Hierarchical Gestures: Gestural Shortcuts for Touchscreen Devices

Master thesis

Frieder Loch
Human-Media-Interaction
University of Twente
frieder@friederloch.de

Abstract

The aim of this project was to evaluate whether hierarchical gestures are a usable interaction technique. Hierarchical gestures are a technique to construct a gesture set for touchscreen devices that I propose in this report. The gestures of the vocabulary are constructed by chaining primitives, for instance *Call* and *Frieder*, to the command *Call Frieder.* These primitives are assigned with gestures. A command is invoked by a concatenation of the gestures that are assigned to the primitives that form the command.

I expected hierarchical gestures to be a usable technique to control a smartphone. The vocabulary is motivated by the benefits that are gained from its structure, which are better learnability and higher expressivity. Furthermore, using hierarchical gestures is expected to be more efficient and to be more satisfying than using the standard interface of the smartphone. Besides, gesture-based interfaces are suitable for mobile interaction because they allow for one-handed use and require low visual attention.

To validate the hypotheses, an application for Android smartphones that uses hierarchical gestures was developed. The design and the implementation of this application is described. The application supports basic functionality of a smartphone operating system: opening contacts, sending SMS, and call. This application was evaluated in a controlled study in a laboratory ($n = 14$). The underlying motivations of learnability, improved efficiency, and improved satisfaction were supported by that study. Improvements of the application were developed based on the results of the evaluation and implemented in the application. An additional field study was carried out with an improved application to find out whether it would be used in practice. The field study proceeded successfully and indicated the validity of the hypothesis. Furthermore, the critical parts of the evaluated application turned out to be robust. However, bugs hampered the users' interaction with the system and the population of the study was small ($n = 3$). I propose revising the application again and putting it in another field study with a bigger number of participants.

Preface

I had the pleasure to meet a lot of exceptional people in the last two years that supported me in doing what resulted in this vast accumulation of paper. In advance, my deepest apologies to the ones that should have been part of the following list.

First of all, thanks to my three supervisors: **Paul van der Vet, Dirk Heylen,** and **Dennis Reidsma.** Even it was difficult, they managed to have me focus on one idea and ensured the successful completion of this project. I want to additionally thank my main supervisor **Paul.** It was a true pleasure to benefit from the knowledge, and enjoy the humor of such a nerdy and warm-hearted man.

I want to thank **Rafael Gieschke** for spending countless nights—including the one in which this was written—with me. Without your company, your spiritual and material advice, I would have slept a lot more, but I would also not have gotten that far.

Thanks to **Steven Gerritsen** and **Gilberto Sepulveda Bradford** who accompanied me in the last two years, gave feedback for this project, bravely endured working with me in projects, and drank countless beers and prepared a lot of delicious meals for me. It would have been a lot less fun without you.

Most importantly:

Thanks to my parents and my brother and sisters for all their love and support. I am thankful to have you.

⋆ ⋆ ⋆

Now, after the nice words, let us turn to the serious world of science—Dear reader, I wish you a fun and instructive reading!

Contents

Chapter 1

Introduction

> People don't really burst into flames if they use their cell phones during an eclipse.
>
> —Dilbert

This report introduces hierarchical gestures, a technique to build a gesture set. Hierarchical gestures are built by combining primitives. A primitive is an atomic, non-reducible element. Vector graphics, for example, are composed of geometrical primitives: amongst others circles, lines, and rectangles. Hierarchical gestures apply this principle to the operating system of a smartphone. In the context of a smartphone operating system the primitives are commands, like *Call* or *Send,* or objects like a contact. A gesture is assigned to each primitive. These gestures are concatenated to assemble a command. To invoke the command *Call Frieder at home,* for example, the gestures that are associated to the primitives *Call, Frieder,* and *at home* are concatenated. The complete gesture is structured as a command of common command language, like the ones that are used to control UNIX or DOS. The command consists of an action, *Call,* an object, *Frieder,* and a modifier, *at home*.

A gesture will be understood as a conscious movement of the body that conveys information. In this report, this definition will be restricted to gestures that can be carried out with one finger on the touchscreen of a touchscreen device. This report targets touchscreen devices that comply with *Android 4.0 Compatibility Definition* (2012). The screen resolution had to be between 320×480 and 480×800. Furthermore, only smartphones were targeted. Wikipedia (2012b) gives a definition of a smartphone and a summary of existing smartphones.

The remainder of this chapter outlines the aims of the project (Section 1.1), provides its contributions (Section 1.2), sketches its motivations (Section 1.3), and summarizes the structure of this report (Section 1.4).

1.1 Aims

The aim of this project was to investigate whether hierarchical gestures are a promising interaction technique for touchscreen devices. This aim was split into two sub-aims: whether hierarchical gestures are usable and whether they can be implemented in a robust and practical application. The first aim was addressed by evaluating the usability of hierarchical gestures and comparing it to the standard interface of the Android operating system. Usability was understood as being composed of efficiency, satisfaction and learnability. To address the second aim, an uncontrolled field study, using an application that was designed and implemented in this project, was carried out. This study should assess whether hierarchical gestures can be integrated in a robust and practical application and whether such an application has the potential to be used in practice.

1.2 Contributions

This projects proposes hierarchical gestures, a gesture set that follows a hierarchic structure. Several gestures for touchscreen devices exist and are implemented in many applications (e.g., slide to unlock or pinch to zoom). To the best of my knowledge, no gesture set that arranges the gestures in an hierarchic structure has been presented so far. Furthermore, the design and the implementation of an application that implements hierarchical gestures for Android is presented. The conduction and the results of a controlled lab study and of an uncontrolled field study are reported and discussed.

1.3 Motivation

This section introduces the motivations of hierarchical gestures. Gestures are shortcuts that speed up the interaction with a system since they allow one to carry out a complex task with one gesture. Thus, they are often faster than a menu-based interface. Hierarchical gestures are expected to be easier to learn than gestures that lack this structure. This was believed because hierarchical gestures offer a consistent and meaningful structure. Shneiderman, Plaisant, Cohen, and Jacobs (2010) report that hierarchic structures improve the learnability of command languages. I expect that this is applicable since hierarchical gestures follow a structure that is similar to the structure of a command language. Another advantage of gestural interfaces is that they are fun to use, as for instance, Ecker, Broy, Butz, and De Luca (2009); Bach, Jæger, Skov, and Thomassen (2008) report. This promises higher subjective satisfaction compared to a menu-based interface.

Gestures leverage one-handed interaction, since they can be carried out with the thumb; if the size of the device allows to do so. The majority of the

participants of a survey ($n = 229$) by Karlson, Bederson, and Contreras-Vidal (1998) reported that they would prefer to interact with their mobile devices entirely one-handed if their devices would allow them to do so. Furthermore, the hierarchical structure allows to support more function with the same amount of gestures than a linearly arranged gesture set does.

Gesture-based interfaces support eyes-free interaction. This capability is beneficial when interacting with the device in public where the environment demands visual attention. Bragdon, Nelson, Li, and Hinckley (2011) report ($n = 15$) that using gestures requires 30-times less eye contact than using soft-buttons for the same task.

Summarizing, hierarchical gestures have the potential to be a usable interaction technique since they are efficient, learnable, and promise subjective satisfaction. Furthermore, they are beneficial in mobile interactions since they leverage one-handed and eyes-free interaction.

1.4 Structure of the report

Chapter 2 provides an introduction into hierarchical gestures. The underlying ideas will be introduced, applied to smartphone operating systems—the use case of this project—and finally motivated. Chapter 3 reviews related work to motivate this project and to position it in existing research. In Chapter 4 the design of the system that implements hierarchical gestures is discussed. Its first section describes the functionality that the application supports and introduces the feedback mechanisms. Following, I explain the choice of the gestures that the application uses. Finally the technical realization is explained by discussing the choice of the gesture recognition algorithm and explaining the structure and the responsibilities of the components of the application. The evaluation procedure and the results are described in Chapter 5. This chapter first describes a controlled lab study, discusses its results, and the adaptations to the application that were motivated by it. Following, an uncontrolled field study is described. The conclusion of this report is found in Chapter 6 and includes a discussion of the methodology and closes with a summary and an outlook.

Chapter 2

Hierarchical Gestures

In this chapter I describe hierarchical gestures in detail. In Section 2.1 the concept of hierarchical gestures is presented, related to command languages, and illustrated by an example. Section 2.2 introduces how hierarchical gestures can be used to support tasks of the use case of this project: smartphone operating systems. In Section 2.3 I discuss the motivations underlying the concept of hierarchical gestures.

2.1 Concept

A hierarchical gesture is structured like a command.[1] A command is a concatenation of primitives. For example the Unix-command to delete all files on a computer is `rm -r /`. This command consists of three primitives: an action—remove represented by the textual command `rm`—a target—the root folder represented by the textual command `/`—and a modifier—a recursive deletion represented by the textual command `r`. The whole command is a concatenation of three primitives. Hierarchical gestures replace the textual commands that are assigned to the primitives, for example `rm,` with a gesture.

The primitives of a sample command language are shown in Table 2.1. The first column contains the actions, the second column the modifiers, and the third one the targets. To express the possible commands as hierarchical gestures, each primitive—which is associated with a textual command in a text-based command language—is associated with a gesture. The user chains these gestures to build a command. The gestures for the commands *Create* and *File* are shown in the left and the right part of Figure 2.1. Note that the shape of these gestures is artificial. To invoke the command *Create File* both gestures are executed sequentially as sketched in Figure 2.2 (the black gestures is executed first, follower by the red one). Larger commands

[1] As a command I understand any simple command (IEEE, 2012) that complies with the syntax that is given by IEEE (2008). Note that I use a different terminology. Instead of *utility_name* I use *action,* instead of *operand* I use *target.*

Table 2.1: Sample command language

Action	Modifier	Target
View		
	Viewer A	
	Viewer B	
		File
Create		
		File

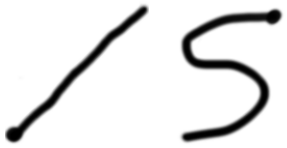


Figure 2.1: The gestures that are assigned to the primitives: *Create* and *File*; The gestures have to be started at the dots

that contain a modifier (e.g., *View Viewer A File*) would yield gestures with three, or more, parts.

Figure 2.2: The gesture to issue the command: *Create File*; The black gesture is carried out first starting from the dot

2.2 Application on smartphones

The concept of hierarchical gestures is not bound to a specific application. For almost any context a set of primitives that form a hierarchic structure can be constructed: to control a model train primitives could be *Train 2,* or *30% Speed;* to control the CD player in a car, primitives could be *CD,* or *Next Track.* I can hardly think of a domain whose commands cannot be decomposed into primitives.

In the following paragraphs I discuss what functions of a smartphone could be supported. Actions could include *Call, Send, Open, Delete,* or *Create.* The targets of these actions can be contacts or files, like MP3s or pictures. A modifier can determine how a file should be sent (e.g., via email or via SMS) or with what application it should be opened. These primitives

allow to express, amongst many others, the following commands: *Call at home Thomas, Delete thesis.tex,* or *Open highwaytohell.mp3,* or *Send Email to Thomas at Work*. Even if operations that involve files as targets are possible, the main application of the system is expected to be related to contacts.

The users are not expected to assign gestures to all contacts or all files on their devices and replace the standard interface with the system. I expect the system, similarly as Zeleznik and Miller (2006) their Fluid Inking system, to be used as a supplement that allows to accelerate frequently used commands. For instance a command like *Call Thomas* is expected to be supported with a gesture while a command like *Open unimportant file.txt* is not.

2.3 Motivation

Hierarchical gestures are motivated by the characteristics that are introduced in the following section. First, the structure of hierarchical gestures is expected to make them learnable. Second, gestures accelerate functions and are more efficient than the standard interface. Third, a gesture-based interface is expected to be more satisfying than a menu-based interface. Fourth, hierarchical gestures are more expressive than non-hierarchical gesture sets. Two motivations especially for mobile devices are that gestures support one-handed and eyes-free interaction. These aspects are discussed in detail in the following paragraphs.

Hierarchical gestures are expected to be easier to learn than gestures that do not follow this structure. This should be explained by an example. Gestures are mapped to commands when being used as shortcuts. One gesture, for instance, invokes the command *Open google.com,* and another one the command *Open facebook.com.* If a gesture for the command *Open facebook.com in new tab* should be defined, a completely new gesture would have to be introduced in a non-hierarchical gesture set. This is in spite of their similarity: they share the action—*open*—and the target—*facebook.com* and are only distinct in the modifier *in new tab*. Such gestures do not represent the structure of the commands with which they are associated. When extending such a vocabulary or learning a new gesture, more effort has to be invested since a completely new gesture has to be created or learned for each command. A hierarchical gesture, on the other hand, follows the structure of the command that it issues since it is a decomposition of its constituents. This link between command and shortcut is expected to facilitate the learning. A hierarchical gesture set arranges the gestures in a hierarchical structure. This structure allows to relate them to each other. Ormrod (2004, p. 224) states that such an internal organization of the concepts supports learning. Shneiderman et al. (2010, p.288–299) support this claim regarding the structure of command languages.

Learnability is supported since a system that uses hierarchical gestures complies with the principles of *predictability, generalizability,* and *consistency* that Dix, Finlay, Abowd, and Beale (2004, p. 261–265) present as means to support learnability. The user can *predict* the effect of commands by using knowledge of previous commands. This is since the system always reacts in the same way to the same commands independent from the context or the state of the application. The user can *generalize* from specific interactions to similar situations since the constituents of already mastered gestures can be reused to assemble new commands. The system is *consistent* since the same input always results in the same response of the system.

A gesture-based interface is expected to be more efficient than a menu-based interface since it allows one to carry out each supported task with a single gesture. Writing an email, for instance, requires five taps and several scrolling operations to open the application and select the recipient using Android's standard email application. A gesture allows to activate the same function with just one operation: drawing the gesture. The Android operating system, in contrast, does not offer a mechanism to accelerate frequently used functionality. Furthermore, as Roudaut, Bailly, Lecolinet, and Nigay (2009) claim, gestures are not only faster, they are also less error prone. Using a menu to select a contact from the, probably long, address book requires several scrolling and pointing tasks that demand more precision than the drawing of a gesture.

A gesture-based interface is expected to be more satisfying then a menu-based interface. Studies compared gesture-based interfaces to touch-based or physical interfaces and report superior scores for the satisfaction of the gesture-based interface (e.g., Ecker et al., 2009; Bach et al., 2008; Moyle & Cockburn, 2003). Saffer (2009, p. 17–20) concludes that gesture-based interfaces are more satisfying since they encourage playful exploration and fit our nature as physical creatures. They allow for more flexible interactions that do not rely on cumbersome hardware like a keyboard or a mouse.

Organizing gestures in a hierarchy allows us to express more commands with the same gesture set than without such an organization. If we assume a gesture set with 15 gestures, arranging them in a two-level hierarchy with three nodes on the first level allows to express at most $3 \times 12 = 36$ commands (in a real application this number might be lower since not all gestures can be combined with each other). A linearly arranged set would only yield the number of 15 commands.

A survey indicates that users prefer to interact with their mobile devices with one hand (Karlson et al., 1998). Gestures are an interaction technique that allows to accelerate functions and carry them out with one hand. The bigger expressivity of hierarchical gestures increases the functionality that can be invoked with one-hand compared to non-hierarchical gestures.

Furthermore, gestures are reported to be superior to buttons in distracting environments that demand visual attention (Bragdon et al., 2011). Similarly,

gestures appear to be be a viable interaction technique when the user's attentional capacities are demanded by another task, for instance driving. In a study ($n = 8$) about automotive interfaces by Alpern and Minardo (2003), users made less errors (i.e., violations of the traffic rules) when using a gesture-based interface for a radio than when using a radio with physical controls. The participants of this study also expressed a subjective preference for the gesture-based interface in qualitative interviews. Bach et al. (2008) present similar findings.

Chapter 3

Related Work

This chapter reviews related literature to provide the foundations for the project and to contextualize it in the existing research. The first section (Section 3.1) covers literature about how gesture vocabularies can be characterized and developed. Section 3.2 introduces gesture alphabets that are used for text-entry. The evaluations of these alphabets that are presented in this section introduce further characteristics of gestures. Following that, I discuss techniques that gesture-based applications use to distinguish gestures from input that is targeted towards the application (Section 3.3). Afterwards three applications that are related to the approach that hierarchical gestures pursue are described (Section 3.4). The final section summarizes systems that were developed to facilitate the learning of gestures (Section 3.5).

3.1 Characterization and design of gestures

In this section I discuss how gestures can be characterized and designed. Section 3.1.1 describes what aspects can be used to characterize, compare, and evaluate gestures. Section 3.1.2 stresses the benefits of user-defined gestures over gestures that were defined for, but not by, the user.

3.1.1 Characterization

This section introduces three techniques to characterize gestures. These techniques will be applied to create the gesture set that the application that is developed later on uses. Gestures can be characterized as being human-based or technology-based, depending on whether they are, for instance, easy to carry out or whether they are easy to recognize. Both categories are not mutually-exclusive and should be seen as two extremes of a continuum. A gesture can be classified by its shape to be either a mark-based gesture or a free-form gesture, depending on whether it only consists of rectilinear segments or not. The third characterization makes use of further features of

the characterized gesture, for instance whether the shape of the gesture is metaphorical or abstract.

M. Nielsen, Störring, Moeslund, and Granum (2004) discuss how to design hand-gestures and distinguish between human-based gestures and technology-based gestures. I believe that this approach is applicable to touchscreen-gestures since the technical side, recognizing a gesture, is comparable and since I think that the requirements for hand- and touchscreen-gestures are, probably not in terms of ergonomics, similar. Human-based gestures are gestures that satisfy common principles of usability that are presented by, amongst others, J. Nielsen (1992). They derive from these usability principles that human-based gestures have to be easy to perform, easy to remember, intuitive, metaphorical, and ergonomic. Similar characteristics of good gestures are also given by Höysniemi, Hämäläinen, Turkki, and Rouvi (2005). Technology-based gestures are gestures that are easy to recognize by a machine. This perspective needs to be considered since a distinction between gestures and, another common kind of shortcut, hotkeys is that gestures require technological effort to be recognized—a recognition algorithm has to compare an input gesture with a set of templates—while hotkeys do not. M. Nielsen et al. claim that functions are often forced on technology-based gestures without a semantic connection.

Bragdon et al. (2011) separate two kinds of gestures based on their shape: mark-based gestures that are only composed of rectilinear segments, and free-form gestures that do not follow this restriction.

A third characterizations is presented by Wobbrock, Morris, and Wilson (2009). They present a taxonomy that classifies hand-gestures according to form, for example whether the pose of the hand changes during the execution (not relevant for touchscreen gestures); nature, for example whether they are metaphorical or abstract; binding, for example whether the location where the gesture is carried out is relevant; and flow, whether the response of the system occurs during or after the gesture.

3.1.2 Design

Studies point out the advantage of human-based gesture vocabularies. Analogous to user-centered design practices, scholars claim that to find gestures that satisfy the human perspective the input has to be taken from the prospective users of the gestures. Wobbrock, Aung, Rothrock, and Myers (2005), for instance, propose to collect gestures for a function from users and to choose the one that is proposed most frequently. By doing so they claimed to have improved the quality of an existing gesture set. M. Nielsen et al. (2004) propose a similar process that asks users to assign gestures to functions, and uses input from the users to evaluate these gestures. The participants of a study of Morris, Wobbrock, and Wilson (2010) ($n = 22$) reported that they prefer gestures that were created by larger groups of

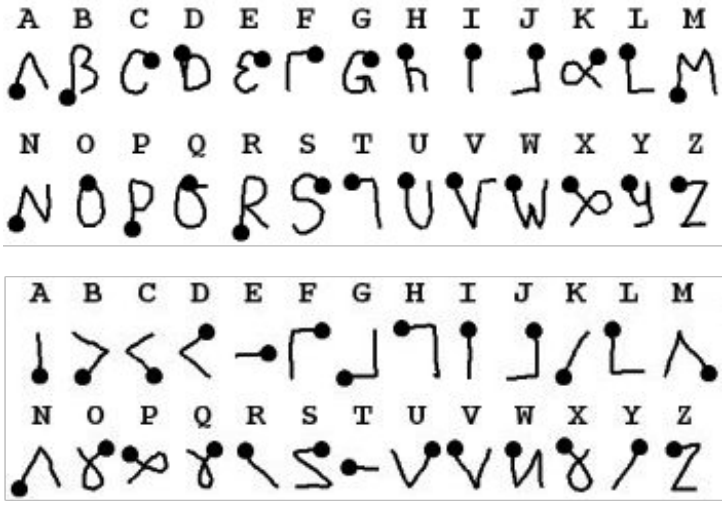


Figure 3.1: The characters of the Graffiti and Unistroke alphabet (MacKenzie & Zhang, 1997); The graffiti characters are shown on the top, the unistrokes characters are shown on the bottom

people. Thus, the application that is developed in this project allows the users to create and use their own gestures (see Section 4.2.2.2).

3.2 Alphabets

This section presents three gesture alphabets. First I discuss two common gesture alphabets Unistrokes and Graffiti. I also introduce a third gesture alphabet that pursues a different approach: Jot. The gestures that were used in the application that was developed in this project were taken from these alphabets (see Section 4.2).

Two single-stroke alphabets appear frequently in research: unistrokes, which was introduced by Goldberg and Richardson (1993), and Graffiti, which is used by Palm in their PDAs. Both alphabets are shown in Figure 3.1. Most unistroke characters are mark-based gestures that are composed of five different strokes whose orientation is varied to create the gestures that represent the letters of the latin alphabet. Strokes that consist of a straight line are assigned to the most common letters. Thus, there is no metaphorical mapping between the gestures and the letters that they represent. Most characters of Graffiti are designed to mimic the letters of the latin alphabet. Both alphabets mix mark-based gestures with free-form gestures. Graffiti contains more free-form gestures, unistrokes contains more mark-based gestures.

Figure 3.2: The characters of the alphabet Jot (Sears & Arora, 2002)

A study of Bragdon et al. (2011) ($n = 15$) evaluates mark-based gestures and free-form gestures. They suggest that mark-based gestures, and thus the unistrokes alphabet, are superior in terms of speed and accuracy. MacKenzie and Zhang (1997) report ($n = 25$) that Graffiti allows for an accuracy of 81.8% after one minute of studying the gestures and 95.8% after five minutes. A study by Castellucci and MacKenzie (2008) compares the usability of the Unistrokes and the Graffiti alphabet ($n = 10$). The shorter unistrokes yielded a lower duration per stroke and faster text entry. The mnemonic[1] gestures of Graffiti that resemble latin letters supported especially novice users. However, MacKenzie and Zhang (1997) also report that differences in handwriting lead to recognition errors that might neutralize this benefit of Graffiti. Sears and Arora (2002) compared Graffiti to a third gesture alphabet: Jot. This alphabet defines up to four templates for each letter to account for individual variations (see Figure 3.2). They report faster text entry of Jot compared to the Graffiti alphabet ($n = 31$). Furthermore, the different approach of Jot led to a lower error rate.

Another approach to deal with individual variations is to ensure that the gestures are sufficiently distinct from each other. If a gestures that is drawn sloppily gets misclassified as another gesture there is not enough "sloppiness space" in between (Goldberg & Richardson, 1993). Goldberg and Richardson (1993) tried to ensure that there is enough sloppiness space between the gesture of the unistrokes alphabet.

3.3 Mode-switching techniques

Gesture-based systems need a technique to distinguish between gestures and touch events that are directed towards the application. Zeleznik and Miller (2006) propose two techniques that they used in *Fluid Inking,* a system that adds gestures to a touchscreen-based note-taking application. The first technique is by starting with a prefix flick: a quickly drawn horizontal line. By drawing on top of such a flick the system knows that this is to be

[1] A mnemonic gestures is a gestures whose shape "[...] aids information retention" (Wikipedia, 2012a).

interpreted as a gesture. The second technique is by terminal punctuation. This can be done by pausing at the end of the gesture, by tapping, or by clicking a hardware button. They do not report negative comments from their users about the extra effort for either technique. Furthermore, users quickly learned and automated the mode-switching techniques.

