact that our sliding input method was remarkably more prone to errors when comparing to traditional tapping. About 90% of our participants also expressed their concern that they made noticeably more mistakes while using *SurfaceSliding* and thus eventually influenced their general opinion towards sliding. This error behavior can be broken down to several reasons. Firstly, having an optical flow based interaction consequently imply that our system is easily influenced by the near surroundings which are been captured by the phone's front camera. Light conditions, unintended movements or the lack of trackable characteristics within the front camera's field of view can definitely influence the performance of our system. Second, unlike to our first prototype in Section 3.2.2, our optical flow based algorithm is unfortunately not resistant to potential rotations of the phone. Rotating the phone

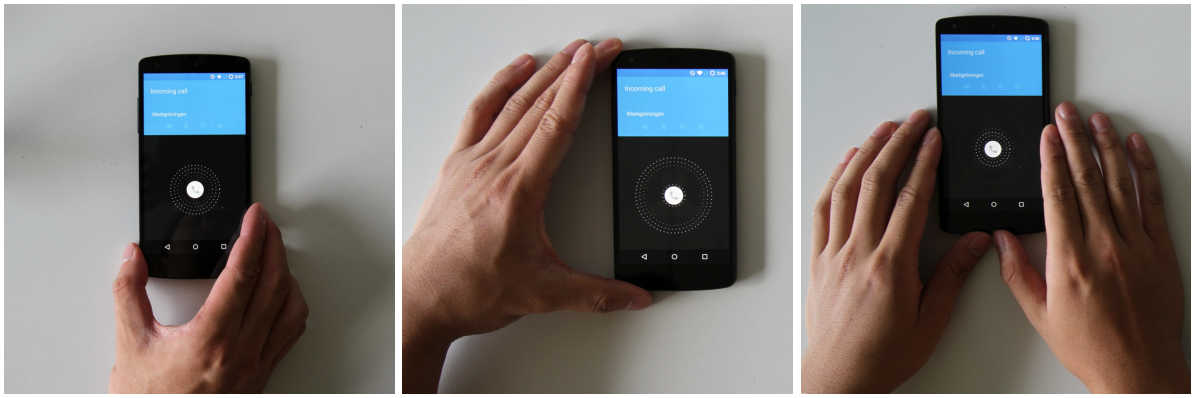
directly affects the system since the front camera detects a change within the optical flow of the images. Thus, we often observed that the system selected the wrong response option when our participants unintentionally rotated the phone while sliding it. This observation was confirmed by two of our participants when they stated that they felt that our system is highly prone to rotations or unintentionally bumps. At last, looking back at the state machine we used for our prototype (see Section 4.3), reveals another problem concerning the processing of the sliding data. Being in the TEXT MSG state, it is only possible to close the listview by sliding the phone to the left. However, by continuously sliding to the left, it often occurred, according to four participants that the system automatically selected the decline option since it was in IDLE state at that point and therefore already waited for the next sliding movement, thus resulting in an error. Fixing these problems should definitely be a top priority when improving the overall system since the majority of our participants saw the most weakest point of *SurfaceSliding* within the fact that it was unusually high prone to errors.
As mentioned in Section 5.1, we also expected that the TW determined by the NASA-TLX would show the same trend as the task completion time and error rate. As a matter of fact, every single component scale was higher when using *SurfaceSliding*. However, different factors have to be considered when looking at these values. A higher mental demand can therefore be reduced to the fact that an interaction method like *SurfaceSliding* also automatically requires more thought process from the participant since it involves a new concept idea which was not known to our participants yet. As mentioned earlier, participants had to cover a higher distance while using *SurfaceSliding* since it required physically sliding the phone over the table rather than only perform on-screen touch actions. As a result, we expected that *SurfaceSliding* would receive a higher score concerning its physical and temporal demands as well as its effort score. Combining these observations with the fact that the majority of our participants noted *SurfaceSliding*'s high vulnerability to errors it consequently implies that also the participants' performance and their frustration would come out higher when comparing to traditional tapping. It would be definitely interesting to see whether these values tend to be the same when developing *SurfaceSliding* further on. By fixing the main problem concerning the high error proneness of our system, we expect significantly better results in regard to the participants' performance and frustration workload. Similar to its SUS score, these values should also be considered as a general guideline for further improvements in the near future rather as an unfortunate setback.