Another mode-switching technique is proposed by Roth and Turner (2009) in a system called *Bezel Swipe.* In their project, gestures must begin within a marked area at the edge of the screen. Bragdon et al. (2011) report an evaluation of mode-switching techniques. They conclude that the mode-switching technique of Bezel Swipe is superior to using a hardware button. The application that is developed in this project uses a mode-switching technique that is similar to Bezel Swipe.

3.4 Related applications

Applications often use gestures as shortcuts. The purpose of a shortcut is to speed up interactions by providing a way to activate a command that is quicker than the standard interaction technique. A well-known type of shortcut is a hotkey. Appert and Zhai (2009) compared two types of shortcuts in a study: gestures and hotkeys. They investigated how users form a mapping between a gesture and a command and compared this to how fast this is the case with a hotkey ($n = 60$, in two sessions on different days). Their study reports superior results for gestures, regarding frequency of use, speed, recall and error rate. After two days participants still recalled on average twelve gestures but only four hotkeys. Participants of their study claimed that gestures are superior since they are easier to associate with real-world concepts.

In this section I discuss two comparable applications: *Gesture Search* and gesture-based menus. Gesture Search also accelerate the access to functionality using gestures. However, this application uses a different gesture set. As a second type of application I discuss gesture-based menus. This part focusses on marking menus and on projects that are advancements of marking menus. These projects are discussed since marking menus also combine gestures to a compound gesture. Besides, the section mentions two efforts that point out the increased satisfaction of gesture-based interfaces.

3.4.1 Gesture Search

Gesture Search is an application that accelerates access to applications and data on mobile devices (Li, 2010a). This is done by an incremental alphabetical search using gestures as input. The ordering of the search results is refined over time based on the user's past selections.

The approach that Gesture Search pursues is index-driven since the data is selected alphabetically. Hierarchical gestures start from actions,

for example calling, that operate on objects, like contacts. Li claims that invoking gesture search as an icon on the home screen is inconvenient and slows down the search. In contrast to Li's approach, hierarchical gestures can be invoked from anywhere by starting from the gesture button. Gesture Search requires the user to select the item from a list after searching for it. This is not necessary with hierarchical gestures since the command is issued after recognizing the gesture. Thus, hierarchical gestures promise a higher benefit in efficiency than Gesture Search. However, they sacrifice the possibility to access all data and applications on the device.

A further difference between Gesture Search and hierarchical gestures that is relevant for the development and the evaluation of the application is that Gesture Search is a closed application. Thus, it cannot affect the interactions of the user with other applications. The application that was developed in this project to evaluate hierarchical gestures is integrated with the operating system and influences the user's interactions with the operating system and with other applications.

3.4.2 Menus

An early gesture-based application, *marking menus,* is reported by Kurtenbach and Buxton (1993). In a marking menu a command is invoked by either using a pie menu (described by Callahan, Hopkins, Weiser, & Shneiderman, 1988) that appears after tapping the screen at the location of the pen, or remembering the position of the target element of the pie menu and draw the stroke without waiting for the appearance of the menu. Non-hierarchic ($n = 36$) and hierarchic menus ($n = 12$) were analyzed and found to be feasible interaction techniques (Kurtenbach, Sellen, & Buxton, 1993; Kurtenbach & Buxton, 1993). Limitations in precision are reported if the size of a menu exceeds eight items. The response time increased linearly with increasing depth of the menu. The implementation of a non-hierarchical marking menu with the breadth of six in a real application to annotate speech data is reported by Kurtenbach and Buxton (1994). Over time the usage of the marking menus surpassed the usage of the menu ($n = 2$). Hierarchical gestures build on hierarchical marking-menus since both approaches combine primitive gestures into a whole. Differences between the approaches exist. In a marking the same stroke can have different meanings depending on its level in the menu. The primitives of hierarchical gestures, however, keep their meaning in all contexts. Furthermore, hierarchical gestures allow for more flexibility in the design of the gestures and are not bound to a specific interaction technique as marking menus are with pie menus.

An adaption of marking menus to the small screens of mobile devices is implemented in *Leaf Menus* (Roudaut et al., 2009). This project is based on non-hierarchic menus that arrange the items in a list. The marks now have a curved shape and are no longer all straight. If the menu does not fit on the

screen because it was activated near the border, it is mirrored. Hence, each item has four stroke shortcuts depending on where the gesture is started. An evaluation ($n = 8$) showed that Leaf Menus are faster and less error-prone than pointing tasks. In contrast to hierarchical gestures Leaf Menus only have one level. Furthermore, the fact that the menu is mirrored depending on where it is summoned and thus requires different marks impedes the possibilities for eyes-free interaction. which is not the case with hierarchical gestures.

Ecker et al. (2009) implemented *pieTouch,* a similar concept for an automotive interface. Their system uses hierarchical pie menus that control the communication and navigation systems of a car. An evaluation ($n = 16$) suggests that their system was faster and more attractive than a reference system that does not use gestures. Despite the fact that the users perceived the system to be less learnable, quantitative data indicated that the system was easier to learn as well.

Moyle and Cockburn (2003) added gestures to a browser to support the *Back* and *Forward* commands. They used a pie menu with two items. The results of a study ($n = 20$) show that using the gestures is about 18% faster, and more satisfying than using the back-button of the software.

3.5 Learning

Gesture-based interfaces have deficiencies in learnability compared to a menu-based interface. In a menu-based interface the items have to be only recognized since their name is present in the menu. This is not the case with most gesture-based interfaces where the gestures have to be remembered since no item of the menu is associated with them. In light of this shortcoming, several systems that aim at supporting the learning of gestures were developed. In the following I present a selection of them.

Kurtenbach, Moran, and Buxton (1994) describe *crib sheets* to reveal the gestures that are available in the current context. The crib sheet is invoked by tapping and holding a stylus on screen. As a result it displays the available commands and the associated gestures. They informally report experiences using the system but left a formal evaluation for further work.

OctoPocus (Bau & Mackay, 2008) offers continuous feedforward and feedback mechanisms. When starting to gesture, labeled templates of the available gestures are displayed as overlays at the cursor position. During the execution, gestures that are no longer possible given the part of the gesture that is already recognized are eliminated using feedback from the gesture recognizer. A within-participants evaluation ($n = 16$) in one session with a 16-item gesture set indicates that this system improves learning and executing gestures compared to a sheet that listed the available commands and the associated gestures.

ShadowGuides (Freeman, Benko, Morris, & Wigdor, 2009) addresses multitouch gestures for surface computing. This system separates a gesture into a registration pose, the position of the hands at the start of the gesture, and the movement. While seeking the registration pose, the system displays how the user is currently touching the screen and a list of possible registration poses of the available gestures. While the gesture is carried out, the user is guided by a version of the OctoPocus system that was adapted to the necessities of multitouch. Critical parts of the gestures are highlighted by annotations. A between-participants evaluation ($n = 22$) with a 15-item gesture set against a video-learning system shows positive results for ShadowGuides.

GesturePlay (Bragdon et al., 2010) uses positive reinforcement in a game-like setting to motivate gesture rehearsal. The gestures are mapped to mechanical objects like springs or wheels that have to be operated (i.e., pressed, turned, touched or moved). The system was evaluated in a between-participants evaluation ($n = 20$) with a 16-item gesture set. Bragdon et al. report an increased propensity to play with the system and rehearse the gestures. Recall was not significantly increased compared with a video-learning system.

GestureBar (Bragdon, Zeleznik, Williamson, Miller, & LaViola, 2009) integrates itself in common UI widgets, like ribbons, to enable a walk-up-and-use experience. This includes an annotated depiction of the gesture and a practice area. This system was evaluated against a crib sheet, and two variations of a crib sheet that displayed the additional information after hovering over the command name. A within-participants and a between-participants evaluation ($n = 24$, $n = 44$) with a 25-item gesture set suggests a preference for GestureBar.

Chapter 4

Design

Chapter 2 described and motivated the concept of hierarchical gestures. To evaluate this concept it was implemented in an Android application. The implementation of this application is described in the following chapter. First, I explain the functionality of the application, its feedback mechanisms, and how it was integrated with the interface of the operating system. The second part of this chapter gives an overview of its technical realization by describing the gesture recognition and the software architecture.

4.1 Application

This section describes the functionality of the application, how the users interact with it, and how the application communicates its state to the user. The first part discusses the functionality that the application supports (Section 4.1.1). Section 4.1.2 describes how the user interacts with the application and how the gestures can be carried out. In Section 4.1.3 the feedback mechanisms of the application are explained.

4.1.1 Functionality

The application supports basic functionality of a phone: calling a contact, sending an SMS to a contact, and opening a contact. Extensions of the functionality are possible and could add sending emails, or files or opening webpages (extensions are discussed in Sections 2.2 and 6.1.1.5).

The gestures for the actions (i.e., *Open, Call,* and *Send SMS*) are predefined (see Section 4.2.2.1). The gestures that are assigned to the contacts are defined by the user (see Section 4.2.2.2). The association between the gestures and the contacts can be configured.

4.1.2 Gesturing

Typically shortcuts are available at any time. A hotkey, for instance, can be used at almost any time when the application is running. Thus, users should have the possibility to carry out a gesture at any time. This was realized by placing a slightly transparent square-shaped button on the left side of the screen as depicted in Figure 4.1.

After touching this button the user can immediately start gesturing. The parts of a gesture can be separated in two ways: by pauses of the users' movement and by lifting the finger from the screen after a gesture. The gestures can be separated by pauses of the users' movement. A pause is assumed when the coordinate of the users' finger does not change by more than two pixels in x- or y-direction. The length of this pause is— relying on the advice of Hinckley, Baudisch, Ramos, and Guimbretiere (2005)—500ms. The command that corresponds to the recognized gestures is carried out when the user lifts his finger from the screen.

The gestures do not have to be drawn interconnected. A gap between the parts is necessary if one gesture ends near the edge of the screen and the following one would, if started at the endpoint of the preceding one, extend over the boundaries of the screen. In such a case the user may lift his finger from the screen and start the following gesture at another point. When he lifts his finger a timer is started. If no new gesture is started before this timer expires the gesture mode is deactivated and the command that corresponds to the recognized gestures is carried out.

This mode-switching technique separates gestures from input that is targeted towards the active application. Such a mode that is only active while the user consciously does an action—touching the screen—is called a spring-loaded mode or a quasimode. Raskin (2000) champions these modes to reduce mode errors. This mode-switching technique is an adaption of the technique that Roth and Turner (2009) used for their *bezel swipe* application. A study by Bragdon et al. (2011) concludes that this mode-switching technique is better than activating the gesture mode using a hardware button. Using a button to activate the gesture mode was not implemented since I expected it to be slower than the implemented technique.

Carrying out the gesture without lifting the finger from screen will be called carrying out the gesture *without a gap* in the remainder of this report. Carrying out the gesture with lifting the finger from the screen will be called carrying out the gesture *with a gap*.

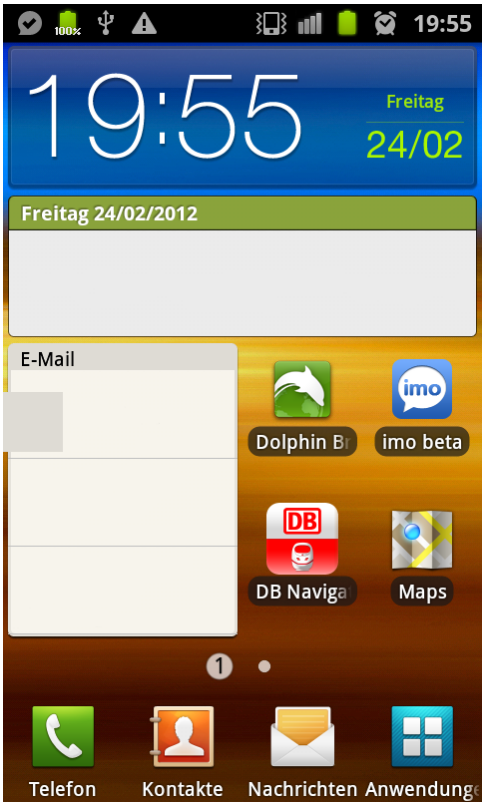


Figure 4.1: Screenshot of the application; The button on which the gestures have to begin is located at the center of the left edge of the screen

4.1.3 Feedback

That an application should give feedback about its current state is stated by usability heuristics (e.g., J. Nielsen, 2005; Norman, 2002). Saffer (2009, p. 20) advises that a system should indicate that it has "heard and understood any commands given to it."

The application offers tactile and visual—thus multimodal—feedback. Multimodal feedback appears to be more effective than unimodal feedback in a meta-study of 43 studies by Burke et al. (2006). The visual feedback is given by a trail of red dots that traces the gesture of the user (see Figure 4.2). The parts of the gesture that were already recognized are drawn in green. The names of the primitives are displayed on top of the screen as they are recognized.

Tactile feedback is given by a vibration of 35ms after a gesture was recognized. The duration of the vibration was set to mimic the tactile feedback that Android gives, for instance when using the on-screen keyboard. Augmenting soft-buttons, like the button on which the gestures start, with tactile feedback increased the performance of the participants of a study

($n = 13$) of Lee and Zhai (2009). Besides, tactile feedback can reduce visual distraction and thus supports eyes-free interaction. This is warranted by a study of Richter, Ecker, Deisler, and Butz (2010) that compares a system with tactile feedback to an equivalent system without tactile feedback. Richter et al. report that tactile feedback reduced the error rate and increased the driving performance of their participants.

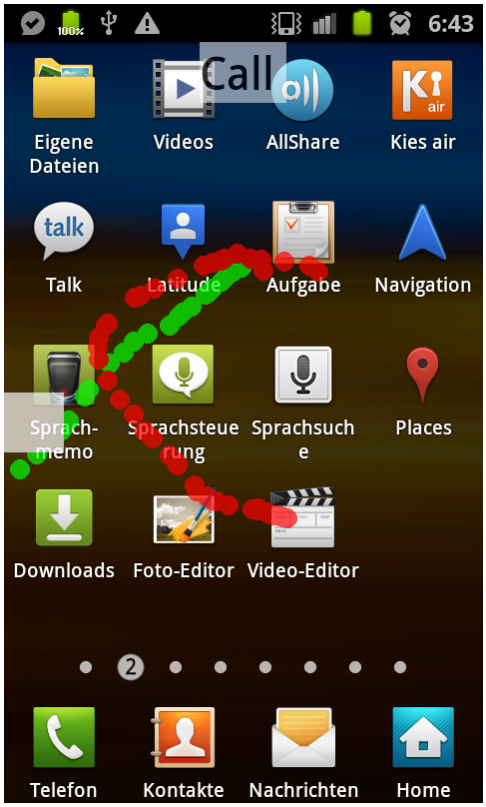


Figure 4.2: The already recognized parts of the gesture are shown in green; The primitives to which the recognized gestures are assigned are displayed on top of the screen

4.2 Gestures

After having discussed the functionality of the application in the preceding section, I now turn to the gestures that the application uses. First, I list and discuss requirements for a gesture set (Section 4.2.1). Section 4.2.2 uses these requirements to arrive at the gesture set that was used in the application.

4.2.1 Requirements

Two perspectives have to be considered when designing a gesture set: the human perspective that demands that gestures have to be drawable with little effort, and have to be easy to remember; and the technical perspective that demands that the gestures have to be recognizable by a computer. In the following section both perspectives will be discussed starting from the human perspective.

A study ($n = 14$) concludes that production time correlates with execution difficulty (Vatavu, Vogel, Casiez, & Grisoni, 2011). Thus, production time can serve as a metric to assess the quality of a gesture from the human perspective. A model to estimate the production time of a gesture is proposed by Isokoski (2001). His model decomposes a gesture into straight line segments that are counted to estimate the production time of the gesture. These lines are the minimal number of lines that make the gesture recognizable. However, Isokoski reports that his model cannot account for considerable variations in drawing time (30%) and that the decomposition into straight line segments is not necessarily unique. Nevertheless it is a simple technique to estimate the production time of a gesture, estimate its execution difficulty, and compare it with other gestures.

Another requirement is that a gesture needs to be ergonomic when being performed with the thumb or with the index finger. Considering one-handed interaction—using the thumb to control the device—is important since it was preferred by most participants of a survey of Karlson et al. (1998). Thumb-usage poses additional requirements since the thumb's movement range is constrained in other respects than the one of the index finger (Hirotaka, 2003). Hirotaka advises to place important elements next to the "home position" of the thumb: the place where it comes to rest naturally. These constraints can be supplemented by further ones that are reported by a study ($n = 20$) of Karlson et al. (1998). They found that diagonal movements of the thumb were significantly slower than straight movements, and that the region in the center of the screen was perceived to be more comfortable than the regions near the edges.

Another requirement for a gesture is intuitiveness. For the domain of in-air gestures, Höysniemi et al. (2005) name two aspects of intuitiveness: previous experience, and a natural mapping between a gesture and the reaction of the system. I have no quantitative evidence that this is applicable but it appears applicable to me. I do not expect that people have relevant previous experience with gestures to invoke functions. However, gestures differ in whether there is a natural mapping between the gesture and the reaction of the system. Such a natural mapping would be if, for instance, the gesture that is assigned to a contact looks like the first letter of its name.

A technical requirement is recognizability. The form of a gesture should allow the recognizer to reliably distinguish it from the other gestures of the

vocabulary. This is important since users often draw gestures quickly and sloppily. Sloppy drawing leads to recognition errors when gestures are too similar. To account for this, Goldberg and Richardson (1993) introduced the term of sloppiness space. If drawing a gesture quickly and sloppily leads to a wrong recognition of the gesture as another one there is not enough sloppiness space between the gestures. Unistrokes (see Figure 4.3), the gesture set that Goldberg and Richardson proposed, was developed with a focus on separating the gestures in sloppiness space to improve the accuracy of the recognition. Another way to deal with this problem is pursued by the gesture alphabet Jot that defines four examples per class to account for individual variation (MacKenzie & Soukoreff, 2002).

A requirement that is posed by the placement of the gesture button on the left side of the screen is that the first gesture of a command has to begin with a stroke to the top, to the bottom, or to the right.

4.2.2 Vocabulary

Two groups of gestures have to be defined. These are gestures for the actions that are discussed in Section 4.2.2.1 and the gestures for the contacts, the targets of the actions, that are discussed in Section 4.2.2.2.

4.2.2.1 Actions

The application supported three actions that had to be assigned with gestures: *Call, Open,* and *Send SMS.* The gestures for these actions were taken from the unistrokes alphabet (see Figure 4.3). This section motivates this choice.

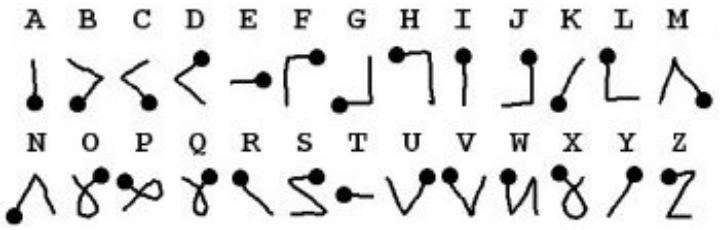


Figure 4.3: Unistrokes (MacKenzie & Zhang, 1997)

Two-types of gestures, mark-based and free-form gestures, can be distinguished. Mark-based gestures consist of axis-aligned or diagonal segments (they are used the in marking-menus of Kurtenbach and Buxton (1993)). Free-form gestures do not follow this restriction. A study by Bragdon et al. (2011) suggests that mark-based gestures can be carried out faster than free-form gestures and are superior in mobile environments as. An evaluation by Castellucci and MacKenzie (2008) for text-entry suggests that unistrokes allow faster text-entry while not being harder to learn than the letters of the Graffiti alphabet that resemble the Latin alphabet. Thus, gestures of the

unistrokes vocabulary were used for the actions since they are expected to be fast to carry out and easy to recognize. A possible disadvantage of this choice is that unistrokes do not carry a mnemonic meaning. The characters of the Graffiti alphabet do so by looking like letters of the latin alphabet. The evaluation will answer whether this downside is relevant. In the following I discuss the choice of the concrete gestures.

The gestures were chosen using the requirements that are defined in Section 4.2.1. First of all, gestures that start with a stroke to the left had to be excluded because of the placement of the gesture button on the left side of the screen (e.g., unistroke E or F)[1]. Second, I chose gestures that can be carried out quickly. This was done by restricting the choice to the gestures that consist of one or two straight line segments when being decomposed using the model of Isokoski (2001). Thus, the chosen gestures belong to the two groups with the lowest production time. This led to the exclusion of, for instance, the unistrokes P, and W.[2] The requirements of Karlson et al. (1998) regarding the movement range of the thumb were taken into account by excluding gestures that force the user to operate near the edges of the screen, when started from the gesture button (e.g., unistroke A or L)[3]. Besides, gestures that contain more diagonal movements (e.g., unistroke B or V) than the other gestures of the alphabet were excluded. The final gesture set was obtained from the remaining set (i.e., unistrokes G, H, K, R, and T) by excluding gestures that contain any diagonal movements (i.e., unistrokes R and K). It was decided to assign the shortest gesture, T, to the presumably most common action *Call*. The assignment between the actions and the gestures is given in Table 4.1.

Table 4.1: Assignment between action and unistroke character

	Character
Call	T
Send SMS	G
Open	H

The association between the actions and the gestures cannot be changed. A downside of these gestures is that there is no intended natural mapping between gestures and functions. This is done to ensure that the foundation of the application works consistently on all installations. However, users might still want to adapt the gestures for the actions according to their preferences. The evaluation should answer whether this is the case.

[1] Excluded were C, D, E, F, J, M, O, Q, S, U, and Y.
[2] Excluded were P, W, X, and Z.
[3] Excluded were A, L, I, and W.

4.2.2.2 Targets

The second group of gestures are the ones for the targets of the actions. These gestures are defined by the users. Using user-defined gesture is motivated by studies suggesting that user-defined gestures are superior over gestures that have been designed for but not by the user (e.g., Wobbrock et al., 2009; Morris et al., 2010).

However, doing so could introduce problems if the user chooses gestures that are hardly or not distinguishable by the recognizer. Whether this problem occurs will be answered by the evaluation.

4.3 Technical realization

This section discusses the technical realization of the system. The choice of the gesture recognition algorithm is discussed in Section 4.3.2. Section 4.3.3 summarizes the components and the structure of the application and how the gestures were stored internally. Section 4.3.3.3 summarizes the testing of the application.

4.3.1 Platform

Two major mobile platforms exist: Apple's iOS and Android that is developed by the Open Handset Alliance[4], a consortium of, amongst others, software companies, mobile operators, and handset manufacturers. The project was implemented for the Android operating system. This was done to, first of all, reuse existing Java knowledge. iOS applications are programmed in Objective C, a language with which I did not have previous experiences. Developing an Android application is free (installing an iOS application on your own device a registration and the payment of a fee is mandatory). A third reason for the choice of Android was that the implementation of the application would have been impossible on iOS since iOS does not allow changes of the user interface, like the addition of the gesture button (see Figure 4.1).

4.3.2 Gesture recognition

Accurate gesture recognition is crucial for the success of a system. Frankish, Hull, and Mirgan (1994) suggests a correlation between the accuracy of the gesture recognizer and how satisfying users perceive the system. Thus, the performance of a recognizer has to be high to convince a user to use and trust the system and to provide a satisfying experience. In the following I discuss the choice of the gesture recognition algorithm.

[4] `http://www.openhandsetalliance.com/`

A gesture recognizer assigns a class to a gesture. Various algorithms for that task have been proposed. I evaluated two kinds of algorithms: feature-based algorithms (e.g., Rubine, 1991; Anderson, Bailey, & Skubic, 2004; Gyung & Cho, 2006) and template-based algorithms (e.g., Li, 2010b; Wobbrock, Wilson, & Li, 2007). A feature-based algorithm extracts features, like the length of the gesture or the size of the bounding box, from examples and uses these features to classify the gesture. Template-based algorithms store templates of the classes and compare the input gesture point-by-point with these templates.