As mentioned in Section 5.2, we performed a semi-constructed interview at the end of each session in regard to the participant's general opinion about *SurfaceSliding* and potential areas of improvement. Each interview took about five minutes and should gave us an insight about the usability of *SurfaceSliding* when comparing it to a

conventional input method like tapping.
At the beginning of each interview session, we asked the participants whether they often find themselves in a situation like in the user study where their phone is lying on the table and a call is incoming. While half of our participants stated that their phone is often resting on a nearby table or shelf, all 16 participants said that it is rather unusual that they are receiving calls nowadays. After asking why, we found out that the majority of our participants rather relay on various text message services like WhatsApp or Google Hangout. However, it should be noted that all of our participants were full-time students, whose ages ranged from 18 to 25, and thus it is highly possible that the answers could differentiate when taking another persona group in account. Nevertheless, conducting a use case which does not often appear in the participants' everyday routine can definitely influence their overall opinion about the application of our interaction method since it could occur that the participants do not see its usefulness in the first place. Therefore, we recommend for future user studies to pick another use case which should take place more often during the participants' everyday life.
We continued the interview by asking the participants to describe their point of view concerning the main differences between the two presented input methods. The majority of the received answers focused on the fact that in contrast to *SurfaceSliding*, the tapping method was already known to all of our participants beforehand and thus required no training phase in order to use it. In addition, participant P5 and P11 said that it is unfortunate that *SurfaceSliding* is restricted to the fact that a flat surface is required for proper usage whereas tapping can also be done while walking and holding the phone in one hand midair. They proposed the idea of translating the concept of *SurfaceSliding* to the third dimension, so that movements, like drawing a circle or wave midair while having the phone in the hand, can be detected and respectively interpreted. Related to this, seven of our participants stated their concern about the required space *SurfaceSliding* needed while using it since the participants own tables often do not offer the same amount of free space as we provided during our user study. Another big major concern our participants expressed was their fear of damaging the phone's outer condition while sliding it. They especially worried about the phone's back camera since most smartphones nowadays feature a back camera which slightly sticks out when comparing to the back surface of the phone. When looking at the current trend of smartphones, in which smartphones become slimmer after each generation, yet the back camera's thickness often remains the same, we definitely share the participants' concern. Throughout the development of *SurfaceSliding*, we even noticed that the overall condition of our Nexus 5's back camera got slightly worse and that the sliding movements clearly left their marks.
Besides their various concerns, our participants also shared their positive opinion with us concerning the general concept of having an on-table interaction like *SurfaceSliding*.

Participant P6 stated that using sliding movements require less cognitive attention when comparing it to traditional tapping since it is not required to perform a pinpoint sliding movement when declining an incoming call for instance. Using the traditional tapping method on the other hand, requires first-off the whole cognitive load from the user and additionally an exact on-screen sliding movement using the fingers. Linked to this recognition, P6 also mentioned the advantage that *SurfaceSliding* offers the possibility of interacting with the phone unobtrusively. Although he put his phone into silent mode, he and a number of spectators noticed during the presentation that he was receiving a call. Even though the phone did not ring loudly, it was still clear to outsiders since the phone's screen was flickering during that moment. Having an interaction method like *SurfaceSliding*, it would have been beneficial to the participant since he would have been able to respond to the caller accordingly without interrupting the flow of the presentation. When asking the remaining participants whether they share the same opinion like P6 concerning the possibility of interacting with the phone unobtrusively, the majority of them expressed their general consents in this matter. We certainly share the participants' opinion and also see it as a clear advantage for *SurfaceSliding* and therefore propose and recommend to further investigating this matter in future iterations.
One question we asked our participants during the interview was whether they could imagine themselves in using *SurfaceSliding* for their own when having the possibility to do so. Out of 16, only four participants stated that they would use it for their own. After further questioning, it became clear that our participants do not see *SurfaceSliding*'s field of application in being a primary input method but rather an additional alternative one. The majority of our participants stated that they could imagine that people who either struggle with using the usual tapping method or have problems with their eyesight could certainly benefit from *SurfaceSliding*. Four participants stated that especially blind people could use *SurfaceSliding* in order to navigate through a menu or react to different incoming events. In order to improve the user feedback for blind people they proposed to include vibration events which can be utilize to indicate actions or changes while sliding. However, other participants also mentioned that being an alternative input method does not automatically mean that the main target group consists of people who have problems with interacting with their phone. Instead, it could be used in cases where the interaction with the phone is obstructed by external circumstances like having dirty hands while receiving a call or interacting with the phone while concentrating on other things simultaneously.

During the conduction of our user study, we took notice of different things concerning the overall system of *SurfaceSliding* which are worth mentioning.
One interesting aspect focuses on the different ways our participants grabbed the phone in order to perform the sliding movements. Figure 6.1 shows all variations of



(a) (b) (c)

Figure 6.1: All three grabbing variations our participants used while sliding the phone.

our participants' grabbing methods. The majority of our participants used their right hand while grabbing the phone on both sides (see Figure 6.1a). Four participants tried to slide the phone while grabbing it at the top and bottom simultaneously (see Figure 6.1b). Interestingly, only participant P7 used both hands in order to grab the phone (see Figure 6.1c). Participant P10 was the only person in our user study who is using the left hand as the dominant one. Yet, the participant decided to use the right hand for grabbing and sliding the phone. After asking why, the participant stated that the decision was rather intuitive than intentional. Unfortunately, we did not find any dependency between the grabbing method and the participant's performance. However, it was interesting to see that people tend to different approaches when being introduced to a new interaction method.

Another thing we noticed during the user study is the fact, that the usage of an optical flow based algorithm also implies a higher drainage of the phone's internal battery since it accesses the front camera throughout the whole interaction which requires more processing power in the end. As noted in Section 5.4, a session took in average about 30 minutes. After those 30 minutes, we observed that the internal battery of our Nexus 5 got eventually decreased by about 10%. This certainly shows another drawback when utilizing the front camera instead of the inertial sensors of a smartphone since these require much less processing power.

During the semi-constructed interview, participant P4 mentioned an idea for improving the general recognition rate of *SurfaceSliding* which is noteworthy in our eyes. The participant suggested implementing a mechanism which signalizes the system whenever an actual sliding movement is performed. Having such a mechanism, prevents the occurrence of input actions which were caused by unintentionally sliding movements, e.g. by unintentionally bumping the phone while trying to grab it or bumping against

Sliding Action	Mean Time in ms
Slide (Right-to-Left, 8 cm)	1221.70
Slide (Left-to-Right, 8 cm)	1409.42
Slide (Upwards, 4 cm)	907.19
Select Item in Listview	4724.63

Table 6.1: Table showing our KLM-similar model specially designed for sliding movements based on the results we collected during the conduction of our user study.

the table and resulting in a shake motion of the phone. The mechanism, the participant proposed, relies on the detection of the number of fingers which are placed on the touchscreen. Consequently, the system should only interpret the sliding data whenever the user is placing at least two fingers on the touchscreen. While the fingers are placed on the screen, the user would then proceed to slide the phone and in the end lift both fingers back up and let the system interpret the recorded data from the executed sliding movement. This also enables the possibility of "drawing" gestures on the table through sliding the phone and having these recognized by the overall system afterwards. In addition, by recording the sliding data only during specific periods of time also allows to save battery power as mentioned earlier since the system does not need to access the front camera all the time.

As mentioned, we also had the intention to define a KLM-similar model which should be specially designed for sliding movements which are performed by using *SurfaceSliding*. Table 6.1 shows the times an average user needed in order to perform different sliding actions. The times should been seen as a rough guideline which can be used as a reference point for future iterations. A horizontal sliding movement covered approximately a distance of 8 cm while a vertical one covered 4 cm. The sliding action "Select Item in Listview" represents the time from displaying the listview to the point of the selection of one listview item.