An evaluation showed that template-based algorithms, and in particular the *Protractor* algorithm of Li (2010b), are superior for the application that is developed in this project. Template-based algorithms are faster since the pre-processing is less expensive than the one that a feature-based algorithm requires. This is since only resampling and, depending on the algorithm, scaling, rotation, or vectorization of the gestures are necessary. The algorithm of Rubine (1991) requires matrix calculations in the pre-processing. Algorithms that use Hidden Markov Models (e.g., Anderson et al., 2004; Sezgin & Davis, 2005) entail an expensive recognition since the most likely path has to be calculated. This benefit in performance is, especially when used on mobile devices with limited and varying processing power and limited battery capacities, crucial. Additionally, the template-based algorithms that I evaluated are easier to implement and to maintain than their feature-based counterparts. Thus, a template-based algorithm was preferred.

A template-based algorithm is—for my gesture set—more robust. One for this is that the low precision of the touchscreens of smartphones renders features useless on which feature-based algorithms rely. For instance, the starting angle of the gesture is affected by jiggle that is caused by the precision of the touchscreen. Removing such features might be consider but may also lead to a feature set that is not sufficient to distinguish gestures. Template-based algorithms, however, are not affected by such artifacts. Besides, template-based algorithms are safer to expand. When extending a feature-based algorithm it is possible that the chosen features cannot distinguish the new gesture from the gestures in the vocabulary, even if the new gesture has a very different shape. This can also occur using a template-based algorithm, but the similarity of the gestures as the source of error is more visible for the user. The Protractor algorithm was finally chosen since it is, as Li (2010b) reports, faster than the otherwise similar template-based $1 Recognizer of Wobbrock et al..

In the beginning one template per class is used. Using more templates is reported to increase the accuracy of recognition (Li, 2010b). Evaluations can determine whether using more than one template is necessary. Possible reasons to do so are variations between and within users, for example differences in handwriting between users, and differences between carrying out a

gesture with the index finger and the thumb. The advantages of using more templates per class are demonstrated by the gesture alphabet Jot and are discussed in Section 3.2.

4.3.3 Implementation

This section describes the implementation of the application. The first section explains the components of the application, their responsibilities, and how they interact (Section 4.3.3.1). Following that I explain how the gestures and the primitives are stored (Section 4.3.3.2). The third section explains how the application was tested (Section 4.3.3.3)

4.3.3.1 Components

The touch events of the user are caught by an overlay that is placed on top of the operating system's interface. This overlay also displays the feedback and is controlled by the **Gesture Overlay Controller** that delegates the touch events to the Gesture Controller. The **Gesture Controller** receives the touch events, stores the points, and chunks the gesture into segments that are passed to the recognizer. A segment is created where the user's finger stopped for more than 500ms, or where the user lifted his finger from the screen. Furthermore, the Gesture Controller holds the templates. The templates are updated if the Gesture Controller is notified by the Gesture Persistence that a template was added or changed. The **Recognizer** recognizes the input gestures. Each gesture is resampled, using the method that Li (2010b) proposes, before it is processed by the recognizer. Resampling ensures that the gestures that are compared consist of an equal number of equidistant points. This is necessary since the number and the spacing of the points varies between gestures. A gesture that is carried out quickly, for instance, has less points than a gesture of equal shape that was carried out slower.

The **Command Maintainer** receives the results of the recognizer, transforms it into a command, and triggers the execution of the command. It sends the command to the system as an intent: an abstract specifications of a command (see *Intent |Android Developers* (2012) for details about intents). The **Query** component is the interface to the data on the device. It is used by the command maintainer to, for instance, retrieve the phone number of a contact when building a call command. The **Feedback Controller** generates the tactile and visual feedback.

Existing gestures can be changed and new gestures can be added. New or modified gestures are passed to the **Gesture Persistence** that stores them in a text file. Storing the gestures in files was considered the best approach since it requires less overhead and is more robust than using a database or serializing the gesture classes. When the Gesture Persistence receives a new gesture or a modified gesture it notifies the Gesture Controller to update its

templates. Figure 4.4 visualizes the components of the application and how they interact.

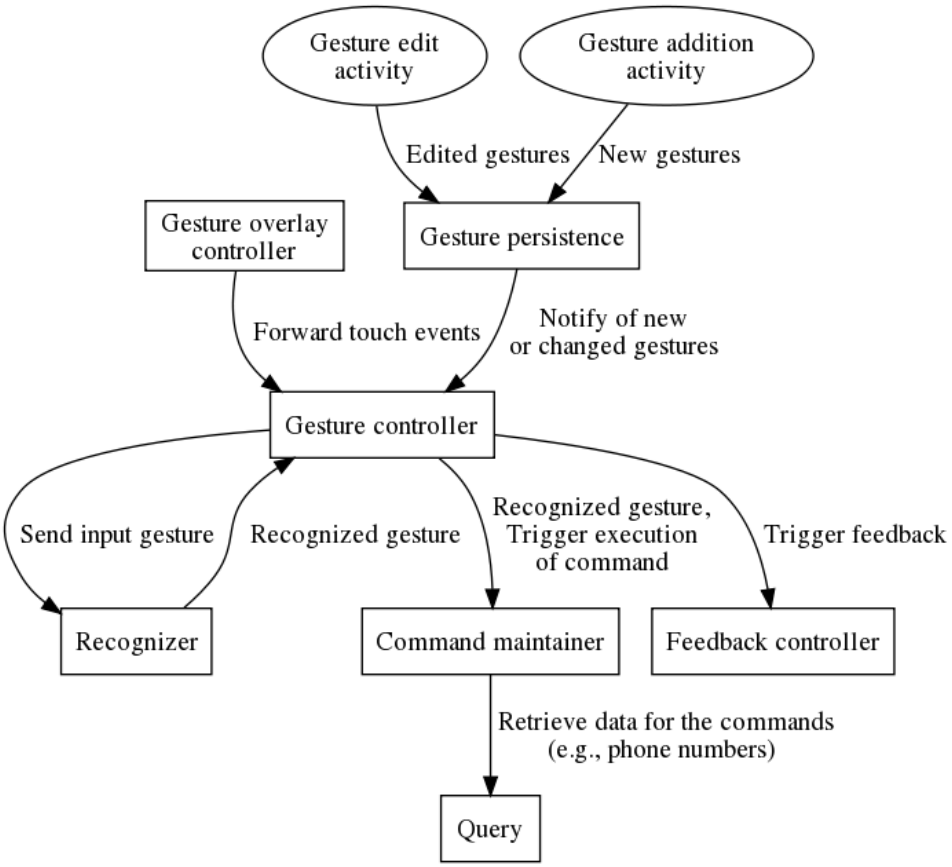


Figure 4.4: Components of the system

I tried to create a modular software architecture to facilitate the adaption of the system to different contexts. If the gestures originate from another source, for instance from a graphic tablet, only the **Gesture Overlay Controller** has to be adapted. A different recognition algorithm could be plugged in by changing the **Recognizer**. If the system is used in a closed application, the **Command Maintainer** can be adapted to no longer send intents to the system but call functions.

4.3.3.2 Primitives

A **Primitive** is identified by its name (e.g., *Call* or *Oliver*). A primitive can be activated by different interaction techniques. In this project a gesture class is associated to a primitive. Each **Gesture Class** has one **Gesture** as an examples and a name. A **Gesture** represents a template of a gesture class

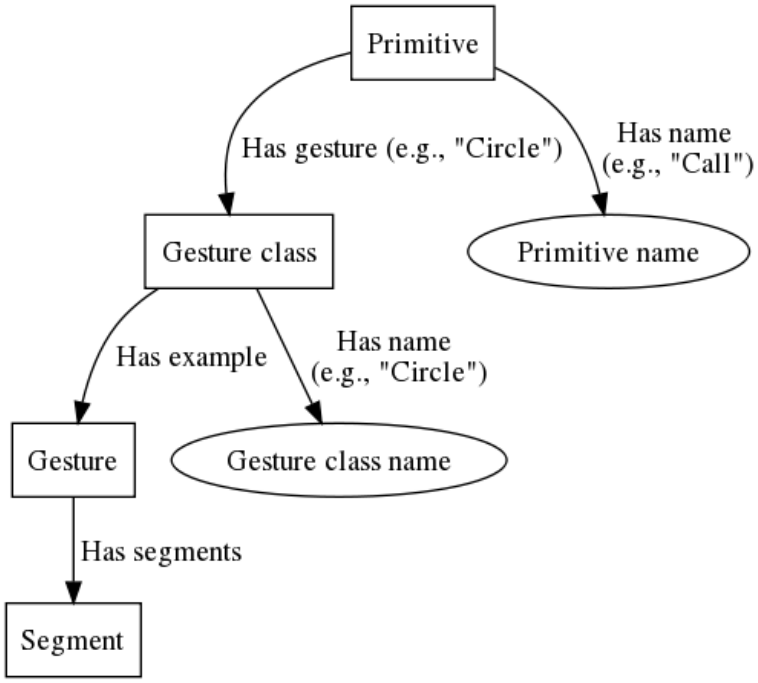


Figure 4.5: Association between a primitive and a gesture class

or an input gesture of the user. In the latter case a gesture is not assigned to a gesture class. The shape of a gesture is defined by a set of points. An input gesture of the user is separated into segments that represent the parts of the complete command. Figure 4.5 visualizes this structure.

This structure can be adapted if the system is controlled by another interaction technique. Primitives could then be associated with spoken or textual commands.

4.3.3.3 Testing

The devices that run Android are highly fragmented. They vastly differ in their technical capabilities (e.g., screen resolution or processing power) and in the version of Android that they run. To verify that the system works on smartphones with varying capabilities it was tested informally on a range of phones that included all supported screen resolutions (see Table 4.2). It was tested whether the system was responsive enough and whether the components that accessed the storage of the device worked as expected. All tests were successful with some expected performance losses on the weaker phones (i.e., Galaxy i7500, HTC Legend).

Table 4.2: Phones on which the application was tested

	Resolution	Processor	Android Version
Samsung Galaxy Nexus	720×1.280	2×1.2 GHz	4.0.3
Samsung Galaxy S2	480×800	2×1.2 GHz	2.3.4, 4.0.3
Samsung Galaxy Gio	320×480	800 MHz	2.3.3
Samsung Galaxy i7500	320×480	528 MHz	2.3.4
HTC Desire	480×800	1 GHz	2.3.3
HTC Legend	320×480	600 MHz	2.3.3
LG Optimus Speed	480×800	2×1 GHz	2.3.4

Chapter 5

Evaluation

That depends on what the phrase
"Fiddle with the numbers" means.

Pointy-haired Boss

Hierarchical gestures were implemented in an application that was described in the previous chapter. This application was evaluated to find out whether the motivations underlying the concept could be confirmed. In this chapter the evaluation will be designed and the results will be presented and discussed. The evaluation was carried out in two steps: a controlled pre-study that was followed by an uncontrolled field study. The reasons for the setup of the evaluation are discussed in Section 5.1. Section 5.2 describes the setting and the results of the pre-study followed by Section 5.3 that targets the field study.

5.1 Methodology

In this section I discuss the methodology that was used to evaluate the application. Most evaluations take place in either an natural or in artificial setting. A natural setting is in the "real world," whereas an artificial setting happens in a controlled environment created for the purpose of research (Kjeldskov & Graham, 2003). Since the characteristics of both settings complement each other, I decided to combine them. Hence, the system was evaluated in a field study that was preceded by a pre-study in the lab. The following paragraphs justify this choice.

A field study can yield more meaningful data than a lab study. This is since the application is evaluated in a natural environment: the real context of use. Maguire (2001) sees, concurring with other scholars, the context of use as a crucial factor for the usability of a system that has to be taken into account in evaluations. Besides, the fact that the user participates in an experiment is less evident since he is in his normal environment and, if the

field study is unsupervised, no moderator is present. These facts can elicit more natural behavior and, thus, yield more meaningful data.

However, this benefit comes by the cost of a drawback: the context in which the user interacts with the application is uncontrollable. Contextual factors, like whether the user is standing, sitting or walking, or whether he is located in a noisy or in a quiet environment, are crucial for the usability of a system. Since these influences are unknown it cannot be told whether the results of the field study might were caused by the conditions in which the user interacted with the system or by characteristics of the evaluated system. Not knowing about the context threatens the validity and the reliability of the results. This is especially the case when the evaluation has a low number of participants as in my study. More reliable and valid results are possible in a study with a high number of participants since the differences in the context and extreme cases average out. If the field study is unsupervised, finding out about the reasons for actions of the participants is harder than in a lab study where incidents can be discussed immediately. When technical problems arise, the participants must cope with them on their own since no assistance can be offered. Carrying out a lab study first should allow to spot the major issues in the application to minimize technical problems in the field study.

A field study allows to collect more data since the user can continuously interact with the evaluated system and does not have to come to a lab for an evaluation session. Extreme cases average out with a greater amount of data and conclusions can be drawn with greater statistical confidence.

Carrying out a field study is novel among related efforts. All but one of the systems that are described in Section 3.5 were evaluated in lab studies of one session. The prevalence of lab studies is also supported by a meta study of 102 publications by Kjeldskov and Graham (2003) that reports a bias towards artificial lab studies. They suggest to evaluate more mobile systems in the field. Other scholars could benefit from experiences regarding the setup and conduction of field studies and regarding the combination of a lab study with a field study.

To summarize, I opted for a field study that was supplemented by a controlled study in the lab since both study types can compensate the other's limitations and, thus, possibly work well as a combination.

5.2 Pre-study

The first part of the evaluation was a controlled pre-study in a lab. This study was meant to validate the motivations of hierarchical gestures and to expose technical and usability flaws of the application that could cause severe problems in the field study where the participants cannot receive assistance. In the following sections I discuss the aims of the evaluation, the plan of

the evaluation, the results that were obtained, and how the application was adapted after the study.

5.2.1 Aims

A main aim of the pre-study was to validate the hypotheses that are presented in Section 5.2.1.1. These hypotheses target three common aspects of many usability definitions (e.g., Shneiderman et al. (2010); J. Nielsen (1993)): efficiency, learnability, and satisfaction. An additional aim was to evaluate the quality of the application (Section 5.2.1.2). The quality of the application is understood as the performance of the gesture recognizer, the responsiveness of the application, the interface of the application, and its integration with the operating system. The findings of the evaluation should suggest adaptations before putting the application in an unsupervised field study.

5.2.1.1 Hypotheses

The pre-study should assess whether hierarchical gestures are a usable technique to control a touchscreen device and was focussed on three aspects of usability: efficiency (H1), learnability (H2, H3), and satisfaction (H4). Four hypotheses to cover these aspects were created and are introduced in the following. How they were addressed is discussed in Section 5.2.2.

- H1: **Hierarchical gestures are more efficient than the standard interface of a smartphone.** This hypothesis was included since a gain in efficiency could justify the effort to learn hierarchical gestures. This hypothesis was expected to be valid since drawing a gesture to invoke a command is faster than traversing the menu hierarchy by pushing buttons. Efficiency is construed as the ratio between invested effort and progress. Invested effort is construed as time-on-task and progress as the successful activation of a command.

- H2: **Hierarchical gestures can be learned by novice users.** Because of their consistent and logical structure it was expected that hierarchical gestures can be mastered quickly by novice users (see Section 2.3). The learnability of hierarchical gestures was evaluated in relation to the standard interface of the Android operating system. This hypothesis was validated by comparing time-on-task between hierarchical gestures and the standard interface over three trials and using the results of two questionnaires.

- H3: **Hierarchical gestures allow constructing untaught gestures.** When building the gesture for a command, knowledge of the structure and the constituents of previously learned gestures can be generalized and reused. This is expected to decrease the steepness of

the learning curve over time since more and more knowledge that can be reused is gained. The structure of hierarchical gesture that is one of its core motivations should be validated by this hypothesis.

- H4: **Hierarchical gestures are satisfying.** Studies report that gesture-based interfaces are more satisfying than interfaces that support the same functionality but do not use gestures (see Section 2.3). Satisfaction is important since a user might abandon the application if he perceives it as less satisfying than its alternatives.

5.2.1.2 Quality of the application

Another aim of the pre-study was to assess whether the application was reliable enough to be used in the unsupervised field study that was carried out after the pre-study. This, first of all, targeted the accuracy of the gesture recognizer—a crucial component of any gesture-based application—that had to be tested by real users. A second aspect was the responsiveness of the application. First of all, the internal responsiveness of the application, which is for instance influenced by the speed of the gesture recognizer, had to be tested. Another aspect is the speed in which the operating system carried out the commands that were triggered by gestures. A third aspect of the quality of the application was whether the interface, for instance the size of the gesture button or the configuration utility, caused usability problems.

5.2.2 Methodology

This section discusses how the aims that were presented in the previous section were pursued and prepares the detailed elaboration of the test procedure in Section 5.2.3. The experiment was carried out with a repeated-measure, within-subject design with the interaction technique as the independent variable and time-on-task as the dependent variable (see Section 5.2.2.1 for the discussion of the setup of the study).

The study pursued four aims: the measurement of efficiency is explained in Section 5.2.2.2, the measurement of learnability in Section 5.2.2.3, and the measurement of satisfaction in Section 5.2.2.4. How the quality of the application was tested is explained in Section 5.2.2.5. The analysis of the data and what conclusions can be drawn is summarized in Section 5.2.2.6.

5.2.2.1 Experiment design

The pre-study was carried out using a repeated-measure, within-subjects design. The independent variable is the interaction technique that was either the application with hierarchical gestures or the standard interface of the Android modification Cyanogen Mod 2.3.4. The measured dependent variable is the time that the participants required to complete the tasks of the study.

Repeated-measures were done since learning effects that manifest themselves over a period of time should be observed. A within-subjects design was chosen since this design is suitable for comparisons (of hierarchical gestures and the standard interface), and since no learning effect between both techniques was expected. The within-subject design is not biased by individual differences in learning patterns as Bau and Mackay (2008) claim. Furthermore, the experiment was carried out with a limited number of participants. In such a case a within-subject design yields a greater amount of data since every participant tries the standard interface and hierarchical gestures.

The participants had to complete three trials of 18 tasks. This was done to average the measurements for the measurement of efficiency, and to see whether learning occurred between the trials by comparing the measurements. The order of tasks and the order of the interaction techniques was counterbalanced between participants (a random sequence was generated for each participant).

5.2.2.2 Efficiency

One hypothesis (H1) is that hierarchical gestures are more efficient in controlling a smartphone than its standard interface. Efficiency was measured using time-on-task, as for instance Tullis and Albert (2008) advise. Hence, the time to complete a task using hierarchical gestures and the standard interface was averaged over the three trials and compared. The users were also asked to fill in a questionnaire to provide their opinion of what interface they perceive to be more efficient.

Time-on-task was measured manually. This was done since it was impossible to measure time-on-task automatically. Although the application that is in the foreground can be determined, it is impossible to retrieve its state, for instance what contact it displays. Furthermore, the thinking time before the user starts to interact with the application can be incorporated easily in a manual measurement. The time measurements were rounded to full seconds to allow errors in the measurement to average out.

The explanatory power of this measurement of efficiency has limitations. This is since the users have little experience with hierarchical gestures. Besides, they used a smaller set of gestures than the one that is expected in a real application. Changes of efficiency that manifest after a longer time of use, probably some days, were not covered. Nevertheless, the measured efficiency of novice users could serve as a baseline since the efficiency of experienced users is expected to improve. Besides, if there is no significant gain in efficiency the system requires alterations.

5.2.2.3 Learnability

Two hypotheses (H2 and H3) targeted learnability. Learnability was assessed by comparing the time-on-task between three trials and by assessing whether the participants of the evaluation memorized the gestures. Grossman, Fitzmaurice, and Attar (2009) provide a taxonomy of learnability that I used as a framework. Their taxonomy includes initial performance, the improvement of performance over time, and whether a specific level of performance can be reached. In the following I explain how these aspects were addressed.

Initial performance is assessed by comparing the time-on-task of the first trial of hierarchical gestures and the standard interface. Whether participants improved their performance over time is assessed by a comparison of time-on-task for the three trials that the participants had to complete with a distractor, the filling out of a questionnaire, in between. A third aspect of the taxonomy of Grossman et al. is whether users achieve a specific performance level. I used the performance with the standard interface as the reference. This is answered by comparing the performance of all trials. The measurement of learnability is complemented by items in a questionnaire that asked users to compare the learnability of hierarchical gestures and the standard interface.

Another hypothesis (H3) is that users can construct untaught gestures. This aim was pursued by a questionnaire that asked to construct gestures that they were not taught initially, for instance a gestures with three primitives. This should test whether they were able to apply the knowledge about the structure of hierarchical gestures.

This account of learnability is limited since it focusses on initial learning and neglects that learning can also be construed as a function of experience as Grossman et al. (2009) summarizes. This limitation is due to the fact that the time between the trials was short. Learning effects that take place over a longer time period, even one day, are out of the scope of this measurement.

5.2.2.4 Satisfaction

It was hypothesized (H4) that hierarchical gestures are more satisfying than the standard interface. This is addressed by evaluating the satisfaction of hierarchical gestures, and by comparing the satisfaction of hierarchical gestures with that of the standard interface. Since satisfaction is a self-reported metric it was assessed by a questionnaire.

Many usability questionnaires exist (see for example the summary of Tullis and Albert (2008)). Most of them target software systems with several screens and thus contain items, for instance about screen design, that are not applicable to my study since the application had a rather limited user interface. Lund (2001) proposes a usability questionnaire that he claims to be domain-independent. The part of this questionnaire that targets satisfaction

was used in the pre-study. The application was also compared against the standard interface. This was done by letting the participants compare the satisfaction of both systems on a 5-point scale. Qualitative data of the users that was gathered in a post-test interview and during the tasks served as additional input for the assessment of satisfaction.

The explanatory power of the results is nevertheless limited. When the participants judged the systems, they had unequal experience with them since the test was their first encounter with hierarchical gestures and they had probably been using Android for years. Thus, their response could be biased because of the novelty of hierarchical gestures or because they are used to the standard interface.

5.2.2.5 Quality of the application

The quality of the application included the performance of the gesture recognizer, the responsiveness of the application, and how its interface was perceived. The performance of the gesture recognizer was measured as the recognition rate. A manual measurement was done since the users' intended gesture has to be known to determine if the recognizer or the user erred. This measurement was used to decide whether the users have to supply the application with individual templates or whether one set of pre-defined templates suffices to achieve a satisfactory recognition rate.

Dix et al. (2004, p. 272) define responsiveness as "the duration of time needed [...] to express state change to the user." Two kinds of responsiveness were considered. First, the time between a gesture of the users and the feedback of the application telling that it was recognized and, second, the time between the end of the gesture and when the completion of the command visible to the user. Another aspect is the stability of the response time, meaning whether the response time changes noticeably between trials. The responsiveness was measured qualitatively by observation.

5.2.2.6 Analysis

The following section explains how the data that was gathered in the study was analyzed to answer the questions that the study asked.

First, the difference between time-on-task for hierarchical gestures and the standard interface was investigated. Median scores were reported since spread results that are caused by outliers were expected. To test for significant differences between both measures the non-parametric Wilcoxon test was used. This test was used since a non-normal distribution of the results was expected because of the small sample size; moreover I expected outliers because the application was used by novices. To verify that the results are not normally distributed a Lilliefors test was used. It was expected that $H_0 : m_{HG} = m_{SI}$ could be rejected in favor of $H_1 : m_{HG} > m_{SI}$ with $\alpha < 0.05$

for the median values of time-on-task.[1] To assess learnability the time-on-task of the the three trials was compared with each other.