7 Conclusion & Future Work

Motivated by the fact that about half the people tend to place their personal smartphone on their table while they are at home or at the office, we had the idea of developing an on-table interaction which enables the interaction with the phone by physically sliding it on a flat surface. This on-table interaction, which we called *SurfaceSliding*, would actually save the user's effort of physically picking the phone up in order to interact with it. Besides the work of Wiese et al. [WSB13], which addressed the topic of people's phone placement behavior during their every day, we also took a look at previous work who either dealt with a similar problematic or provided important insights about different issues related to our work. Inspired by their works, we took the initial approach of implementing *SurfaceSliding* by utilizing the different inertial sensors of nowadays smartphones for tracking the momentarily position of the phone when being slid on a table. Our first prototype focused on utilizing the acceleration and rotation sensors of the Nexus 5 and double integrating its data in order to calculate the momentarily change of the phone's position while sliding. However, we quickly realized that this approach was unfortunately highly prone to emerging drift effects due to the fact that nowadays smartphone sensors do not offer the necessary accuracy. Looking for better results and inspired by the approach of sensor fusion, we extended the functionality of our prototype by implementing a movement detector in form of an optical flow based algorithm using the images captured by the phone's front camera while lying on the table. Unfortunately, the emerging drift effects still persisted due to the incapableness of the phone's front camera of keeping up with the high sampling rate of the inertial sensors. Pleasantly surprised, we eventually stuck with the approach of implementing a prototype which relied on the optical flow data rather than the acceleration data given the inertial sensor. Having a working prototype, we conducted a focus group with the aim of finding applicable use cases for our system. As the participants proposed several interesting use cases, we decided to implement a use case within our system which consisted of the possibility of responding to an incoming call by using *SurfaceSliding*. We conducted a user study for comparing *SurfaceSliding* to the more conventional tapping method with regard to its performance, efficiency and usability. Although the results revealed *SurfaceSliding*'s inferiority to the traditional tapping method, we yet received several proposals and interesting insights about the participants' opinions regarding our interaction method. One main concern, which

the majority of our participants shared, was *SurfaceSliding*'s high proneness to a various number of side effects and therefore resulted in inaccuracy or malfunctions. Another concern lies in the fact that the constant sliding of the phone can eventually cause abrasion to the phone's outer condition, especially the phone's back camera. However, based on the responses of our participants we have recognized that the strength of *SurfaceSliding* lies in the possibility of being an additional input method for navigating and interacting with the smartphone. Especially groups of people who do have problems with regularly operating a smartphone, e.g. blind people, can eventually benefit from an interaction method like *SurfaceSliding*. In addition, the possibility of interacting with a smartphone unobtrusively and without the need of requiring a high cognitive load has to also be considered when looking at *SurfaceSliding*'s advantages. Additionally, inspired by the work of Card et al. [CMN80], we defined a similar KLM model for sliding movements which should act as rough guideline for determining the needed time when performing different tasks using *SurfaceSliding*.

7.1 Future Work

As mentioned in Chapter 6, one advantage of *SurfaceSliding* lies in the possibility of interacting with the phone unobtrusively. Combined with the fact that *SurfaceSliding* does not require a high amount of cognitive load, future work could design a second user study which takes advantage of those two. It would be interesting to see how well users would perceive the interaction when taking the incoming call use case within context. For instance, designing an use case in which one participant's task is to respond to an incoming call by using *SurfaceSliding* while simultaneously performing another task could help at investigating this matter.
Furthermore, as stated in Chapter 6, participant P4 proposed the idea of implementing an additional mechanism for signalizing the performance of an actual sliding movement in order to prevent the interpretation of data which arose from unintended actions like bumping the phone when trying to grab it. For implementing, P4 suggested using the touchscreen for detecting the number of fingers the user placed on. Collecting more data concerning the actual sliding movement and the possibility of interpreting the data afterwards could eventually result in a lower error rate as well as help at the recognition of different "drawing" movements as mentioned in Figure 3.2.