The results of the questionnaire were tested for significant differences from the neutral answer. This was done by testing whether $H_0 : m = 3$ could be rejected in favor of $H_1 : m \neq 3$ with $\alpha < 0.05$. Only items where this was the case were used to draw conclusions. The non-parametric Wilcoxon test is used since the number of responses per questions was only 14 and thus a non-normal distribution was expected. A Lilliefors test was used to validate that the values are not normally distributed. A calculation of the Cronbach's Alpha provided the internal consistency of the questionnaires. According to Kline (2000) I only drew conclusions when the score was higher than 0.6. For the third questionnaire the percentage of correct responses was reported.

5.2.3 Plan

This section develops the course of the evaluation. The evaluation began with a briefing (see Section 5.2.3.4) that introduced the purpose and the content of the experiment. Next, the participants were given a questionnaire that collects demographic information and asks about their experience with smartphones (see Section 5.2.3.5). After that the participants had to complete three trials using both interaction techniques (see Section 5.2.3.6). After the first trial the participants had to complete a questionnaire in which they assessed the satisfaction of hierarchical gestures (see Section 5.2.3.7). After the second trial they were presented with a questionnaire to compare hierarchical gestures with the standard interface regarding their usability (see Section 5.2.3.7). The participants were also asked to construct gestures that were not shown to the user in the beginning. After that they had to complete a third trial. At the end an informal interview was carried out. Figure 5.1 illustrates the course of the pre-study.

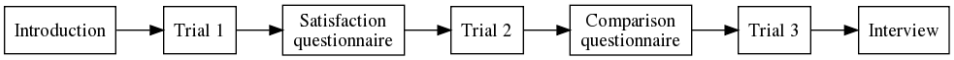


Figure 5.1: Course of the pre-study

5.2.3.1 Technical setup

All participants completed the evaluation with the same smartphone (LG Optimus Speed, Android 2.3.4, CyanogenMod). All participants were seated. The task descriptions were presented verbally. The questionnaires were filled out on paper. Time-on-task was measured manually using a stopwatch application and recorded on paper.

[1] HG denotes hierarchical gestures, SI denotes standard interface.

Using one phone for all tests instead of the participants' phone was meant to prevent the confounding influence of variations in the characteristics of the phones (e.g., in screen resolution or physical size). However, vendor-specific differences—vendors adapt the interface of Android in different ways—of the interface might influence the performance of users that are not accustomed to the interface that they had to use in the test. To address this issue the interface of CyanogenMod was used. The interface of CyanogenMod was thus expected to be equally unfamiliar to all participants and was expected to provide more equal conditions for all participants. The potential problems that might be introduced by having the participants use an uncommon user interface, for instance that the different interface distracts them from their tasks, have been accepted for the sake of a more repeatable study.

5.2.3.2 Privacy

An important aspect in the design of the study was to protect the privacy of the participants. This was done by presenting the questionnaires and recording the measurements on paper forms. Thus, no evaluation data was transferred to servers of a third party what would happen if an online tool, as it is provided by Google, would be used. Besides, no information about the time when a session was carried out was recorded. The data of the participants was identified by a randomly assigned number. The consent forms were not linked to these numbers. These means make the participants hardly identifiable for the experimenter, and for third parties.

5.2.3.3 Participants

The study was advertised by email and in social networks. No financial compensation for the participation was given. 14 participants took part in the study ($\mu_{age} = 24.5$; $\sigma_{age} = 2$; 5 female; all right-handed). A pilot study with one participant was carried out beforehand to test the setup of the study. The number of 14 participants is comparable with that of similar projects (cf., Appert & Zhai, 2009; Bragdon et al., 2011). The population is unbiased in reported smartphone experience ($\mu = 2$)[2] and almost unbiased in reported touchscreen experience ($\mu = 2.4$)[3]. 9 participants reported to own a smartphone.

5.2.3.4 Briefing

Informed consent is a crucial aspect of research (Loue, 2002). To achieve informed consent the participants need to be informed as completely as possible about the research in which they might take part. I used the

[2]Ranging from 1 (novice user) to 3 (advanced user).
[3]Ranging from 1 (novice user) to 3 (advanced user).

summary of Loue (2002, p. 117–118) as a guideline for the content of the briefing. In the briefing the participants were informed about the purpose and the content of the study, how the gathered data was used, how their privacy was maintained, that they could leave at any time, and who to contact in case of further questions. A consent form that repeats the briefing in written form was signed by the participants to document their agreement with the conditions of the study. The consent form is given in Section A.1.

5.2.3.5 Introduction

After the briefing a questionnaire that asks for demographic information and previous experience with touchscreen devices and smartphones was given to the participants. Then the experiment procedure was explained. Following the concept of hierarchical gestures and how the application is used are presented. After that the users create the gestures that they want to use for the contacts during the evaluation. These parts of the introduction are explained by the following paragraphs.

Questionnaire. A questionnaire was given to the participants after the briefing. It contains demographic questions and questions about the experience of the participants with touchscreen devices. The questionnaire is given in Section A.2.

Demographics. The answers to the demographic questions characterize the participants by age and gender. A question about the handedness of the participant is included to see whether the judgments between right-handed and left-handed participants differ. However, recruitment did not focus on getting samples from both groups. Both-handed participants were asked to enter the hand that they used to control their phones.

Smartphone experience. A set of questions tests the participants' experience with smartphones. Doing so allows to report whether the population is equally divided or biased in their previous experience. An unbiased population would support the generalizability of the results.

Introduction. First, the concept of hierarchical gestures was explained by an example. Two examples were given: in the first example the gesture is drawn without a gap, and in the second example with a gap (see Section 4.1.2 for the definition of the terms "with a gap" and ńńwithout a gap"). This demonstrates both ways to use the application. No instructions on which of the two options the participants should use or whether the participants should use their thumb or their index finger to gesture were given. The script of the introduction is given in Section A.4. The gestures for the actions

were presented on a sheet of paper (see Section A.3 for this sheet). After the introduction the participants were allowed to try out a sample gesture to get a feeling for the application. The experiment proceeded when the participants reported to feel comfortable with the application.

The 15 contacts that were used in the evaluation were pre-installed on the device. 15 contacts were used to make the tasks with the standard interface more difficult since then scrolling is necessary to find the targets. The names of the contacts were taken from a list of common Dutch, English, and German names (see Section A.5 for the list of all 15 names).[4] The following five names were randomly selected as targets for the evaluation.

- Emily
- Jack
- Eric
- James
- Oliver

Two names that start with E, and two names that start with J were deliberately chosen to force the user not to assign the gestures by the first letter of the name. The resulting total number of 15 gestures (3 actions $\times$ 5 contacts) is comparable with that used in similar evaluations (e.g., Bau & Mackay, 2008; Freeman et al., 2009; Grossman, Dragicevic, & Balakrishnan, 2007).

The participants were asked to create individual gestures for each contact. This was intended to support the memorization of the association between the contact and the gesture. Besides, in the final application the gestures for the targets will be user-defined as well. As a further measure to support the memorization the participants were asked to draw the gestures that they created on a sheet of paper (the sheet is given in Section A.3). This was also intended to compensate for the missing functionality of the application to display the available gestures and to which contact they are assigned. The participants were asked not to use gestures that consist of a straight line or resemble the gestures for the commands since the participant of the pilot study caused recognition errors by doing so. The participants were asked to try each gesture to practice them and to verify that the recognizer can distinguish the chosen gestures. If the recognizer could not distinguish the gestures the participants were asked to modify them.

5.2.3.6 Tasks

The experiment setup features three tasks: *Call, Open a contact,* and *Send SMS.* Every action was carried out nine times in both conditions (hierarchical gestures and standard interface) in each trial. Each participant carried out 3 Trials $\times$ 2 Conditions $\times$ 9 Repetitions = **54 Tasks**. In the first two trials

[4] See `http://en.wikipedia.org/wiki/List_of_most_popular_given_names`.

the participants were allowed to use the reference sheet with their gestures. The third trial tested whether the participants memorized the gestures and was carried out without the reference sheet.

Questions of the participants were not answered during the tasks. However, when defining and configuring the gestures the participants received assistance if necessary. This was done since the interface of this part of the application was simple and its possible shortcomings should not interfere with the actual evaluation of hierarchical gestures, for instance by causing frustration. Assistance was also offered when a participant was stuck because of technical errors of the application, or after the time expired that was allowed for the completion of a task.

In the following sections the three types of tasks, calling, sending SMS, and opening contacts, that the participants had to carry out are described. I will explain when the time measurement was started and stopped, when a task was judged as successfully completed, when a task was judged as failed, and how the tasks were introduced to the participants.

Call a contact. This task requires the participant to call a contact using a hierarchical gesture and the standard interface.

- **Introduction.** "Call Emily using the standard interface / a gesture."
- **Time-on-task measurement.** The time was started after the task description was read. The time was stopped when the call to the correct contact was initiated.
- **Success criterion.** The call to the correct contact was initiated.
- **Failure criteria.** The participant does not meet the success criterion after the third try when using hierarchical gestures or the participant does not meet the success criterion in one minute.

Send SMS. This task requires the participant to send an SMS to a contact using hierarchical gestures and the standard interface.

- **Introduction.** "Send an SMS to Oliver using the standard interface / a gesture."
- **Time-on-task measurement.** The time was started after the task description was read. The time was stopped when the SMS interface with the correct recipient was opened.
- **Success criterion.** The SMS interface with the correct recipient was opened.

- **Failure criteria.** The participant does not meet the success criterion with the third try when using hierarchical gestures or the participant does not meet the success criterion in one minute.

Open contact. This task requires the participant to open a contact using a hierarchical gesture and the standard interface.

- **Introduction.** "Open Oliver using the standard interface / a gesture."
- **Time-on-task measurement.** The time was started after the task description was read. The time was stopped when the correct contact was opened.
- **Success criterion.** The correct contact was opened.
- **Failure criteria.** The participant does not meet the success criterion with the third try when using hierarchical gestures or the participant does not meet the success criterion in one minute.

5.2.3.7 Questionnaire

A usability questionnaire was filled out after the first trial. The questions are taken from Lund (2001) and answered on a 5-point Likert scale. This questionnaire was meant to assess whether the users perceive the application to be satisfying (see Table A.1 in the appendix for the questionnaire). After the second trial, the participants had to fill out a questionnaire to compare the efficiency, learnability, and satisfaction of both systems on a 5-point scale. This questionnaire contains three questions that target efficiency, four questions that target satisfaction, and two questions that target learnability (see Table A.2 in the appendix for the questionnaire). The items of this questionnaire are taken from the questionnaire of Lund (2001). After filling out this questionnaire the participants were asked to carry out the tasks for a third time. A third questionnaire asked the participants to construct two gestures (see Section A.2 for the questionnaire). The first question asked to construct a three-part primitives to test whether users understand the relation between the structure of the command and the structure of the gesture. The second question asked to combine a new gesture with a known gesture.

5.2.4 Results

This section presents the results of the pre-study. In Section 5.2.4.1 I report the time-on-task measurements. The following Section 5.2.4.2 provides the results of the questionnaires. Technical characteristics of the application and its interface are discussed in Section 5.2.4.3. Section 5.2.4.4 discusses how

the participants constructed their gestures and how characteristics of the gestures affected their learnability.

5.2.4.1 Time on task

The results for the time-on-task of all participants are reported in the following. Figure 5.2 shows a histogram for the time-on-task using hierarchical gestures, Figure 5.3 shows a histogram for time-on-task using the standard interface.[5] The histograms, the mean, and the median values show that the time-on-task with hierarchical gestures was lower than with the standard interface ($m_{HG} = 4$, $m_{SI} = 6$). The difference of the medians was significant.

This difference in efficiency was most salient in the most complex task *Send SMS*. Figure 5.4 gives the histogram for hierarchical gestures for the *Send SMS* task and Figure 5.5 gives the histogram for the standard interface of the *Send SMS* task ($m_{HG} = 4$, $m_{SI} = 7$).

No significant difference in time-on-task between the first and the second, and the third trial could be observed with hierarchical gestures ($p = 0.7$; $m_{1,2} = 4$, $m_3 = 4$). Time-on-task between the first and the second trial with the standard interface decreased significantly ($m_1 = 7$, $m_2 = 5$). This was not the case regarding the difference between the second and the third trial ($m_2 = 5$, $m_3 = 5$).

[5] Completion times of more than 20s were cropped to 20s.

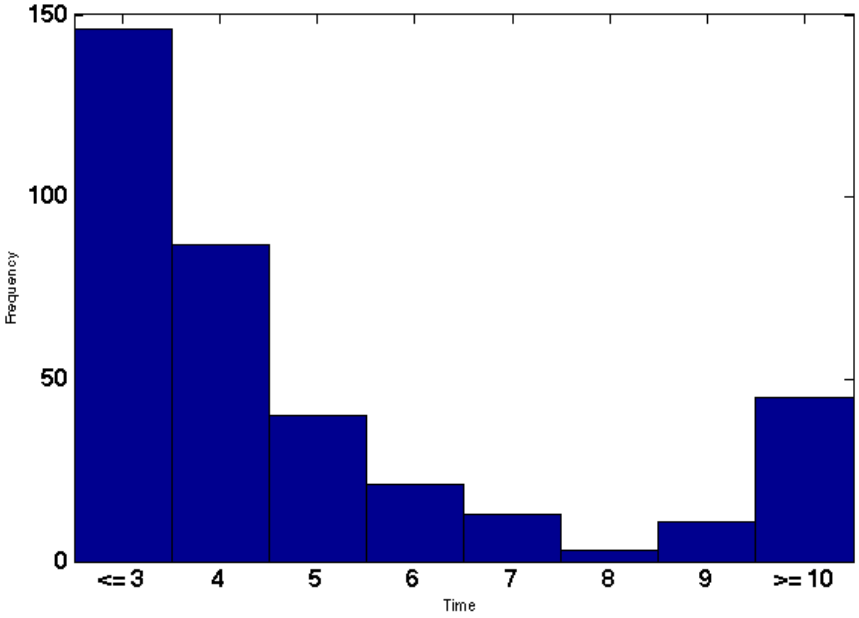


Figure 5.2: Histogram for all tasks with hierarchical gestures; The x-axis gives how long the users needed to invoke a function, the y-axis gives how many times the users needed the given time to invoke a function

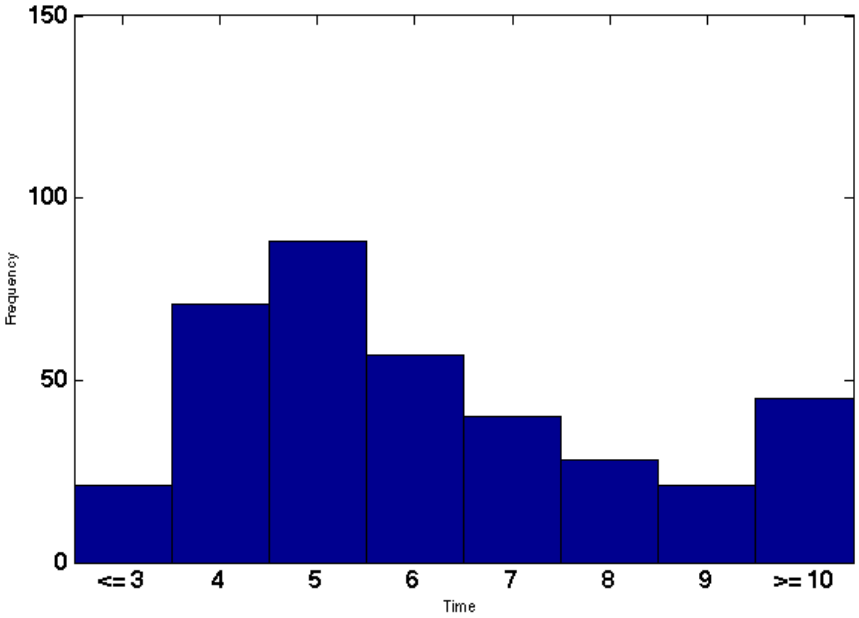


Figure 5.3: Histogram for all tasks with the standard interface

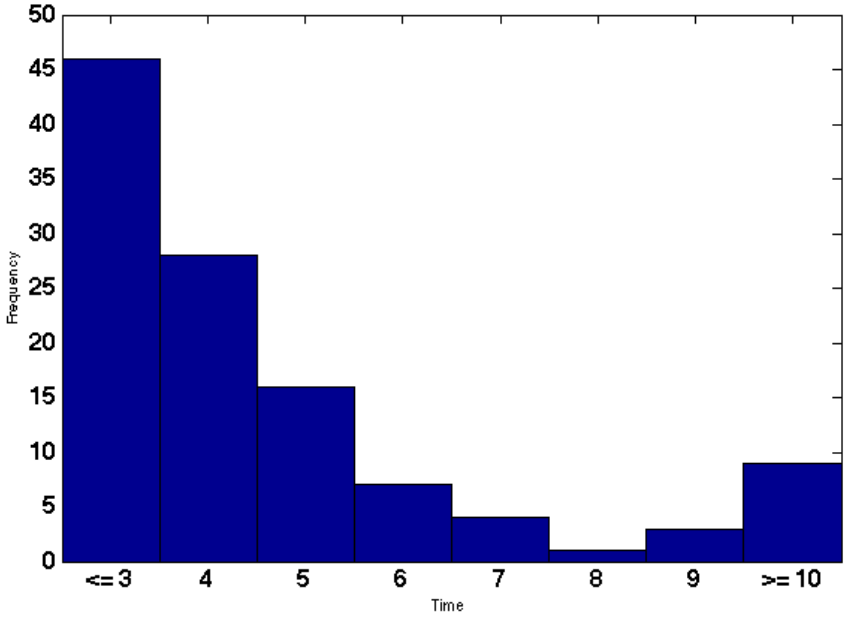


Figure 5.4: Histogram for *Send SMS* tasks with hierarchical gestures

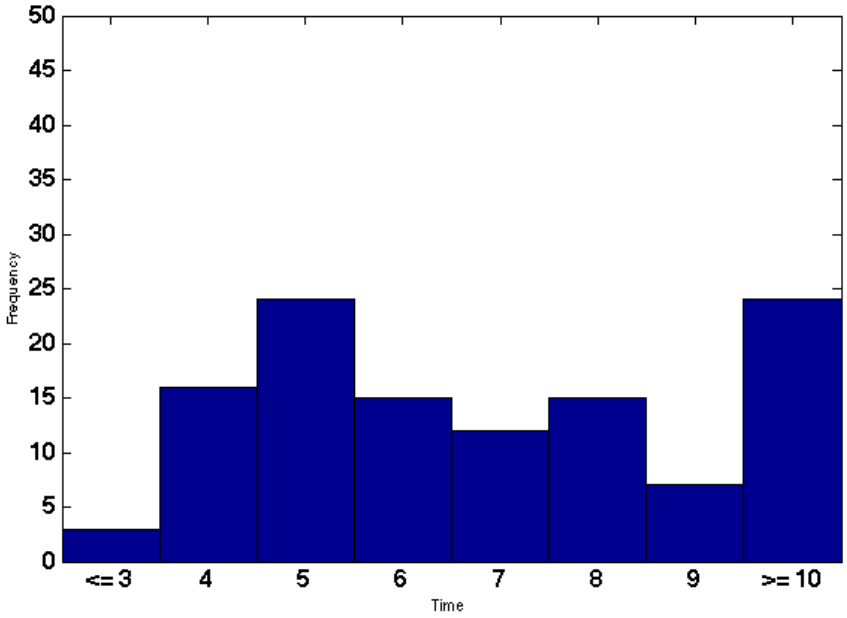


Figure 5.5: Histogram for *Send SMS* with the standard interface

5.2.4.2 Questionnaires

Table 5.1 presents the results of the first questionnaire (1, strongly disagree; 5, strongly agree). The questionnaire yields a significantly positive score for the satisfaction of the application (a median of 4 over all questions). The Cronbach's Alpha of 0.67 is not wholly satisfying, indicates deficiencies of the questionnaire but was, nevertheless, acceptable.

Table 5.1: Results of the first questionnaire; Items that did not achieve significancy are grayed

	Mean	SD	Median	p
I am satisfied with it.	4.3	0.8	4	$< .05$
I would recommend it to a friend.	4.1	0.5	4	$< .05$
It is fun to use.	4.3	1.1	5	$< .05$
It works the way I want it to work.	3.5	1	4	$< .05$
It is wonderful.	4	0.7	4	$< .05$
I feel I need to have it.	3.2	0.9	3	$> .05$
It is pleasant to use.	4.3	0.9	4	$< .05$
Total ($\alpha = 0.67$)	4	0.9	4	$< .05$

Table 5.2 presents the results of the second questionnaire that asked for a comparison of both systems (1, gesture-based system; 5, standard interface). Significantly positive results for efficiency and satisfaction were observed (median values of 2). The score for learnability was balanced (median value of 3). The Cronbach's Alpha is satisfying for two of the three groups (satisfaction and learnability). The items that targeted efficiencies scored only 0.55. A cause of this might be that the participants were unsure about how to understand the notions that were used in the questions. They were, for instance, uncertain how to construe efficiency or whether efficiency has the same meaning as being fast.

Table 5.3 presents the results of the third questionnaire that asks the users to construct untaught gestures. Both scores (69% correct responses) are surprisingly high considering the fact that the questions were answered on paper without any assistance or further instruction.

5.2.4.3 Application

The following section describes how the participants interacted with the application and how well the application performed technically. The first section describes how the participants carried out the gestures. Then I describe how the participants thought that they could use the application in practice. Following, I describe how the gesture recognizer performed. The next paragraph describes the responsiveness of the application and technical

Table 5.2: Results of the second questionnaire; Items that did not achieve significancy are grayed

	Mean	SD	Median	p
This system is more efficient.	2.1	1.1	2	$< .05$
This system is faster.	2.7	1.2	2.5	$> .05$
This system saves time.	2	0.9	2	$< .05$
Total ($\alpha = 0.55$)	2.3	1.1	2	$< .05$
This system is more fun.	2	1.3	1.5	$< .05$
I would prefer to use this system.	2.3	1.1	2	$< .05$
This system is more attractive.	2.2	0.9	2	$< .05$
Using this system is more pleasant.	2.5	1.2	2	$< .05$
Total ($\alpha = 0.74$)	2.3	1.1	2	$< .05$
This system is easy to learn.	2.5	1.3	2.5	$> .05$
I learned to use this system quickly.	2.6	1.2	3	$> .05$
Total ($\alpha = 0.95$)	2.6	1.2	3	$< .05$

Table 5.3: Results of the third questionnaire

	% Correct
Call Emily at home	69%
Send email to Oliver	69%

aspects that influenced it. In the closing paragraph I describe observations about the feedback mechanisms of the application.

Interaction. The application allows to carry out the gestures with a gap and without a gap. However, all participants carried out the gestures with a gap. Participants mentioned that doing the gesture without a gap would allow them to pause to think of the gesture for the second part. This is not possible when leaving a gap since the length of the gap must not exceed a fixed threshold.

After the user finished the first part of a gesture and lifted his finger from the screen a timer was started. If no new gesture is started while the timer is running the gesture mode is deactivated and the gesture is sent to the recognizer. This is sketched in Figure 5.6. If the participants were too slow in starting the second part of their gesture and exceeded the interval they were forced to start the gesture again since the application deactivated the gesture mode and sent the incomplete gesture to the recognizer.