Another limitation focuses on the selection of our apparatus. As stated in Section 3.2.1, we solely used the Google Nexus 5 when developing *SurfaceSliding*. Our system's performance is highly dependent on the selection of the smartphone since it mainly relies on the accuracy and reliability of different hardware components, like the inertial

sensors or the front camera. Future work should therefore look into the system's overall performance when using another smartphone which features better hardware components than the Google Nexus 5. More accurate acceleration sensors could then translate to more precise algorithms for determining the phone's momentarily position while being slid. Having a more accurate system could eventually help at implementing the other use cases which we explored during the conduction of our focus group (see Section 3.3), e.g. simulating a mouse computer or exploring a map/image through sliding the phone on a table.

Another future work deals with the further extension and integration of *SurfaceSliding*. Like mentioned earlier in Chapter 6, our participants liked the idea of implementing *SurfaceSliding* as an additional input method. As we only developed *SurfaceSliding* as an independent Android application in this work, it would be nice to see when future work could integrate it deeper into the Android system itself, assuming that the position's recognition is working accurately enough. For example, future work could integrate it as an independent background service whose information can then be read by third-party applications allowing the recognition and interpretation of sliding movements.
Furthermore, having a more stable and accurate system could also enable the recognition of gestures or movements which are been performed in the third dimension. Taking an additional dimension in account removes the limitation of being restricted to a flat surface and also extends the portfolio of possible "sliding" movements.

Bibliography

[Bro+96] J. Brooke et al. “SUS-A quick and dirty usability scale.” In: *Usability evaluation in industry* 189.194 (1996), pp. 4–7 (cit. on pp. 55, 62).

[CMN80] S. K. Card, T. P. Moran, and A. Newell. “The Keystroke-level Model for User Performance Time with Interactive Systems.” In: *Commun. ACM* 23.7 (July 1980), pp. 396–410. ISSN: 0001-0782. DOI: 10.1145/358886.358895 (cit. on pp. 49, 82).

[GWP12] M. Goel, J. Wobbrock, and S. Patel. “GripSense: Using Built-in Sensors to Detect Hand Posture and Pressure on Commodity Mobile Phones.” In: *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. UIST ’12. Cambridge, Massachusetts, USA: ACM, 2012, pp. 545–554. ISBN: 978-1-4503-1580-7. DOI: 10.1145/2380116.2380184 (cit. on p. 17).

[Goe+14] M. Goel, B. Lee, M. T. Islam Aumi, S. Patel, G. Borriello, S. Hibino, and B. Begole. “SurfaceLink: Using Inertial and Acoustic Sensing to Enable Multi-device Interaction on a Surface.” In: *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*. CHI ’14. Toronto, Ontario, Canada: ACM, 2014, pp. 1387–1396. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557120 (cit. on pp. 17, 23).

[Gve13] I. Gvero. “Observing the User Experience, 2Nd Edition: A Practitioner’s Guide to User Research by Elizabeth Goodman, Mike Kuniavsky, and Andrea Moed.” In: *SIGSOFT Softw. Eng. Notes* 38.2 (Mar. 2013), pp. 35–35. ISSN: 0163-5948. DOI: 10.1145/2439976.2439993 (cit. on p. 39).

[HS88] S. G. Hart and L. E. Staveland. “Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research.” In: *Human mental workload* 1.3 (1988), pp. 139–183 (cit. on pp. 55, 62).

[KYR10] H. Ketabdar, K. A. Yüksel, and M. Roshandel. “MagiTact: Interaction with Mobile Devices Based on Compass (Magnetic) Sensor.” In: *Proceedings of the 15th International Conference on Intelligent User Interfaces*. IUI ’10. Hong Kong, China: ACM, 2010, pp. 413–414. ISBN: 978-1-60558-515-4. DOI: 10.1145/1719970.1720048 (cit. on p. 17).