Another timer detects when the user paused on the screen. This timer is necessary to allow carrying out a gesture without a gap and determines the end of the first gesture. This timer was reset when a move event was

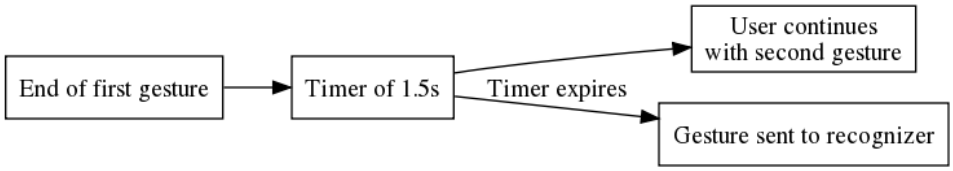


Figure 5.6: Within-gesture threshold

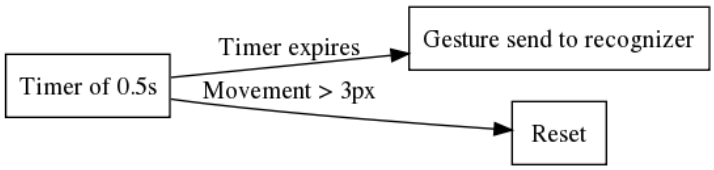


Figure 5.7: Within-gesture threshold

received and the new coordinate was at least three points in x- or y-direction away from the old coordinate. This is sketched in Figure 5.7. However, when users gestured slowly these resets were not triggered and the timer expired. Thus, the system considered the gesture to be complete and sent it to the recognizer.

Both issues had the negative effect that the participants felt pressured to gesture quickly and thus sloppily. Having to redraw a gesture caused frustration. This pressure was reported by the participants and was also visible in their interactions with the application.

The square on which the gestures had to start was sometimes missed. The participants did not wait for the feedback that the application gives to indicate that they hit the button, started the gesture and then discovered that the application did not recognize their gestures.

The addition and configuration of the gestures was cumbersome. It was necessary to first name the gesture and then associate the name with a contact. These three steps are technically necessary since the gesture class is created, named, and then assigned to a primitive (see the discussion in Section 4.3.3). However, from the perspective of a user it is not.

Practical usage. The motivation that gestures foster eyes-free and one-handed interaction could be supported. Participants recognized and appreciated the possibility to use the application for eyes-free interaction. This was justified by the fact that doing a gesture does not demand permanent visual contact with the device. Besides, the possibility for one-handed use was mentioned by the participants. Thus the application would be usable, for instance, when initiating a call while cycling.

Participants concurred that the application is usable to accelerate the access to about ten frequently used contacts. Learning gestures for a large

number of contacts would demand too much effort. Also, gestures for rarely used contacts would be forgotten quickly.

Gesture recognition. The gesture recognizer performed satisfactorily. Errors were almost always caused by sloppy gesturing. When the gestures that a participant chose were similar to other gestures, wrong recognitions of correctly drawn gestures occurred. This was problematic since the similarity between the gestures was not always obvious to the user. I observed that the participants mostly focussed on the shape of a gesture whereas aspects like the direction were neglected. A clockwise-drawn circle, for instance, was wrongly considered to be equivalent to a circle that was drawn counter-clockwise by the participant. For the participants it was not intuitive that both gestures are distinct. However, the recognizer considers the direction of the gesture and, thus, treats both circles distinctly. The participants seemed to regard a gesture as a static shape without directionality. This mismatch forced the users to adapt their model of the gestures to the one of the recognizer.

Using multiple templates per gesture class does not seem warranted by the results. However, the number of eight distinct gestures is low, compared with a realistic application in which more gestures can be expected. In an application with more gestures classes more templates per class might be necessary. Li (2010b) reported an error rate of 3% for a set of 16 gestures. Using two templates decreased the error rate to 1.5%. Thus, defining more templates might be required if the gesture set grows.

Responsiveness. When discussing the responsiveness of the application we have to differentiate between the responsiveness of the isolated application and the responsiveness of the application in its context: the Android operating system. The application itself was sufficiently responsive during the evaluation (less than one second). However, the requests of the application that trigger the actions of the operating system (e.g., call someone) have to be handled by the operating system. In most cases the time between carrying out the gesture and the invocation of the action by the operating system was sufficiently short (less than one second). However, some requests took about three seconds to be processed. A delay of this length interrupts the user's flow of thought as J. Nielsen (1993, 2009) suggests. Long response times even caused participants to redo a correct gesture since they inferred from the long response time that the application did not recognize their gesture. Summarizing, the response time of the application itself is low and stable while the response time of the application in the context of the operating system is not and sometimes exceeds three seconds. The efficiency gain of hierarchical gestures would be bigger than the one that was measured, when

the unstable response times of the operating system would not be included in the measurement.

Feedback. The participants did not comment about the feedback mechanisms of the application.

I observed that the visual feedback sometimes disappeared too quickly to be readable. However, this constitutes a tradeoff since no interactions with the standard-interface of the operating system are possible when the feedback is displayed. This is the case since the view that displays the feedback prevents the touch actions of the user from being passed to the operating system.

5.2.4.4 Gestures

This section presents the results of the evaluation that describe how the participants used and perceived the gestures. The first section discusses techniques that the participants used to construct the gestures. Next, observations that are related to the learnability of gestures are provided.

Gesture vocabularies. The gestures that the participants created were analyzed to discern techniques that they used to build gesture vocabularies. Three techniques that are discussed in the following were observed.

1. Build the gesture using letters of the contact's name. When contacts shared the starting letter, as was the case in the evaluation, the participants used gestures that combined the first two or three letters. The participants often used simplified or abstracted versions of the letters.

2. Use a mnemonic gesture that is not related to a letter. Examples are using the letter B if the contact had the same name as the participant's brother, or a heart if the contact had the same name as the participant's boyfriend or girlfriend.

3. Use artificial gestures. Gesture sets that were created with this technique used greek letters and geometrical shapes (e.g., triangles or rectangles). However, this technique cannot be distinguished from technique 2 since what appears artificial to the observer might be mnemonic for the user. Gesture vocabularies that were defined using this technique were distinguished from the others by the comments of the participants during the evaluation and in the post-test interview.

Participants combined these techniques to create their gestures. A circle, for instance, was frequently used for the contact Oliver while the remaining contacts were associated with gestures that were created using the other techniques.

Learnability. The participants had to memorize two kinds of gestures: the gestures for the commands and the gestures for the contacts. It was observed that the errors of the participants were mostly caused by forgetting the gesture for the contact and not by forgetting the gesture for the command. Most errors happened when the gestures were designed artificially (see Technique 3 that was described in the previous section).

Participants praised the possibility to associate the gestures for the contacts with concepts that they know. Some participants reported that they would have performed better if they knew persons with the names of the contacts that they were using in the evaluation. To increase the complexity of concepts that can be depicted with gestures, it was suggested to allow defining multi-stroke gestures.

Participants criticized that the gestures for the commands were fixed. It was suggested to make them more mnemonic and less artificial, for instance by using the first letter of the name of the command. However, some participants supported the motivations behind the gestures for the commands by calling them fast and robust. Nevertheless, as also reported by Appert and Zhai (2009), participants said they formed associations between these gestures and the commands. One participant, for instance, associated the gesture for opening with the handle of a door. These associations were not prompted or even intended by the design of the study.

5.2.5 Discussion

The following section discusses the results that were presented in the previous section, draws conclusions, and proposes adaptations to the application. In Section 5.2.5.1 the hypotheses that were addressed by the experiment are discussed. Adaptations to the application that are motivated by findings of the evaluation are proposed in Section 5.2.5.2.

5.2.5.1 Hypotheses

This section discusses the results of the study with a focus on the four hypotheses that were presented in Section 5.2.1.1. H1 (efficiency) and H4 (satisfaction) are supported by the results of the experiment. The experiment provided indications for the validity of H2 (learnability) and H3 (construction of new gestures). These hypotheses demand more investigation in subsequent studies.

H1. This hypothesis claims that using hierarchical gestures is more efficient than using the standard interface. The gathered data shows a significantly lower time-on-task (the median scores differed by two seconds) and, thus, indicates that this hypotheses is valid. As expected, the highest benefit is gained for commands that require a series of actions with the menu-based

interface. For the *Send SMS* task the median scores differ by three seconds. The results of the second questionnaire showed that the participants also perceived the gesture-based system to be more efficient. Thus, hierarchical gestures seem to be a promising technique to speed up interactions with smartphones.

Gestures are in general faster. This is since the maximum speed of a task is defined by the physical effort that is required to complete it. Sending an SMS, for instance, requires four tasks: Selecting the menu icon, clicking on "New Message," selecting the field to enter the recipient, and entering the recipient. The maximum level of efficiency is higher when doing a gesture since only one task is required: the physical motion to carry out the gesture.

The results of the pre-study indicate that gesture-based interfaces and hierarchical gestures are more efficient for several reasons. First, the task to carry out a gesture is claimed to be less error-prone than the precision-demanding pointing tasks of menu-based interactions (Roudaut et al., 2009). The number of contacts in the pre-study was with 15 low. The participants of the field study that was carried out after this study had 97 contacts in their address books (see Section 5.3.7.2). This increases the likelihood of error using the standard interface. The likelihood of error with hierarchical gestures is expected to be lower since the participants claimed that they would define about 10 gestures. Second, the efficiency of the menu-based interface is influenced by the state of the application, for instance a large number of contacts in the address book makes the scrolling operations require more time no matter how trained the user is.

The gain of efficiency with hierarchical gestures could be enhanced by targeting the communication between the application and the operating system. This causes a delay of up to three seconds and thus severely hampered the performance of the participants of the study.

H2. The time-on-task measurements indicate that using hierarchical gestures is significantly faster than using the standard interface. This result was achieved in a setting where little training with each technique was possible. One conclusion that is warranted by these results is that the initial learnability of hierarchical gestures is superior to the standard interface. This was indicated by the fact that time-on-task for hierarchical gestures is lower than time-on-task for the standard interface over all the trials. The second aspect. change of performance over time, cannot be answered since the median values of time-on-task did not vary significantly between the trials. The third aspect of the hypothesis, whether users achieve a specific performance level over time, is supported since time-on-task was lower over all trials. A study over a longer time period appears necessary to answer the hypothesis in a more complete and more sound fashion.

According to the results of the questionnaire, the users do not perceive the application to be significantly more learnable than the standard interface. Causes for this perception, which was in contrast to the time-on-task measurements, were indicated by the comments of the participants. The proposed causes spanned inherent deficiencies of gestural interfaces, for instance that they are not as self-revealing as menu-based interfaces where all options are arranged in a list and that gestures need training to be memorized and to practice their execution. This result was expected since users with previous smartphone experience are used to the standard interface, and since the time that was available for the users to learn the gestures was short (some gestures were only practiced four times before they had to be memorized in the third trial).

The result that users perceive the application as subjectively less learnable despite quantitative and qualitative results indicating the opposite, was similar to the one that Ecker et al. (2009) reports. Ecker et al. compared a gesture-based interface to control a car with a menu-based interface ($n = 16$). They similarly explain this result with the fact that their gesture-based system requires more training. It appears that users judge their own performance more negatively than it really is.

Letting the participants create their own gestures appeared to be beneficial for their learnability. This supports learnability since they make use of visual imagery and thus have meaning for their users (Ormrod, 2004). A study by Appert and Zhai (2009) similarly indicates the superiority of gestures over shortcuts since gestures "give the participants richer cues to form an association." Thus, gestures seem inherently more learnable than a menu-based interface that uses letters to identify its items. Although I did not obtain quantitative data to support this, the participants concurred with this conclusion (e.g., "Oliver, this was the O." or "This is my brother so it is the B.").

Allowing individual gestures for the commands does not appear necessary. This is since these gestures are practiced a lot while using the application and thus can be memorized more easily. The reasons to choose these gestures, they can be carried out quickly and can be recognized easily, were accepted by the participants. Besides, few participants had problems to recall these gestures.

H3. A high portion of participants (69%) answered the two questions to construct unknown gestures correctly. They constructed a three-part gesture (Call Emily at Home), without being told that gestures with three parts exist. They also combined a new primitive with known gestures. This result indicates that introducing three-part gestures in future versions is viable. Also, the motivations that users can generalize from known to unknown gestures appears supported. These indications are weak since the users

performed the gestures on paper and not in the context of the application. Thus, a study that addresses this hypothesis in practice should be carried out to fully answer this hypothesis.

H4. This hypothesis expects that using hierarchical gestures is more satisfying than using the standard interface. The results of both questionnaires show that the satisfaction of the application was perceived significantly positively on its own, and in comparison with the standard interface. The comments of the participants during and after the evaluation support these findings. Hence, this hypothesis is supported by the results of the pre-study.

5.2.5.2 Adaptations

Apart from the validation of the hypotheses, adaptations of the application were derived from the results of the pre-study. These adaptations address three issues that were exposed in the pre-study and are listed in Section 5.2.4.3. First, I explain how the issues that are caused by the timer that determines when the user's finger remains still on the screen were addressed. The second issue was that the time interval in which the user has to start the second gesture after ending the first one was too short. Some participants did not start the second part of the gesture fast enough. This caused frustration since the users had to redo the gesture. Four adaptations to this issue will be presented. Third, I discuss the size of the gesture button.

Within-gestures threshold. The timer that detects when the user's finger remains still on screen expired unintentionally for slow users. This happened since the reset of the timer was not triggered by very slow movements. To reduce this problem the threshold to reset the timer was lowered. Now, a movement of at least two pixels—instead of three—in x- or y-direction resets the timer. A test showed that this number is still large enough to allow gestures without a gap, while alloying to draw gestures more slowly.

Between-gestures threshold. The maximal time interval between the parts of a command was too short for some participants. Four adaptations that address this issue are presented here: implement a threshold that adapts itself to the speed of the user, use a command stack, automatically tell gestures from input towards the foreground application, and adding an extra button to deactivate the gesture mode.

Adapting threshold. It was observed that the length of the gap that the participants left between both parts of a command varied depending on their skills and experience. The application could mirror this by adapting itself to the speed of the user. The interval that the application allowed in

between would initially be set to a default value. If the user carries out a number of gestures quickly with a short gap between both parts, the length of the interval would decrease to allow the user to be faster. Analogously, if the user gestures slowly the length of the interval would increase.

However, this concept does not appear promising. The amount of data from which the length of the gap would have to be inferred is small. Thus, errors and frustration are probable. The speed of the user might converge to a certain value. This value might change when new gestures are added, or when a different person uses the phone. Besides, it is questionable whether the speed of the gestures or the length of the gap between gestures can serve as such an indicator at all. Thus, this concept was not pursued.

Stack. Another adaptation would be to put the commands on a stack. If the user carries out the gesture for an action, the application does not expect an argument to follow immediately. If the user supplies a suitable argument, the application takes the action from the stack and carries out the action using the supplied argument. This technique can be implemented in two ways:

1. Each gesture of a command has to start on the gesture button. After the first gesture was carried out the systems displays what primitive it has recognized. The user can carry out the second part of the command, the gesture of the contact or the contacts, later on. If the user wants to start a new command he can do the gesture that is assigned to the action of this command.

2. The gesture mode remains activated until a command was invoked or until the gesture mode is left by pressing some button.

Thus, the problems the timer caused cannot occur since a timer is not necessary in both approaches.

A modification of this concept is to provide the commands in Reverse Polish Notation (RPN). RPN reverses the order of action and argument: the argument is specified before the action. Using this technique, the command *Call Simon* would be issued by first carrying out the gesture for *Simon* and then the gesture for *Call.* This technique allows to use more than one argument for a command since the algorithm can be greedy and take all suitable arguments from the stack to, for instance, send an SMS to multiple recipients. When providing multiple arguments to the concept that was presented at the beginning of this section, the user has to indicate when he provided all arguments and wants to issue the command. A command with multiple arguments would be, for example, an SMS that is addressed to a group of people. RPN does not require to do so since all suitable arguments can be taken from the stack after an action was received.

Although both adaptations that were presented in this section seem promising they were not implemented, since doing so would changed the ideas underlying hierarchical gestures. Thus, the pre-study and the field study would no longer complement each other since they test a different concept.

Gesture detection. This approach should automatically tell gestures from interactions with the standard interface. Doing so the application recognizes when the user continues the gesture that he began, or when he stops gesturing and continues to interact with the interface of the foreground application.

If the user finishes a part of a gesture it is sent to the recognizer. The overlay that intercepts the touch events is put in a state in which it still listens for touch events of the user but passes them to the operating system. If the touch events are found to not be a gesture, for instance taps on icons or zooming of a map, the application stops listening and the gesture mode is deactivated. If the touch events are found to be gestures they are treated as a part of the gesture that the user began before and the touch events are intercepted.

This approach appears promising since it allows to abandon the timer while at the same time not interfering with the interactions of the user. Unfortunately, security constraints of Android preclude the realization of this concept. These considerations forbid applications to forward touch events to the operating system. Technically an implementation is possible but root rights are required. Since the evaluation was carried out on unrooted phones this approach could not be implemented. Nevertheless, I will sketch how it could be realized.

The critical task of this approach is to distinguish a gesture from input that is directed towards an underlying application. This distinction has to be made while the user is gesturing, since panning or scrolling operations benefit from immediate feedback. If the application is certain that the user carries out a gesture the input is no longer passed to the underlying application and interpreted as a gesture. *Gesture Search* (Li, 2010a) uses a three-step process to make this distinction.

1. **Size of the Bounding Box.** If the bounding box of the input is smaller than a threshold the input is not considered to be a gesture. This rule should identify taps.

2. **Squareness Measure.** The squareness measure is the aspect ratio of the bounding box. This rule is used to discern swipes or scrolling-gestures that have rather rectangular-shaped bounding boxes from gestures that typically have rather square-shaped bounding boxes. Li (2010a) uses a threshold of 0.275.

3. **Additional Heuristics.** Further measures can be added. Such heuristics can use the features of Rubine (1991), for instance the length of the input or its speed. Li (2010a) also uses whether the input of the user triggers a valid operation of the underlying application in the foreground as a criterion.

When evaluating Gestures Search it becomes apparent that this approach is difficult to implement. While Gesture Search successfully separates taps from other input, scrolling events are virtually always classified as a gesture. This behavior is possibly acceptable inside an application like Gesture Search. However, the application that is developed in this projected would interfere with the users' interactions with system applications and the operating system in general. Mistakes in the gesture detection lead to costly errors caused by, for example, unintentional calls. Summarizing, this approach seems to require a high implementation effort while promising a mediocre result as shown by the implementation in the Gesture Search application.

Mode button. A fourth way to address the described issue is to offer a button that deactivates the gesture mode. This button is displayed after a gesture is finished.

This concept is robust since it does not rely on heuristics or assumptions like some of the concepts that were presented in the preceding paragraphs. Furthermore, the implementation is simpler than the implementation of the other concepts. However, it demands additional effort from the users since a button has to be pressed to leave the gesture mode. Having to press a button also removes the advantage of the spring-loaded mode and, at least partly, transforms the gesture mode into a true mode. However, this button only has to be used when the user does an incorrect gesture or hits the gesture button by accident. Thus the extra effort that is required by this approach appears low. This concept is implemented in the version of the application that was used in the field study (see Figure 5.8).

Gesture button. Some participants missed the gesture button, probably because of its size. However, the size was not increased to not interfere with the interface of any application. I decided to decrease its size a little since it occluded elements of system applications. Furthermore, I expected that the problem of missing the gesture button would decrease with increasing practice. To make it stand out stronger visually, a black border was added. The same adaptation was made to the display of the textual feedback (see Figure 5.8).

Figure 5.8: The button at the bottom of the screen deactivates the gesture mode. The gesture button on the left and the text on the top of the screen received a black border to make them easier to identify.

5.2.6 Additions

The application that was used in the pre-study lacked features that are needed for independent use in an uncontrolled field study. For instance, it did not contain a feature to edit or delete gesture classes. The additions that were implemented to make the application suitable for the field study, a gesture reference and an improved configuration utility, are discussed in the following sections.

5.2.6.1 Reference

A reference of the defined gestures and their assignments is a crucial feature since the user need to know what gesture they have defined, how they look, and to what contacts they are assigned. This function was fulfilled by a sheet of paper on which the participants drew their gestures in the pre-study. In the field study this function has to be taken over by the application.

Such a system can work inside or outside of the context of the application that it supports. *Fluid Inking,* (see Section 3.5) for instance, is integrated with the user interface of the base application (e.g., a note-taking software). *Gesture Play* on the other hand is situated in a different context. Hierarchical gestures support the functionality of system applications. Integrating the system with a system application would require to change the code of that application. This is not feasible, since the evaluation should be carried out on unchanged stock devices. Thus, a help functionality that works in a different context was implemented.

The help functionality is shown in Figure 5.9. It displays the name of the gesture and to which primitive it is assigned. The first two points of the gesture are drawn in green to indicate the starting point and the direction of the gesture.

The help function can be summoned by a gesture, an *h.* Additionally, an item in the main menu of the application is available (see Figure 5.12).

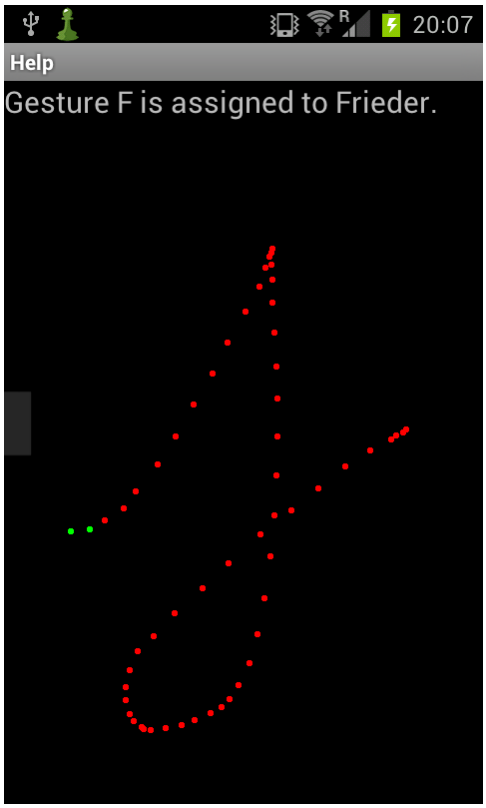


Figure 5.9: The help functionality of the application. It shows the template of the gesture class and its assignment to a primitive.

5.2.6.2 Persistence

The application that was used in the pre-study looses the templates when restarted since the templates were only stored in the temporary memory of the application. An important addition was to store gestures that the users defined persistently. The improved application stores the templates in text files that contain the points of the gesture and its name. Using text files allows for a simple implementation in which the component that writes the templates to the files notifies the Gesture Controller to refresh its templates if a file was changed or created (see Section 4.3.3.1).

The association between the gestures and the primitives are stored in the shared preferences of the application. Shared preferences[6] are a persistent storage technique that is provided by the operating system to store key-value pairs. Two places to store the data had to be used since shared preferences can only hold primitive types and, thus, not arrays of points as it would be necessary for the template of a gesture.

5.2.6.3 Edit gestures

A function to edit the gestures was implemented. This function is displayed in Figure 5.10. The shape of the gesture class is displayed in red. The new template can be drawn on top of it in green. Tapping the button submits the changes. A functionality to delete a gesture class was added as well.

5.2.6.4 Show or hide gesture button

The gesture button can be displayed or hidden using a toggle in the application's menu.

5.2.6.5 Autostart

To nudge users to use the application, a functionality to start the application when the operating system booted was necessary. Whether this autostart should happen can be determined by the user. It is activated by default.

5.2.6.6 Icon

To make the application easier to identify in between the other applications on the home screen, an icon was designed. The icon is depicted in Figure 5.11. The icon shows the letter "G" to represent "Gestures," the content of the application. Furthermore, the icon represents how the application is to be used, namely by starting a gesture on the left side of the screen on the gesture button. Furthermore, individual icons for the items of the user interface of the application were created (see Figure 5.12).

[6] see `http://developer.android.com/guide/topics/data/data-storage.html#pref`.

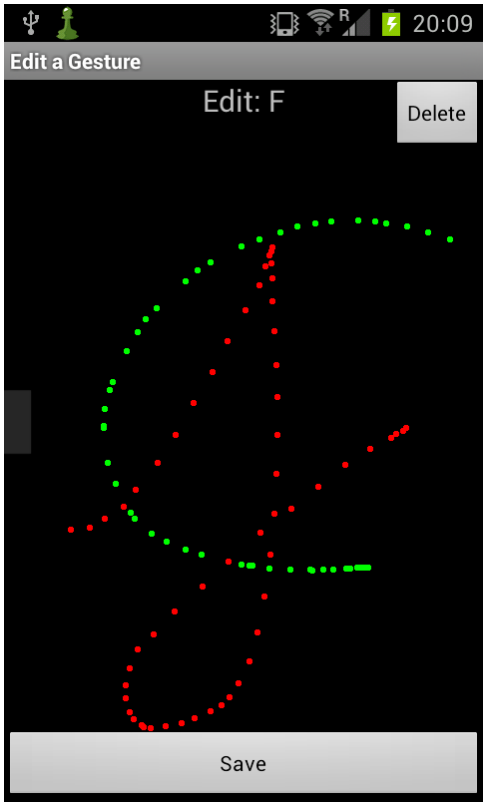


Figure 5.10: The interface to edit a gesture. The template of the gesture that is edited is shown in red. The new gesture is shown in green.

Figure 5.11: The icon of the application.

5.3 Field study

The results of the pre-study support the hypothesis that hierarchical gestures are more satisfying and the hypothesis the hierarchical gestures are more efficient than a menu-based interface. Also, the results indicate the validity of the hypotheses that target learnability. However, the results do not warrant a sound conclusion because of the short study that does not allow to validate all aspects of learnability that were targeted. A study over a longer time period was carried out to obtain more sound judgments. Apart from that, I wanted to find out whether users would use the application. The setup, the results, and the conclusions that were drawn from this study are discussed in the following section.

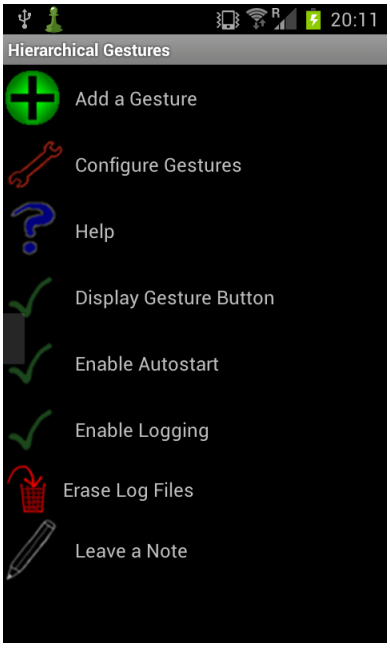


Figure 5.12: The user interface of the application.

5.3.1 Approach

A field study takes place where the evaluated system is being used. For my application, this is wherever the participants use their smartphones. Two settings to carry out the field study will be discussed in the following: a controlled setting in which the participants carry out tasks, and an open setting in which the participants can interact with the application as they want to, and if they want to. In the following section I discuss the characteristics of both approaches, and choose the most suitable one for the aim of this study. The aim of the field study was to find out whether the application would be used in practice and to assess whether the application is robust enough to be used in practice.

A controlled setting is less realistic since the participants are influenced in their interactions with the application. In reality there is always the possibility to use or not to use an application. Doing a controlled study would regularly remind the participants that they are part of an evaluation and, thus, lower the benefit of increased realism of a field study. A realistic judgment about whether an application that implements hierarchical gestures would be used was an aim of the study. Thus, an uncontrolled setting appears superior in this respect since it might yield more realistic data.

Having more control over the interactions of the participants lowers the risk that the participants stop interacting with the application over the course of the evaluation. However, even if not a satisfying one, the abandoning

of the application is a result that either indicates the necessity of further improvements of the application or motivates to give-up this line of research.

A controlled setting would provide more concrete pointers to issues and bugs of the application than an open setting. Still, feedback of the user can help to point out these issues after the test in an uncontrolled setting. Furthermore, the results of a controlled setting might be easier to use for a quantitative evaluation.

Li (2010a) carried out an uncontrolled field study with an approach similar to mine. He logged the interactions of the user with the application and used a satisfaction questionnaire that was distributed after the test. His study yielded results that were suitable to judge the quality of the application and to assess whether it would be used in practice. This example and the discussion that was carried out above led to the choice of an uncontrolled field study to pursue the aims of the field study.

5.3.2 Aims

The field study focusses on three issues: to find out whether an application that implements hierarchical gestures will be used in practice (see Section 5.3.2.1), to support the findings of the pre-study in a more realistic setting (see Sections 5.3.2.3 and 5.3.2.2), and to assess whether hierarchical gestures can be implemented in a robust application (see Section 5.3.2.3). A wide range of further aims for the field study can be motivated by the results of the pre-study. A discussion of issues that can be the subject of further investigation is provided in Section 6.1.2. Section 5.3.3 explains how the aims were pursued.

5.3.2.1 Hypothesis

One hypothesis was created for the field study to answer the question whether the application would be used in practice. It compares how often hierarchical gestures are used to how often the standard interface is used.

- HF1: **Participants use hierarchical gestures more often than the standard interface.** The pre-study supported the motivations of hierarchical gestures and indicated that hierarchical gestures are viable and usable. This hypothesis assesses whether the benefits of hierarchical gestures can be transferred from the lab to the daily practice of smartphone users. In a more realistic setting more environmental influences are present and users are not urged to use the application as they were in the pre-study. Thus, a more sound judgment is possible. The expectation is that hierarchical gestures get used more often than the equivalent function of the standard interface (e.g., use a gesture for *Call Frieder* or the use the standard interface to do so). This was

expected since the benefits of hierarchical gestures compared to the standard interface justify their usage.

5.3.2.2 Gesture usage

The pre-study yields indications of how hierarchical gestures are used in practice. These indication are about how many gestures the users define, how hey create the gestures, what functions they accelerate with gestures, and how they carry out the gestures.

- **How many gestures are defined?** It was expected, and stated by the participants of the pre-study, that five to ten gestures would be defined in total since then a boundary of learnability would be reached. This study tries to back this expectation with practical data. The data of the study allows to test whether there is a common fraction of contacts of the address book that users accelerate with gestures. It is, for example, possible that users with a relatively large address book use a larger number of gestures than users with a smaller address book.
- **How are the gestures constructed?** Techniques to construct gesture vocabularies have been discovered in the pre-study (see Section 5.2.4.4). However, these techniques were found in an artificial setting. The field study should indicate whether these techniques are also used in practice.
- **What functions are used?** The pre-study shows that the gain in efficiency is highest for functionality with a relatively high complexity. This warrants the expectation that gestures will be used most often for the commands *Send SMS* and *Call*, and least often for the command *Open.*
- **How are the gestures carried out?** All participants of the pre-study gestured with a gap. This is also expected for the field study since gesturing with a gap is faster than gesturing without a gap.

5.3.2.3 Application

The application was extended after the pre-study. Features, like the configuration utility, were added to make the application suitable for uncontrolled usage in the field study. It should be evaluated whether all features of the configuration utility are actually used. This data can motivate the presence and the further development of the features of the configuration utility. The mentioned extension introduced new sources of error. Thus it also had to be verified whether the application is robust.

- **Configuration.** The data of the field study can answer whether users use the functionality to modify the shape of the gesture classes and their assignment after their creation and their initial assignment to a contact. It is not expected that this functionality is used often since I expect a strong link between a gesture and its name. If a gesture is shaped like the letter *F*, for example. it is not expected that it gets assigned from the contact *Frieder* to, for instance, the contact *Thomas*.

- **Robustness.** An uncontrolled study allows to assess whether the implementation is robust enough to be used in reality. Tests indicate that no serious problems are to be expected. However, issues that are caused by rare circumstances or the long duration of the study are possible and are more likely to be exposed in an uncontrolled study with participants that were not involved in the development of the application. Furthermore the study allows to assess whether the application is compatible with different devices, a crucial issue for any Android application.

5.3.3 Methodology

The experiment was carried out as an uncontrolled field study over a duration of one week. This section describes the methodology of the experiment. First, it is discussed how the frequency of usage to address HF1 (Section 5.3.3.1) was measured. The second section discusses how the aims that target the usage of the gestures were pursued (Section 5.3.3.2). Section 5.3.3.3 explains how the aims regarding the robustness of the application were pursued. Section 5.3.3.3 explains how the usage of the application was measured. The users were encouraged to maintain a diary about their experiences with the application. This is explained in Section 5.3.3.4. The analysis of the data is explained in Section 5.3.3.5.

5.3.3.1 Hypothesis

HF1 (see Section 5.3.2.1) states that the participants of the study will use hierarchical gestures more often than the same function with the standard interface. This hypothesis was measured by logging the usage of the application. The usage frequency of hierarchical gestures was compared against how many times the same action was carried out with the standard interface (see Section 5.3.4.2 for a detailed explanation of the logging mechanisms).

5.3.3.2 Gesture usage

The aims that Section 5.3.2.2 introduced were pursued by logging the usage of the application, counting the number of gesture classes that were defined, and analyzing the templates of the gesture classes.

5.3.3.3 Application

The questions regarding the usage of the functions of the application were answered by logging the usage of each functionality (i.e., change the shape of the gesture, change the assignment to a contact, delete/add a gesture). It is unlikely that users change the shape of the gesture classes since it would distort the association between the name of the gesture class and its shape. Similarly, frequent changing of the assignment between a gesture class and a primitive is not expected. The robustness of the application is assessed using the reports of the participants after the study.

5.3.3.4 Diary

The participants were encouraged to note down anything that came to their mind during the field study. This should include problems with the application, or notes about the conditions in which they encountered bugs. No further means, for instance by daily popups, to request the user to use the diary were built into the application to not influence their usage of the application. This function was added to compensate for the lacking possibility to get prompt feedback at critical incidents of an uncontrolled field study. This data should support the analysis of the quantitative data of the evaluation.

The diary was a text field in which unstructured text can be entered. This data was written to a file and assigned a timestamp. No possibilities to retrieve or change entries—apart from manually accessing the log files—are available.

5.3.3.5 Analysis

No statistical tests were done for the analysis of the data, because the low number of participants did not yield a sufficient amount of data to do so. Median values were used for the comparison because of the low number of participants that did not yield a normal distribution of the results.

The numbers that were used in the analysis were obtained from the log files in which the raw data was stored. To answer whether the gesture-based interface or the standard interface was used more often for a task, the ratio between both numbers was calculated as $R = \frac{G}{SI}$. If the score for the standard interface was 0 it was set to 1 to obtain a defined result. If the score for the gesture-based interface was 0 the fraction evaluates to 0. No serious effect of this on the result of the study were expected since aggregated scores where used and little 0 values were expected.

Only contacts to which also a gesture was assigned were counted as uses of the standard interface. *Call Paul* was only counted as a use of the standard interface when there was a gesture assigned to *Paul*.

5.3.4 Technical setup

The following section summarizes the technical aspects of the field study. First, I explain the setup. Following, I summarize what data was logged and how the logging mechanisms were implemented. The third section discusses the privacy aspects of this study and what measures were taken to protect the privacy of the participants.

5.3.4.1 Test devices

The field study was carried out using the smartphones of the participants. The option of giving out smartphones for the purpose of the application was turned down since it would help to create a more artificial setting—participants use a different device—and, thus, might not log real usage. Besides, a set of such testing devices was not available to me. I hoped that there was no reluctance of the participants to install the prototype of an application on their own phones.

5.3.4.2 Logging

Various data about the behavior of the users had to be logged to pursue the aims of the evaluation. In the first part of this section, I explain what data was logged. The second part explains the implementation of the logging.

The following section lists the data that was logged during the study and why this was done. I only logged data that was necessary since the least amount of private data, for example the names of the contacts or the times when participants initiated phone calls, should be collected. All data that is presented in the following was assigned a timestamp.

Log files. The log of the participants' actions was written to a file in the storage of the device. This file only contained raw data. All processing was done after the evaluation to avoid possible negative effects on the battery life or the performance of the target device. The log file was not expected to reach a critical size. A log file with 2000 lines, for instance, has the still manageable size of 32kB. Creating a log of such a size would require about 150 actions per day over one week.

Listing 5.1 shows an exemplary log file. The last token in each line gives the timestamp (in milliseconds) at which the line was added to the file. In the following I explain how the information was encoded in the log file.

- The gestures that were recognized. Lines that start with a **G** log that a gesture has been recognized. The name of the gesture class is given as the second token.
- The commands that the application executed and how the gestures that invoked the command were carried out (with or without a gap).

Lines that start with a **C** log that a command has been carried out. The primitives of this command are given as the following tokens. The third token indicates how the gesture was carried out. *UU,* for instance, means that the gesture was carried out with a gap between both parts.

- The addition of gesture classes. Lines that start with an **A** log that a gesture class has been added. The second token provides the name of this gesture class.
- The modification of gesture classes. This was necessary to find out whether participants used the configuration utility and to pursue the aims that are discussed in Section 5.3.2.3. Lines that start with an **M** log that a gesture class has been modified. The second token provides what has been modified. An *S* indicates that the shape of that gesture was modified, an *A* indicates that the assignment between that gesture and a contact was modified. If the assignment to a contact was modified an additional token that indicates to what contact the gesture was assigned is added.
- The deletion of gesture classes. Lines that start with a **D** log that a gesture class was deleted. The second token provides the name of the gesture class.
- The size of the address book. Lines that start with an **S** log the size of the address book of the participant. The size of the address book is logged whenever the user added or deleted a gesture class to maintain the relation between the number of gesture classes and the size of the address book entries (see Section 5.3.2.2).
- The call history of the user and the recipients of the calls. Lines that start with **CA** log that a call was initiated. The name of the contact is given in the second token. Note that I use the hash value of the name of the contact to ensure the privacy of the participants. If such a line is preceded by a line `C <Call>` it was counted as a usage of the gesture interface. If this is not the case it is counted as a usage of the standard interface.
- The templates of the gesture classes.

Listing 5.1: Log file

```
G <EN> 1338639376218
G <S> 1338639377156
C <SMS> <-1560369887> UU 1338639377292
```

Call and SMS history. It is not possible to tell calls that were initiated with the standard interface from calls that were initiated using a hierarchical gesture. The same is the case for SMS. Thus, a heuristic had to be used. Whenever a call command was executed it was checked whether the system logged a call in the following ten seconds and whether no other command was issued using a hierarchical gesture. If this is the case it was assumed that the command was invoked using a gesture. To account for the production time of an SMS, a time interval of five minutes was used.

Sending SMS cannot be logged in the same way as the calling activities. Since the sending of SMS cannot be logged via broadcasts as the call events can, they have to be retrieved from the phone's database of sent SMS. This technique suffers from performance losses since each time the whole SMS database is written to the file. Appending new SMS to the file was turned down since it would require to check the content of the file and, thus, not bring performance benefits. To lessen the decrease in performance, the files were written in a separate thread. Listing 5.2 shows a log file with the SMS activities of the participant. The first token of each line is the hashed name of the contact, the second token is the timestamp.

Listing 5.2: SMS log file

```
-1114499585 1338191574252
-1801017768 1338112740532
322275427 1337970129474
-1114499585 1337853147229
```

Analysis. The log files were analyzed with a Perl script.

5.3.5 Privacy

When compared to the pre-study the field study poses additional privacy requirements. This is the case since the privacy of a third party, the contacts in the participant's address book, is affected since their names and phone numbers are processed by the application. One definition of privacy that is given by Introna (1997) is that privacy is the "control of information over oneself." If the application would log the phone numbers or the names of the contacts of the participants, their privacy would be violated since they did not consent to providing their phone numbers and cannot execute control over their information. Measures to keep the recipients of calls and SMS unidentifiable were taken since getting consent from the affected persons was not feasible. The information about when the user is initiating a call or sending an SMS is private, too. However, all this information had to be logged to carry out the study. In the following section I explain two mechanisms that were used to maintain the privacy of the participants.

5.3.5.1 Identification of contacts.

The times and the recipients of outgoing calls and sent SMS were logged. No further data was logged or retrieved by the application. Identical recipients had to be identifiable to compare how many times a gesture and how many times the standard interface was used to initiate a call or send an SMS. The name of the contact or his phone number could not serve as an identifier because of privacy reasons. Instead, I created a hash of the name of the contact using the `hashCode()` function of Java's String class. A random integer constant, which was unknown to me, was added to the hash values to make it more difficult to reverse the hash function and retrieve the name of the contact. Since the same input always creates the same hash value, and since the hash function is stable over multiple executions of the application it is possible to identify identical contacts while keeping their names and phone numbers hidden.

5.3.5.2 User control.

As the definition of privacy that I gave above claims, the control over information is a prerequisite of privacy. The participants had the possibility to disable or enable the logging at any time to be in control of their data. They could also erase the log files at any time.

5.3.5.3 Informed consent.

Loue (2002) stresses the role of the informed consent of the participants when doing research. The participants of the study needed to be informed about and consent to what was logged during the study, how this data was processed, and how their privacy was maintained. Thus, a new consent form that especially targets privacy was created (see Section B.1).

5.3.6 Plan

The following section explains the plan of the field study. The first section explains the setup of the study (see Section 5.3.6.1) followed by a summary of the participants of the evaluation (see Section 5.3.6.2).

The study commenced with an introduction in which the function of the application was explained (see Section 5.3.6.3). An experimenter did not have to be present during the introduction. Next, the application was installed on the devices of the participants and the participants could use the application for about one week. After this week they were scheduled for a post-test session in which an interview was conducted and the log-files were collected (see Section 5.3.6.4).

5.3.6.1 Setup

The field study was carried out as an uncontrolled and unsupervised field study over a duration of one week. This duration was chosen since I expected that it would be long enough to judge the validity of the hypothesis. If a participant does not use, or only rarely uses the application he is not expected to do so in a longer study. Similarly, if a participant uses the application within one week a longer study is not expected to yield more conclusive data.

5.3.6.2 Participants

The study was advertised by email. Three participants took part in the study ($\mu_{age} = 25.3$; $\sigma_{age} = 0.6$; 0 female; all right-handed). The population was biased in terms of experience with touchscreen devices ($\mu = 3$) and in terms of smartphone experience ($\mu = 2.7$). The population was consciously biased. This was done since I expected bugs in the application and hoped that a population with rather high technical experience would be less likely to stop using the application after being confronted with them.

The participants used the following three devices:

- Samsung Galaxy Gio, 320×480, 800 MHz, Android 2.2
- Samsung Galaxy S, 480×800, 1 GHz, Android 2.3.3
- Sony Experia Ray, 480×854, 1 GHz, Android 4.0.3

This selection of phones was sufficient for this study since it covers all screen resolution that were targeted by this project. For the future, phones with higher resolution need to be considered since they are getting increasingly common. Phones with a lower resolution than the minimal resolution that I targeted exist (e.g., Sony Ericson Xperia X10 Mini with a resolution of 240×320). However the chart that is provided by *Android Fragmentation Visualized* (2012) indicates that phones with a screen resolution of 240×320 and less are rare—and will probably become more rare in the future. Thus they were neglected.

5.3.6.3 Introduction

The introduction to the field study was done by delivering a manual (see Section B.2) and a document that explained what private data was collected (see Section B.1). This ensures similar conditions between the participants. Furthermore, the conditions were more realistic since in reality there is also only a written introduction and no oral training session available. After the introduction the participants received final instructions that repeated the most important aspects:

- The possibility to erase the log files at any time and the possibility to dis- and enable the logging at any time if desired.
- The possibility to enter notes in the application in case of bugs, remarkable events, or if the user has feedback to provide.
- To get in contact with me in case of problems.

Then the field study commenced. During the field study the participants could contact me and received assistance when they had questions, and in case of problems with the application. The participants did not receive instructions about how to use the application during the field study.

5.3.6.4 Post-test session

After the field study the participants were invited to a post-test session. First, the logs were downloaded from their phones. Then an informal interview followed. The purpose of this interview was to find out about the opinion of the participants on the application. This, furthermore, included how the participants used the application, what problems they encountered, and how they think that the application could be improved. The questionnaire that was used in the pre-study (see A.2) was used again to characterize the population.[7]

5.3.7 Results

This section presents the results of the field study. It is split in five sections: Section 5.3.7.1 presents the results that are used to validate the hypothesis, Section 5.3.7.3 gives the results about how many times the functions of the configuration utility were used, Section 5.3.7.4 describes the observation about the robustness of the application, and Section 5.3.7.5 describes observations about the user interface.

5.3.7.1 Hypothesis

This section presents the results of the field study that were used to validate the hypothesis. Table 5.4 shows how often the gestures were used. This table shows that all participants interacted with the application over the course of the study. Table 5.5 shows how many times the commands were used.[8] Table 5.5 shows that a considerable part of these gestures were tryouts and not complete commands since the number of gesture was higher than the number of commands.

[7] The question "Do you own a smartphone?" was removed since owning a smartphone was a requirement to participate in the study.

[8] G denotes actions that were carried out with hierarchical gestures, SI denotes actions that were carried out with the standard interface.

The results show that the participants used the gesture-based interface more often than they used the standard interface (a median of the ratios of all functions of 1.25 for all participants, a ratio of 3.6 when aggregating the values for all participants).

Table 5.4: Results of the field study; Gesture usage; The left column gives the names of the gesture classes, the following columns give the scores for the separate participants

	P1	P2	P3	Median
E	22	33	33	33
EN	10	1	3	3
ES	2	11	0	2
H	2	4	4	4
Others	5	11	5	5
$\sum$	49	68	48	49
Defined Gestures	4	3	3	**3**

Table 5.5: Results of the field study; Command usage

	P1	P2	P3	$\sum$	Median
Call_G	3	11	5	19	**5**
Call_SI	0	2	0	2	0
Send SMS_G	2	0	0	2	**0**
$\text{Send SMS}_\text{SI}$	4	0	0	4	0
$\sum \text{G}_\text{pp}$	5	11	5	21	5
$\sum \text{SI}_\text{pp}$	4	2	0	6	2
$\frac{\sum \text{G}_\text{pp}}{\sum \text{SI}_\text{pp}}$	1.25	5.5	1.25	—	**1.25**
$\frac{\sum \text{G}}{\sum \text{SI}}$				**3.6**	

5.3.7.2 Gesture usage

Table 5.4 shows that the participants created a low number of gesture class in relation with the size of the address book (with a median of 3 gesture classes, Table 5.6 shows the median value of the size of the address book of the participants which was 97). This finding is consistent with the expectation that only a number of less than ten gestures will be defined by the users.

The expectation that the more complex (i.e., *Call, Send SMS*) functions would be accelerated most of the time is not supported by the data of the study. Tables 5.5 and 5.7 show that *Open* is not the least used command (the median for *Open* is 1, the median for *Call* is 5, the median for *Send SMS*

Table 5.6: Results of the field study; Size of the address book

P1	P2	P3	Median
97	96	130	**97**

Table 5.7: Results of the field study; Usage of open and help

	P1	P2	P3	$\sum$	Median
Open_{G}	2	1	0	3	**1**
Help	2	4	4	10	4

is 0). This contradiction with the hypothesized behavior of the participants may be caused by the characteristic of the population that does not send many SMS (the median for *Send SMS* with the standard interface is 0).

The gestures that the participants used were all but one created using a letter as a model. I assume that this is the first letter of the contact since the gesture classes were given names that start with this letter. One gesture was, probably, created using a mnemonic association. None of the more complex shapes that were observed in the pre-study showed up.

Participants used all the gestures that they defined. The expectation that not all defined gestures are used is not supported. A probable reason for this is that only few gestures were defined in total (the median of defined gestures is 3). All gestures were carried out without a gap.

5.3.7.3 Configuration tool

Table 5.8 shows how many times the functions of the configuration utility were used. The usage of the functions was as expected. Most functions of the application were used with a similar frequency by all participants. The shape of a gesture was never changed. Sadly, the *Leave a Note* function was used only once.

Table 5.8: Results of the field study; Configuration tool usage

	P1	P2	P3	Median
Change Shape	0	0	0	0
Change Assignment	5	5	4	5
Add Gesture	12	8	3	8
Delete Gesture	8	5	0	5
Leave a Note	0	1	0	0
				5

5.3.7.4 Robustness of the application

The application turned out to be robust. The gesture recognition and the persistence of the gesture classes and the assignments performed well, and only few crashes under rather rare circumstances were reported. However, bugs hampered the proper functioning of the application.

This section discusses these observations in detail. The first part discusses observations about the performance of the gesture recognizer. The second part discusses bugs that the participants of the study reported.

Gesture recognition. Participants mentioned that the gesture recognition worked well, both in terms of performance and precision. Wrong recognitions did only occur when gestures were drawn sloppily; in fact so sloppy that the participants were aware that their sloppy gesturing caused the wrong recognition.

Bugs. As expected, the field study exposed bugs since it was used on heterogenous devices and by inexperienced users. In the following I discuss the two major issues.

- On one device (i.e., Samsung Galaxy S) the gesture button sometimes disappeared from the user interface. This happened when the user returned to the home screen after interacting with another application. Unfortunately, I was unable to trace back the source of this problem since I could not reproduce it on the phone that I used during development. Thus I can only speculate about the causes of this problem. Three explanations of this problem were considered:
  - The operating system kills the background service in which the application runs because of resource shortage. This explanation is unlikely since the service is classified as a foreground service, a service that the system should not kill even if it is low on resources (foreground services are for instance used for music players).
  - The disappearance of the gesture button is caused by the redrawing routine of the operating system. Probably, after leaving the application the complete user interface of the operating system is redrawn without the gesture button.
  - A task-killer application was installed on a phone on which this bug occurred frequently. It is possible that this application killed the background service of the application.
- It was not defined what phone number (e.g., mobile, work, or home) of a contact the application used. This led to unpredictable behavior and, obviously, the users were unable to choose what number they wanted to

use. A quick-fix was done by first using the mobile number, if no mobile number exists the private number, and if no private number exists the work number and documenting this in the manual. Approaches to completely solve this issue are discussed in Section 5.3.9.1.

5.3.7.5 User interface

The field study exposed two issues that are related to the user interface of the application. The first is, as in the pre-study, the size and the placement of the gesture button that led to undesired activations of the gesture mode. The second set of issues was caused by insufficient consideration of the fragmentation of Android devices. How the problems that are presented in the following were addressed is discussed in Section 5.3.8.4.

Gesture button. A salient problem was the unintentional activation of the gesture mode. The change of the size f the gesture button—it was made more narrow—after the pre-study was not sufficient to prevent accidental activation. Participants proposed to change its location, for instance by moving it towards the top or the bottom of the screen. Another suggestion was to decrease its breadth even more to reduce the risk of undesired activation.

Further feedback about the gesture button addressed its transparency. Participants claimed that it was somewhat too transparent at the beginning. However, after some time of using the application the transparency of the gesture button felt just right. A gradual shift between a rather opaque gesture button in the beginning and a more transparent gesture button at the end was proposed. Being able to adjust the transparency in the preferences of the application was proposed as well.

A similar comment was given about the dimensions of the gesture button. While appearing slightly too small in the beginning, the size was perceived to be just right later on after having gained some experience with the application.

Help. The help function was not salient enough for one participant. He started to interact with the application without reading the manual or consulting the help function first. Thus, he was not aware of the fact that the gesture for the actions are pre-configured.

Fragmentation. The application worked well on devices with different processing power and Android versions. However, devices with different screen resolution were not sufficiently supported. This caused the issues that are summarized in the following.

- The icon of the application was not suitable when being displayed on a display with a different pixel density different from that of the Samsung

Galaxy SII that was used during the development. For instance on a Galaxy S the icon is displayed too small.

- The size of the gesture button was specified as an absolute value. Thus, on devices with a lower screen resolution than the one of the development phone, the gesture button is too big. On devices with a higher resolution it will be too small. This problem for a screen with a too low screen resolution is shown in Figure 5.13.
- The templates of the gestures for the actions were recorded on a screen with a rather high resolution. The help function merely displayed the templates without scaling them down to the resolution of the display on which they are displayed. Thus, parts of the templates were drawn outside of the screen since, for example, the point $(400, 400)$ is visible on a Samsung Galaxy SII with a screen resolution of 480×800 but invisible on a Samsung Galaxy Gio with a screen resolution 320×480. Figure 5.14 depicts the occurrence of this problem.
- A two-lined label in the configuration utility was not entirely visible.

5.3.8 Discussion

This section interprets and discusses the results of the field study. First of all, the results of the study indicate that the hypothesis is valid and that hierarchical gestures would be used in practice. The application was in most situations sufficiently robust but exposed some bugs. However, aspects of the evaluation hinder a sound judgment. The test population was biased on purpose: they had high technical experience and did not use their phones a lot. Furthermore, I expect that the prototypic nature of the application influenced the results of the evaluation. Thus, the results that are discussed in the following section have to be taken with a grain of salt.

The first part of this section discusses whether the results of study can support the hypothesis (Section 5.3.8.1). Following that, I discuss how the participants used and perceived the gestures (Section 5.3.8.2). Section 5.3.8.3 discusses the comments regarding the help functionality. The field study motivated adaptations to the application that are discussed in Section 5.3.8.4.

5.3.8.1 Hypothesis

The results regarding the hypothesis of the study were positive. The ratio of 3.5 between the sum of all commands that were carried out with gestures and the sum of the usages of the standard interface for the same functionality indicate that the participants preferred the application with hierarchical gestures over the standard interface. Thus, I conclude that the hypothesis that should be tested in this field study is supported by this result. The

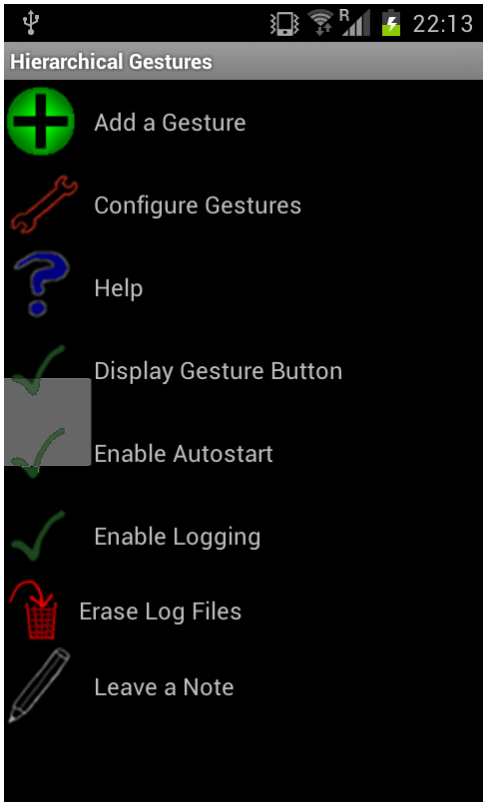


Figure 5.13: The gesture button appeared too big on devices with a screen resolution than the one of the development phone

conclusions regarding the practical usage of the application can be regarded as stronger than the conclusions that were drawn from the pre-study. This is since in this study was more realistic and since the participants were allowed to stop using the application and return to the standard at any time.

However, doubts about the soundness of the results are warranted. First of all, the population of the study is small ($n = 3$). Thus only 165 gestures were carried out in total. This is a low number compared to the 756 gestures that were carried out in the pre-study. The population was furthermore not representative, since only male participants were tested and since all participants had a high technical experience. The results also indicate that the participants rarely used the functionalities that hierarchical gestures support. A study with more frequent phone users would help to better support this hypothesis. To summarize, further studies with a more representative and bigger population appear advisable to obtain a more sound judgment of the validity of the hypothesis.

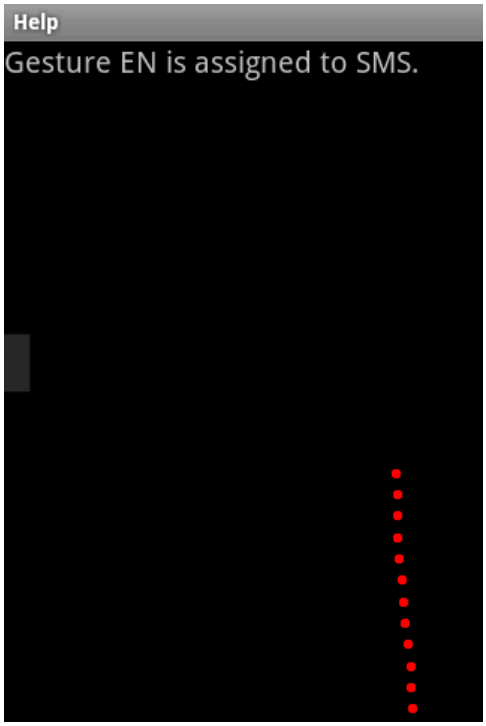


Figure 5.14: The template of the gesture is partly drawn outside of the screen when the help functionality is used on a device with a low screen resolution.

5.3.8.2 Gesture Usage

The majority of gestures was constructed using the first letter of the name of the contact to which it should be assigned. This judgment is based on the assumption that if the gesture class has the name of a contact, for instance *Frieder,* it should also be assigned to that contact. This was in contrast to the findings of the pre-study where more mnemonic or other techniques to contract gestures were used (see Section 5.2.4.4). This might be the case since the users could spent more time thinking about the gestures in the pre-study. Users seem to prefer using more obvious techniques to construct their gestures to finding and using mnemonic gestures. The results show that the gestures that are to be expected most frequently, at least for users that are used to the latin alphabet, are latin letters or gestures that are very similar to latin letters. The gesture recognizer and the user interface of the configuration utility can be tailored to this requirement to improve their performance.

One participant expressed a desire to use multi-stroke gestures. Although this might allow to express more complex gestures, an addition of multi-stroke gesture would introduce technical problems (e.g., segmentation) and decrease the benefit in efficiency. I believe that it is more likely that there are lots of

associations with contacts, but only few of them can be expressed visually as a gesture.

As in the pre-study participants mentioned the desire to have the possibility to assign their own gestures to the actions. A probable reason for this is that the gestures for the actions did not contain an obvious mnemonic connection to the actions. Participants even tried to define the gestures for the actions and only learned later on that this was impossible. As after the pre-study and for the same reasons—the gestures can be carried out quickly and having a consistent basis of the application over all installations—I decided to keep the gesture for the actions fixed. Furthermore, the gestures that the participants proposed for the actions, for instance a *C* for *Call,* would not be usable given the placement of the gesture button on the left side of the screen (see Figure 5.8). When trying to draw a *C* the user would inevitably collide with the boundaries of the screen.

No participant used the option to carry out the gesture without a gap. This behavior is understandable since it is slower in the current implementation. If the option to carry out the gesture as one stroke should be supported, it appears crucial to implement an algorithm that segments the multi-part gestures automatically. With such an algorithm carrying out the gesture in one stroke might be faster than doing so in several strokes. The application poses additional requirements since the segmentation should happen online, while the user is gesturing, to provide the user with immediate feedback. Segmentation is discussed again as further work (see Section 6.1.1.4).

It can be discussed whether the possibility to carry out a gesture in one continuous stroke is necessary at all since it introduces additional complexity. To allow this option the implementation of segmentation is vital. Implementing segmentation is not a trivial task and is expected to require considerable implementation effort. Besides, the user's finger might collide with the edges of the screen when carrying out a gesture with three or more parts. This might no longer be the case when adapting the application for tablet-PCs that have bigger screens. Pre-defined gestures, that are oriented in varying directions, could diminish this problem. However, my application allows the user to define gestures. The option to carry out the gesture in one stroke might be abandoned in future version of the application. It could be supported in applications that are restricted to gestures with only two parts or on tablet-PCs where more screen space is available.

5.3.8.3 Help function

I did not receive any comments of the participants about the help function. Thus, I consider its general concept to be supported. Adaptations and extension of the help functionality will be discussed in Section 5.3.9.3. One addition that was motivated by the comments of a participant is discussed in the following.

One participant did not consult the manual or the help functionality before using the application and thus encountered problems. Similar behavior of users can be expected if the application is deployed to a larger user base. A tutorial that appears when the application is started the first time can be added to support novice users of the application. During the tutorial they can be guided through the steps of adding a gesture, assigning it to a contact, and using the gesture. This can help to ensure that they are aware of all functionalities of the application and have an easier start.

5.3.8.4 Adaptations

The field study did not yield strong reasons to adapt the user interface. However, three adaptations that are necessary to address bugs are proposed in the following and have to be implemented in future versions of the application. Functional additions to the application that do not directly follow from the results of the study are presented in Section 5.3.9.

Fragmentation. Two adaptations were motivated by problems that were caused by insufficient consideration of the fragmentation of Android devices, especially the differing screen resolutions .

Size of the gesture button. To give the gesture button an equal size on devices with different screen resolutions, its height and width were specified relative to the height and the width of the screen on which it was displayed. The gesture button received a height of $\frac{1}{12}$ of the screen height and a width of $\frac{1}{17}$ of the screen width. These values were chosen to match the dimensions of the gesture button on the device on which I developed the application.

Scaling of templates. In the help functionality the templates of the actions were partly drawn outside of the screen of the users. This happened since these templates were not recorded on the devices of the users. Since the help functionality did not scale the templates the parts where the x- or y-coordinate was bigger than the width or height of the screen were invisible. To quick-fix this issue the templates were shifted upwards. This ensured that they are almost entirely visible even on devices with a small screen.

However, this adaptation does not fix the underlying problem. A real fix that was implemented after the completion of the field study was to scale down the templates so that they entirely fit the the screen on which they are displayed. Additionally, the templates were moved upwards so that they start in the middle of the gesture button, the place where the users start them anyway. The transformation was done in the following way[9]:

[9] X_{Old} denotes the untransformed point, X_{New} denotes the transformed point, $Width$ and $Height$ denote the dimensions of the screen, $Y[1]$ denotes the first point of the gesture.

$$X_{New} = \frac{X_{Old} * Width}{480}$$

$$Y_{New} = \frac{Y_{Old} * Height}{800} - (\frac{Height}{2} - Y[1])$$

The successful fix of this bug was verified by a test on a phone with a resolution of 320×480. I did not test on devices with lower resolutions since such devices are rare and were not available to me.

Icons. The icons that are used in the user interface of the application turned out to be too dark (see Figure 5.12). Their dark tone in combination with the black background of the application makes them hard to recognize and to distinguish in bright environments, for instance under sunlight. Thus, the color of the icons should be brightened in future versions of the application.

Position of the gesture button. The gesture mode was sometimes activated by coincidence. This happened since the gesture button was placed at a position that is, for instance, touched when panning a website. Three fixes for this issue are discussed in the following.

- Narrow the gesture button. The Bezel Swipe system of Roth and Turner (2009) uses a more narrow gesture-sensitive area than my application. Making the gesture-sensitive area more narrow could encourage the user to start the gesture by sliding from the frame of the phone into the screen. If the user could be trained to use this motion to start a gesture, the gesture-sensitive area could be shrunk so that accidental activation is less likely than it is now. The height of the gesture button can be increased to facilitate finding it.
- Move the gesture button to another position, for instance to the right edge of the screen. I chose the current position of the gesture button since I believe that this position is easy to reach. Thus, I expect that moving the gesture button to another position introduces problems regarding the ergonomics of the application. When moving the gesture button to the right edge of the screen for instance, users that usually write from left to right—possibly the majority in western Europe—would be forced to gesture against their usual writing direction.
- Allow users to configure the position of the gesture button. The advantage of this approach is that it is the user can determine the most comfortable and the least impeding position of the gesture button by himself.

The approach to make the gesture button more narrow appears to be the most promising one. Doing so is expected to reduce undesired activation

while at the same time keeping it at a position that is easy to reach. The other approaches were turned down since it appears difficult to keep the size of the gesture button and to find a position that is at the same time ergonomic, for instance by being easy to reach, and not impeding the interactions of the user with his device.

For a similar reason the possibility to configure the placement of the gesture button was turned down. Having the possibility to configure the position might lead to a position of the button that is either not ergonomic, and thus yields little usage of the application, or that impedes the user, and thus does not solve the problem at all.

Nevertheless, a configurable placement of the gesture button is promising for another reason. This reason uses the concept of the home position of the thumb that was introduced by Hirotaka (2003). This position is "where the thumb comes to rest naturally." However, this position is not an absolute position that is valid under all circumstances. The home position is likely to be affected by the device on which the application runs, by the size of the hand and of the fingers of the users, and how a user holds his device. Thus, a configurable placement may be necessary to account for these variations. Evaluations are needed to take a good decision between the choice to keep the position and narrow the gesture button, and adding an option in the preferences to set the location and the size. Evaluations may also answer the question of whether there is one position of the gesture button that satisfies the needs of all users. However, smartphones are increasing in its screen size (e.g., the Samsung Galaxy Nexus with 4.65 inches or the Samsung Galaxy SIII with 4.8 inches). Thus, the preference for one-handed usage that is for instance reported by Karlson et al. (1998) might no longer be valid for current devices, or thump usage is no longer feasible at all on devices with big screens. For the next version of the application the gesture button should be narrowed and increased in its height.

5.3.9 Additions and adaptations

Two additions, which do not follow directly from the results of the field study, may be added to a third version of the application: allowing to use a certain number of a contact as the default number (e.g., work, mobile or home), and allowing to change the position and the size of the gesture button.

5.3.9.1 Association to a specific number

A necessary addition is to choose which of the probably many numbers of a contact should be used (e.g., work, home, or mobile) when initiating a call or sending a SMS. This is hard-coded in the application that was used in the field study. This fact limits the practical usability of the application since the user cannot use all numbers of a contact. Two possibilities to allow

selecting which number to use will be considered: a property can be added to the association between a gesture and a primitive to indicate what number of the contact should be used (i.e., introduce primitives *Call at home* or *Frieder at home*), or three-part primitives (i.e., a third gesture to indicate *at home*) to do so can be introduced. A third approach would be to select a default number of a contact that is used when the user does not supply a gesture to specify which number of the contact should be used.

Evaluations may assess which of the three approaches is superior in terms of usability. However, adding the property to the association (the first approach) erodes the concept of hierarchical gestures since the gesture is then no longer a direct decomposition of the command. This is because parts of the command are pre-specified before. Furthermore it introduces the necessity to remember more gestures and decreases the performance of the recognizer since the gesture set is bigger. Thus an approach that introduces three-part gestures seems for further work (see Section 6.1.1.6).

5.3.9.2 Show gestures as lines

The templates of a gesture and the gestures that the user carries out are displayed as a series of dots by the application (see for instance Figure 5.10). This is done since the application receives and stores the gestures of the user as a series of coordinates and it is, thus, easy to draw them as such. Nevertheless, this way of displaying the templates and the users' gestures may contradict their mental model of a gesture. I expect users to see a gesture as one complete shape instead of as a series of isolated points. Thus the adaptation to display the templates as one continuous line should be implemented and evaluated.

The missing points between the dots can be, for instance, calculated by interpolation. Another way to implement this is to extend the Gesture Overlay View that is provided by the Android SDK and already displays the gestures as a line.

5.3.9.3 Help functionality

The help functionality should serve as a reference that lists the available gestures. The usage of this function was somewhat cumbersome since the user could only scroll through the list of gestures classes by means of swipe gestures. To improve this function, an overview page that displays all gesture classes and the primitives to which they are assigned in a list should be introduced. This overview could also include a small version of the shape of the gesture. After tapping an item, the fullscreen version of the current help functionality (see Figure 5.9) is opened.

In a later version the configuration utility and the help function could be integrated. This can be done by allowing to change the assignment of the gesture class or the look of a template at the same place where it is shown in the help function.

Chapter 6

Conclusion

It's not an exact science.

AD AGENT

THIS chapter wraps up the project, presents possibilities for further work, and draws the conclusions. First I present possibilities for further work that could be tackled in future projects. This includes extensions to the application and aspects of hierarchical gestures that further evaluations can investigate. The second section evaluates the methodology that was applied, summarizes the results and provides an outlook.

6.1 Further work

In this section I introduce possibilities for further work on the application, and on the concept of hierarchical gestures. The first part (Section 6.1.1) describes extensions in several directions that yield new research questions. The second part targets aspects of the concept of hierarchical gestures that can be addressed in greater detail in further studies (Section 6.1.2).

6.1.1 Extensions

The application can be extended in several directions. In Section 6.1.1.1 I propose to extend the feedback mechanism and include multimodal feedback that applies research about earcons and tactons. The Sections 6.1.1.2 and 6.1.1.3 target interface design and describe how hierarchical gestures could be better integrated with applications and how the available gestures could be made more visible. To be able to draw a hierarchical gesture in one stroke, segmentation is necessary. Considerations about segmentation are presented in Section 6.1.1.4. Section 6.1.1.5 explains further functions that might be supported by hierarchical gestures. Section 6.1.1.6 discusses the introduction of gesture with three or more parts and Section 6.1.1.7 proposes

to apply hierarchical gestures to other use cases. The usage of Reserve Polish Notation that was already suggested in Section 5.2.5.2 is further explained in Section 6.1.1.8.

6.1.1.1 Feedback

The application uses multimodal feedback: the recognition of a gesture is signalized by a vibration, and the name of the recognized primitive is displayed on screen. However, the application does not use acoustic feedback. Besides, more elaborate tactile feedback could be added. These additions may help to better exploit the potential of multimodal feedback to possibly improve the application. In the following, I sketch how this could be done.

If the user gestures without looking on the screen, he receives tactile feedback when a gesture was recognized. However, he cannot know whether the gesture that he intended was recognized. Without such information the user has to look on the screen to determine whether he has to restart his gesture because a wrong one was recognized, or whether he can continue. This information may be provided by tactons (e.g., Brewster & Brown, 2004)—structured tactile messages— or earcons (e.g., Blattner, Sumikawa, & Greenberg, 1989)—structured auditory messages. A tacton is a sequence of pulses and pauses that is realized by a tactor. An earcon is a sequence of sounds that are distinct by tone, pitch, and duration.

The structure of the command can be represented by the feedback that the application gives. Tactons and earcons can be assigned to each primitive and produced in the order in which they appear in the command. I will demonstrate this with an example. *Call* and *Frieder* would each have a distinct tacton and earcon. Two possibilities two produce the feedback for the command *Call Frieder* will be considered:

- If the application recognizes a primitive it produces the associated tacton and earcon immediately. After recognizing *Call* the tacton and the earcon for *Call* would be produced, and after recognizing *Frieder* the tacton and the earcon for *Frieder* would be produced. This approach would provide the information about the result of the recognition while the user is gesturing. Thus, mistakes can be recognized early. However, this approach is probably more obtrusive.

- The tactons and the earcons of the constituents of a command are produced after a complete command is recognized. After recognizing the command *Call Frieder* an earcon and a tacton that consists of a concatenation of the earcons and tactons of *Call* and *Frieder* are produced. This approach is less obtrusive than the approach that was presented above. However, the information about the recognition is available only after the command was recognized and thus only after

the command is carried out. A possibility for the correction of the error while the user is gesturing does not exist

Studies indicate that tactons and earcons are viable techniques to transport complex messages. Brewster, Räty, and Kortekangas (1996) report that earcons can convey information about four-level hierarchies. Tactons have been successfully used to provide navigational cues in projects of Pielot, Poppinga, and Boll (2010) and Lin, Cheng, and Yu (2008). The latter studies are relevant since they applied tactons in a use case with a high level of distraction: while navigating in a city and on the campus of a university.

The described concept can be inserted into the architecture of the application. The feedback can be attached to the primitives as sketched in Figure 6.1.

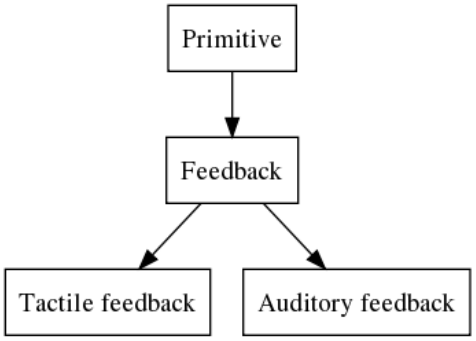


Figure 6.1: Feedback

The tactons can be given in an XML-format like the one that is given in Listing 6.1. The realization of such feedback is simple. The `Vibrator` component of the Android SDK can be supplied with a pattern that specifies the sequence and duration of pulses and pauses.

Listing 6.1: Specification of a tacton in XML

```
<tacton primitive="Call">
<pulse duration="100" />
<silence duration="50" />
<pulse duration="300" />
</tacton>
```

The earcons can also be specified in XML, for instance using MusicXML (*MusicXML 3.0 Specification*, 2011), and produced by the application. Another option is to provide the earcons as complete sound files.

A further question is how the tactons and the earcons are created. This could be done automatically. Williamson and Murray-Smith (2002), for example, propose techniques to automatically derive earcons from a gesture.

Techniques to support users in designing earcons may be based on grammars. Hankinson and Edwards (1999), for instance, propose to use musical grammars to design earcons.

Tactons can be either created automatically or user-defined. To the best of my knowledge, no research about the automatic creation of tactons exist. However, techniques to support and constrain the design of tactons have been proposed. Brown, Brewster, and Purchase (2006) propose a technique to use musical grammars. Most studies about tactons use non-standard tactors that can, for instance, convey different waveforms. For example Brown, Brewster, and Purchase (2005) convey the sensation of roughness using amplitude modulated sinusoids. The tactors of today's smartphones cannot convey such waveforms. Thus, these studies are only partly applicable to this project. Studies that make use of the tactors of stock devices exist. The study of Qian, Kuber, and Sears (2009) provides measures to determine whether two tactons that can be realized by available phones are distinguishable and, thus, can be used in the same vocabulary of tactons. This study implies limitations on the amount of tactons that can be used in one vocabulary.

A first study in this area might investigate whether the addition of the described feedback mechanisms yields benefits. Such benefits could motivate detailed investigations in this area. Further investigations may target the characteristics of the two ways to produce the feedback (giving feedback after each primitive, or after the recognition of the complete command) and how tactons and earcons can be designed.

6.1.1.2 Integration with applications

Zeleznik and Miller (2006) present a gesture-based system, Fluid Inking, that is integrated with a note-taking application. In this application, the first part of a two-part gesture can set the context for the second part. The context can be set by selecting text, for example using a lasso gesture. If the user draws an X on top of the selection, the selection is cut and transferred to the clipboard.

Similar possibilities can be offered by hierarchical gestures. The target of an operation, for instance a contact, could be selected by starting the gesture on top of it. This would allow one to express more complex commands with shorter gestures. The command *Call Frieder* could be carried out by only one gesture *Call* that is started on top of the contact *Frieder.* Doing so, a three-part command can be expressed with only two gestures. This possibility increases the efficiency of hierarchical gestures since the same result can be achieved with less effort. However, it also requires a new mode-switching technique since the gestures can start anywhere on the screen.

Possibilities to integrate hierarchical gestures with other applications exist. After selecting text, for instance using a lasso-gesture, a gesture for *Search google.com* could be carried out. This would use the selected text as

an input for a search in Google. Similarly the gesture for *Open Text Editor* could be carried out on top of a file, to open the file in a dedicated text editor, or in another application. Such possibilities could also be offered in a web browser. A picture on a website, for instance, could be stored or sent to a contact by starting a hierarchical gesture on top of it.

Even if the presented functionalities sound promising they aggravate the problem of mode-switching. A mode-switching technique like the one that the application uses is no longer possible since gestures can now start anywhere on the screen. Zeleznik and Miller (2006) were not confronted with this problem since their system is integrated with an application for free-form inking. Thus, after deciding that the input was a gesture the ink can be removed from screen. Such a technique is impossible in a system that is integrated with an application like a web browser, since the user input immediately triggers actions like panning. The distinction between a gesture and user input has to happen quickly. The application Gesture Search (Li, 2012) was confronted with the same challenge. A technique to solve this problem, which is based on the findings of Li, is described in Section 5.2.5.2 of this report. However, considering the non-optimal solution of this problem in the application of Li, it is worthwhile but difficult to tackle the challenge of designing and implementing a satisfying technique for this issue.

6.1.1.3 Visibility

An inherent problem of gesture-based systems is that the available gestures are not immediately visible to the user. In a text-based menu, in contrast, each option is visible as an item of a list. The application that was used in the field study offers a reference over the available gestures to make the available gestures visible. However, this system requires to leave the context of the application and is limited in its function. Having to switch the context is necessary since the code of the operating system could not be altered since the system has to run on non-rooted phones. If the requirement of being suitable for non-rooted phones was abandoned, a help-system that is more integrated with the application that it supports can be developed since then the code of the operating system could be adapted. Such a system would allow to, for instance, display gestures in the address book next to the contacts to which they are assigned. An example of a system that presents the available gestures and is integrated with the underlying application is the crib sheets that Kurtenbach et al. (1994) evaluated. A similar effort is the GestureBar system that Bragdon et al. (2009) present. Both efforts report positive effects of their systems on gesture learning and recall. These systems might be adapted to the mobile domain to test whether the benefits can be transferred.

After having made gestures more visible, a subsequent question is how to motivate the users to learn them. Grossman et al. (2007) studied techniques

to make people learn hotkeys. They distinguish two strategies: manipulating menu feedback (e.g., by increasing the visibility of the shortcut) and manipulating menu cost (e.g., by increasing the difficulty of using the non-accelerated way to invoke a command). Two approaches were reported to be most successful: one that uses audio feedback to notify the user of the presence of a hotkey, and one that simply disables the menu item and, thus, forced the user to use the hotkey. Data about how the users perceived these probably obtrusive techniques is not reported. These techniques can be applied to hierarchical gestures to evaluate whether they work in this domain as well.

6.1.1.4 Segmentation

The application cannot segment gestures automatically. Thus it is up to the user to indicate where a gesture ends and where a new one starts. This is done by pausing the movement or by lifting the finger from the screen. The effort to indicate the segment grows with the size of the gesture since future gestures should use gestures with three or more parts (cf. Section 6.1.1.6). The results of the evaluation show that the option to carry out the gesture without a gap was neglected by all participants. One reason is that it is slower than carrying out the gesture with a gap. Segmentation allows one to draw a gesture as one continuous stroke and the user does not have to indicate the start and the end of the segments. This extension makes the option to carry out the gestures as one stroke more attractive, since it would be faster than before and, possibly, faster than gesturing with a gap.

Starting points for segmentation techniques can be found in the field of handwriting recognition where this is a crucial capability (cf., e.g., Plamondon and Srihari (2000) for an overview of handwriting recognition). However, hierarchical gestures might pose different requirements since, compared to handwriting recognition where the latin alphabet is used, the templates are not known beforehand. Furthermore, the templates might vary in size depending on the context in which they are used (e.g., a gesture that is a part of a two-part gesture might be drawn bigger than the same gesture that is a part of a four-part gesture).

6.1.1.5 Extend functionality

The range of functionality that the application supports is narrow. An application that supports more functionality might be used more often and might support the adoption of hierarchical gestures. Additional functionality could include: starting applications, sending emails, or pictures. The implementation of these functions is, as long as they are supported by the Android SDK, simple. Technically, opening an application and opening a

contact is the same; the only difference is that the operating system selects a different application to carry out the action.

6.1.1.6 More-part gestures

The application is restricted to gestures with two parts. Gestures with three or more parts should be introduced to allow for more fine-grained commands and to evaluate the aspect of extensibility of hierarchical gestures in greater detail. Three-part gestures, for instance, allow one to distinguish between the commands *Call Frieder at Home* and *Call Frieder at work.* Evaluations should research the effect of this extension on the characteristics of hierarchical gestures. Such longer gestures might reduce the benefit in efficiency, or they might increase the error rate of the user or the error rate of the recognizer.

The Command Maintainer (see Section 4.3.3.1) is simple since it is only suitable for gestures with a length of two. Longer gestures require a more elaborate Command Maintainer since parsing the commands to check whether they are valid and to carry them out gets more complex. The current Command Maintainer checks the consistency of a command using a series of conditions. An exemplary condition is shown in Listing 6.2.

Listing 6.2: Consistency check

```
if (command.get(0).getType() == ACTION
  && command.get(1).getType() == CONTACT) {
  //Carry out the command
} else {
  throw new UnknownCommandException();
}
```

Enumerating all conditions is not manageable for long gestures and if more functionality that operates on different targets (e.g., files, URLs) is supported. As an extensible solution, specifying the valid commands using a formal grammar might be considered. The implementation of the parsing can be supported by available algorithms and libraries that work on grammars.

6.1.1.7 Further use cases and target devices

Hierarchical gestures are not limited to the use case that was investigated in this project: smartphone operating systems. Other use cases, for instance in the field of automotive user interfaces, could be targeted in further projects. As further target devices tablet-PCs, surface computers, or interactive whiteboards can be considered. Furthermore, it may be investigated whether the concept can be transferred to in-air gestures.

6.1.1.8 Reverse Polish Notation

In Section 5.2.5.2 it was proposed to specify the commands in Reverse Polish Notation. The mentioned motivations let this option seem a promising direction. To the best of my knowledge there is no research that investigates the usability of applications that use Reverse Polish Notation to enter commands. This would add the possibility to use more than one argument for a command without extra effort.

6.1.2 Evaluation

I focussed on the practical application of the concept in the second part of the project. Because of this, a field study was carried out, instead of, for instance, doing a second lab study. However, several aspects that could be evaluated in greater detail in further studies exist and are summarized in the following.

- The pre-study only provides a limited account of the **learnability** of hierarchical gestures. Further studies could delve deeper into learnability to, for instance, address whether broad hierarchies of gestures are easier to learn than deep hierarchies. The aspect of long-term learnability could be considered as well. This might be addressed by carrying out multiple sessions and measuring recall. Comparing the results over time would provide a judgment of long-term learnability. Participants might or might not be allowed to interact with the application between the evaluation sessions. Such a setting would consider the definition of learnability as "a function of experience" that Dix et al. (2004) propose.
- **Extensibility** is a possible limitation of hierarchical gestures. Further investigations may target the limitations of hierarchical gestures regarding extensibility. This can be done by, for instance, testing whether the efficiency of hierarchical gestures over the standard interface decreases if a gesture set exceeds a certain depth. The interaction between the size of a gesture set and its learnability can be targeted as well. This aim can be pursued by manipulating the size of the gesture set and measuring the effect on some dependent variable.
- **Ergonomics** is another characteristic of a gesture. Hierarchical gestures concatenate gestures. Thus, questions regarding their ergonomics arise since concatenating two ergonomic gestures might not yield an ergonomic gesture.

6.2 Discussion

The following section summarizes this report. The first section reflects about the applied methodology and concludes that it was suitable to pursue the

aims of the project and is, given more thorough testing, a good methodology to evaluate a new interaction technique. The second section discusses whether the project achieved its aims and provides an outlook.

6.2.1 Methodology

The methodology of this project consists of a controlled pre-study that was followed by an uncontrolled field study. The combination of these studies appears to be a good setup. A prototype is put into practice in a lab study to spot substantial flaws, generate feedback, and to verify the motivations of the evaluated interaction technique. After the pre-study an improved prototype is tested in an uncontrolled field study to find out whether the application might be used in practice, to test the concept in a more realistic setting, and to expose bugs in the application. Taken together, both studies can provide a first judgment about the feasibility and the usability of a new interaction technique, and provide sufficient feedback to implement a mature working prototype. In this section I discuss the methodology of both studies.

6.2.1.1 Pre-study

The pre-study proceeded successfully and the methodology by which it was carried out did not expose serious flaws. Three issues came up.

- The section of the usability questionnaire that targets efficiency could be improved since it achieved an insufficient internal validity. This can be done by finding, testing, and using items that are less ambiguous (e.g., participants did not recognize that “faster” and “saves time” ask for the same thing). Nevertheless, another usability questionnaire that fits the project and whose internal validity is supported by quantitative data might be adopted.

- Reading out the task descriptions, measuring and noting down time-on-task, and observing the participants was too demanding for one experimenter. Hence, the addition of a second experimenter would be beneficial. One experimenter could be responsible for everything that involves communicating with the participant (i.e., reading out the tasks descriptions and answering questions). The second experimenter could be responsible for measuring time-on-task and observing the participant. The second experimenter may ask questions in the interview after the evaluation. This might lead to more precise measurements and thus to more conclusive data.

- The actions of the participants were observed on the phone that they used during the test. This made it difficult to precisely observe what they were doing at anytime. A technical means to observe the screen of the device on a computer should be introduced.

The results of the pre-study indicate the validity of the motivations of hierarchical gestures. Possibilities for adaptations of the application were collected. After the pre-study the application was adapted based on the results and tested in an uncontrolled field study. The field study should find out whether this application would be used in practice and is sufficiently robust.

6.2.1.2 Field study

The methodology of the field study was simple. Participants received a manual and the application and were asked to try it out for a week. After a week the log files were collected and an interview was carried out. This methodology did not expose serious flaws. One issue came up.

- A pre-test training session should be carried out to ensure that all participants have the same starting point. This is motivated by one participant who claimed that he did not read the manual in the beginning and thus encountered problems that the ones that read the manual did not encounter.
- A more structured approach to the post-test interview appears advisable. Such an approach helps to ensure completeness and might elicit more and more conclusive data.

The field study yielded useful data despite the low number of three participants. This data is expected to be more sound than the data that was obtained by the pre-study since the participants interacted with the application for a longer time and in a more realistic setting.

The quantitative results of the field study were limited. These limitations were caused by bugs in the application and by the low number of participants. Both reasons will be discussed in the following. As expected, the field study exposed bugs in the application. The major bugs were caused by too narrow testing on a device on which the bugs did not occur. A lesson that has to be learned from this is that the fragmentation of Android devices needs to be considered by using a broad range of devices for testing. Testing devices have to cover various screen resolutions (at least from 320×480 to 480×800), and different Android versions (i.e., 2.3.x and 4.x, if tablet-PCs are targeted also 3.x,). Also the interplay with other applications, for instance task killers, that might affect the application has to be checked.

The development of an Android application is more difficult than the development of an iOS application when it comes to testing. Pachal (2012) reports that there are 3997 different devices that run Android with highly different capabilities. Applications for iOS have to target only five iPhones, three iPads, and four generations of the iPod touch. These devices are similar in many respects (e.g., screen resolution between the iPhone and the iPod

touch). Nevertheless, I would still choose Android as the target platform for similar projects (see Section 4.3.1 for the reasons).

The low number of participants for the field study was justified by the expectation that even a low number of participants can expose the major bugs. Besides, a test with a low number of participants reduces the risk of wasting participants with a flawed prototype. Nevertheless, a study with more participants and a more mature prototype yields more and more conclusive data.

I advise to improve the application using the results of the field study. The improved application should be tested in another field study with more participants to draw more sound conclusions.

Summarizing, a field study is a good approach to gather quantitative data. The participants of the field study expressed that they were willing to interact with the application but were hindered by the flaws of the application.

6.2.1.3 Summary

Given bigger resources for development and especially for testing, I would recommend this methodology for a first complete investigation of an interaction technique. The pre-study validates—or not—the motivations of the tested interaction technique. Feedback to address the most severe flaws of the application before the field study can be obtained as well. The field study then tests whether the interaction technique, as it is implemented in the application, will be used in practice and whether the motivations of the interaction technique can be confirmed in a realistic setting. Taken together, both studies allow for a first but complete judgment of the characteristics of an interaction technique.

6.2.2 Summary and outlook

The concept of hierarchical gestures appears promising after the investigations that were carried out in this project. It has motivations based on theory (see Section 2.3) that justify its usability and its suitability for mobile applications. These motivations were supported by a lab study (see Section 5.2.4) with 14 participants. This study showed that hierarchical gestures are perceived to be significantly more satisfying and significantly more efficient than a reference interface (the standard interface of the Android operating system) that does not use gestures. Only the subjective scores for learnability do not indicate a significant preference for hierarchical gestures. The scores for efficiency show that users, nevertheless, mastered the application quickly. In fact the performance was on an equal level over all three trials, and better than with the standard interface. This indicates that hierarchical gestures are, despite the perception of the users, learnable.

Apart from answering the research question several technical challenges were successfully addressed. The critical parts of the application—in my opinion the gesture recognition and the storage of the primitives and the templates—worked flawlessly on a wide range of different devices over one week. This indicates that hierarchical gestures can be implemented in a robust and practical application. However, it became obvious that the transfer from the lab to the daily practice was not as easy as hoped. The practical evaluation of the application in a field study was impeded by technical shortcomings (see Sections 5.3.7.4 and 5.3.7.5). Because of this, and because of the low number of participants, the conclusions of the field study must not be taken as sound conclusions. The technical problems were caused by too narrow testing of the application. Propositions to address these issues have been presented (see Section 5.3.8.4) and implemented in the application. I recommend to further improve the application, subject it to rigid and exhaustive testing, and then carry out another field study with more participants and a more representative population to obtain sound quantitative data.

Although it was the focus of this project, it is important to note that the concept of hierarchical gestures should not be restricted to the use case of smartphone operating systems. Further use cases that could benefit from hierarchical gestures can be targeted in further investigations (see Section 6.1.1.7). As one example, in automotive interfaces low visual distraction, one characteristic of gesture-based interfaces, is a core requirement. Cars contain heterogenous devices (e.g., CD player, navigation, radio, climate control). Hierarchical gestures could provide a unified gesture language to address all these devices.

Further investigations into hierarchical gestures can contribute to the research in gesture-based interaction and introduce new research questions. The focus of current research about gesture-based interaction is on the design and the evaluation of concrete gesture-based applications and interaction techniques (see Section 3.4). However, few efforts target gestures in a systematic and fundamental way. Future research, for example about hierarchical gestures, could address how gestures that are intrinsically usable can be developed, or how a usable set of gestures can be designed for a given set of commands. Further efforts about hierarchical gestures motivate an interdisciplinary approach, for instance with the field of psychology. This is necessary to answer questions like how users form a mapping between a person, represented as a contact, and a gesture, or what kind of command structure, for instance the traditional order *Action Target Modifier* or Reverse Polish Notation, is easier to learn. Doing so might yield results from which other gesture-based interfaces could benefit as well.

I proposed hierarchical gestures, a technique to compose a gesture set. This technique has characteristics that make it usable. Practical investigations indicate that the use case of a smartphone operating system can benefit

from hierarchical gestures. This is since the results of two studies indicate that hierarchical gestures are a usable technique and meet the requirements of mobile interaction. However, further investigations to address the flaws of the application and to arrive at more sound conclusions are necessary. To summarize, I believe that further efforts about hierarchical gestures are interesting and worthwhile.

Appendix A

Pre-Study

A.1 Consent form

Purpose

Welcome and thanks for participating in this user test. The purpose of this test is to evaluate a prototype of a gesture-based interface for smartphones. By participating in this test you help to identify aspects of the system that have to be modified in the final version.

Procedure

During the test, you will be asked to complete some tasks while I observe you. Furthermore, you will be asked to fill in questionnaires. Feel free to ask any questions that you might have at any time during the evaluation. Please note that I might refrain from answering questions about the prototype during the user test. The session is expected to last about fifteen minutes. Approximately 10 people will participate in total.

Risks

No foreseeable risks are associated with your participation.

Breaks and withdrawal

If for any reason you are uncomfortable during the session and do not want to complete a task, you may say so and we will move on to the next task. In addition, if you do not want to continue, you may take a break or end the session and leave at any time.

Usage of the results and confidentiality

The results will be included in a report that will be presented to my supervisors at the university and might be published in further papers. Your name will not be included in the report nor will you be associated with any session data collected. I will take all reasonable efforts to keep your results unidentifiable and keep the data confidential.

Compensation

Regrettably, I can offer no monetary compensation for your participation.

Questions

You may ask questions at any time during the test. If you have any questions after the test, or if you are interested in the results, contact Frieder Loch at f.loch@student.utwente.nl.

By signing this form you indicate that you agree to the terms stated in this form and that you participate in this user testing session voluntarily.

Date:

A.2 Questionnaires

Gender

- Female
- Male
- Prefer not to disclose

Age

- ___
- Prefer not to disclose

Are you

- right-handed
- left-handed
- Prefer not to disclose

Do you own a smartphone?

- Yes
- No

How would you rate your experience with smartphones?

- Novice
- Intermediate
- Advanced

How would you rate your experience with touchscreen devices in general?

- Novice
- Intermediate
- Advanced

Table A.1: First questionnaire of the pre-study

	Strongly Disagree				Strongly Agree
I am satisfied with the system.	☐	☐	☐	☐	☐
I would recommend the system to a friend.	☐	☐	☐	☐	☐
The system is fun to use.	☐	☐	☐	☐	☐
The system works the way I want it to work.	☐	☐	☐	☐	☐
The system is wonderful.	☐	☐	☐	☐	☐
I feel I need to have this system.	☐	☐	☐	☐	☐
The system is pleasant to use.	☐	☐	☐	☐	☐

Table A.2: Second questionnaire of the pre-study

	Gesture-based System				Standard Interface
This system is more efficient.	☐	☐	☐	☐	☐
I would prefer to use this system.	☐	☐	☐	☐	☐
I learned to use it quickly.	☐	☐	☐	☐	☐
This system is more fun.	☐	☐	☐	☐	☐
This system saves time.	☐	☐	☐	☐	☐
This system is more attractive.	☐	☐	☐	☐	☐
Using this system is more pleasant.	☐	☐	☐	☐	☐
This system is faster.	☐	☐	☐	☐	☐
This system is easy to learn.	☐	☐	☐	☐	☐

Create Gestures

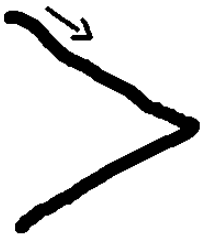
In this section you are asked to create three gestures that you did not use in the test. Please apply the knowledge of the gestures that you have used. Draw your gestures on the blank space.

Call Emily at Home

In the test you used a gesture to *Call Emily.* The gesture that is drawn below means *at home.* Please draw the gesture to *Call Emily at Home.*

Send Email to Oliver

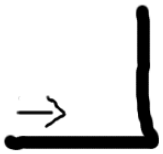
In the test you used a gesture to *Send SMS to Oliver.* The gesture that is drawn below means *Send Email.* Please draw the gesture for *Send Email to Oliver.*



A.3 Reference Sheets

Call

Send SMS



Open

Emily

Jack

Eric

James

Oliver

A.4 Scripts

Using this interface you can control certain functions of a smartphone, for example you can call a contact. To test whether this interface is usable I will ask you to carry out tasks using the gesture-based interface and the standard interface of the smartphone. You will have to carry out three trials with 18 tasks. Please carry them out as fast and as precise as possible.

Structure

At first, I will explain how the gestures are structured. Have a look at this gesture. (Show a gesture, for example Figure 2.2) As you can see, the gesture consists of two parts that are, in this sketch, distinguished by their color. Each part represents a part of the whole command. The black part is carried out first and represents *Call,* the red part is carried out afterwards and represents the fictional contact *A* in your address book. By combining both gestures you express the command *Call A.*

Demonstration

Now, I demonstrate how to carry out this gesture on a smartphone. You have to begin each gesture on the white square on the left side of the screen. (Carry out the gesture without a gap) As you can see, you have to pause shortly after each part of the gesture until the device confirms the recognition. If necessary you can also leave a gap between both parts of the gesture. (Carry out the gesture with a gap) Please try out the sample gesture until you feel comfortable with the system.

Gestures

You can express three commands with the system. This is done by these gestures. (Show gestures for *Call, Open,* and *Send SMS*) You can assign the gestures that are used for the contacts by yourself. In this evaluation you will use five contacts: *Emily, Jack, Eric, James,* and *Oliver.* Please assign a gesture to each of them and draw the gesture on this sheet of paper. You are allowed to use the sheet for reference in the first two trials. The third trial has to be carried out without this sheet. Please do not choose gestures that resemble the gestures of the commands to precent recognition errors. (Draw the gestures on the sheet, Enter the gestures into the system) Please try to open each contact once to practice the gestures and to make sure that the system can distinguish them.

A.5 