[Li+13] C.-L. Li, C. Laoudias, G. Larkou, Y.-K. Tsai, D. Zeinalipour-Yazti, and C. G. Panayiotou. “Indoor Geolocation on Multi-sensor Smartphones.” In: *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*. MobiSys ’13. Taipei, Taiwan: ACM, 2013, pp. 503–504. ISBN: 978-1-4503-1672-9. DOI: 10.1145/2462456.2465704 (cit. on p. 22).

[ML14] W. McGrath and Y. Li. “Detecting Tapping Motion on the Side of Mobile Devices by Probabilistically Combining Hand Postures.” In: *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*. UIST ’14. Honolulu, Hawaii, USA: ACM, 2014, pp. 215–219. ISBN: 978-1-4503-3069-5. DOI: 10.1145/2642918.2647363 (cit. on p. 17).

[Neg+12] M. Negulescu, J. Ruiz, Y. Li, and E. Lank. “Tap, Swipe, or Move: Attentional Demands for Distracted Smartphone Input.” In: *Proceedings of the International Working Conference on Advanced Visual Interfaces*. AVI ’12. Capri Island, Italy: ACM, 2012, pp. 173–180. ISBN: 978-1-4503-1287-5. DOI: 10.1145/2254556.2254589 (cit. on pp. 20, 21, 23).

[RB10] J. Rico and S. Brewster. “Usable Gestures for Mobile Interfaces: Evaluating Social Acceptability.” In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’10. Atlanta, Georgia, USA: ACM, 2010, pp. 887–896. ISBN: 978-1-60558-929-9. DOI: 10.1145/1753326.1753458 (cit. on pp. 21–23).

[RL11] J. Ruiz and Y. Li. “DoubleFlip: A Motion Gesture Delimiter for Mobile Interaction.” In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’11. Vancouver, BC, Canada: ACM, 2011, pp. 2717–2720. ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1979341 (cit. on p. 21).

[RLL11] J. Ruiz, Y. Li, and E. Lank. “User-defined Motion Gestures for Mobile Interaction.” In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’11. Vancouver, BC, Canada: ACM, 2011, pp. 197–206. ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1978971 (cit. on pp. 19, 21, 23).

[SC07] K. Seifert and O. Camacho. “Implementing positioning algorithms using accelerometers.” In: *Freescale Semiconductor* (2007) (cit. on pp. 27, 44).

[SR11] U. Shala and A. Rodriguez. “Indoor positioning using sensor-fusion in android devices.” In: (2011) (cit. on pp. 22, 23).

[WCV14] W. Waqar, Y. Chen, and A. Vardy. “Incorporating User Motion Information for Indoor Smartphone Positioning in Sparse Wi-Fi Environments.” In: *Proceedings of the 17th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. MSWiM ’14. Montreal, QC, Canada: ACM, 2014, pp. 267–274. ISBN: 978-1-4503-3030-5. DOI: 10.1145/2641798.2641812 (cit. on p. 22).

[WSB13] J. Wiese, T. S. Saponas, and A. B. Brush. “Phoneprioception: Enabling Mobile Phones to Infer Where They Are Kept.” In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’13. Paris, France: ACM, 2013, pp. 2157–2166. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2481296 (cit. on pp. 13, 15, 16, 23, 41, 81).

[Woz+16] P. Wozniak, N. Goyal, P. Kucharski, L. Lischke, S. Mayer, and M. Fjeld. “RAMPARTS: Supporting Sensemaking with Spatially-Aware Mobile Interactions.” In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI ’16. Santa Clara, California, USA: ACM, 2016, pp. 2447–2460. ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858491 (cit. on pp. 18, 23).

[Zha+15] C. Zhang, A. Guo, D. Zhang, C. Southern, R. Arriaga, and G. Abowd. “BeyondTouch: Extending the Input Language with Built-in Sensors on Commodity Smartphones.” In: *Proceedings of the 20th International Conference on Intelligent User Interfaces*. IUI ’15. Atlanta, Georgia, USA: ACM, 2015, pp. 67–77. ISBN: 978-1-4503-3306-1. DOI: 10.1145/2678025.2701374 (cit. on pp. 17, 23).

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature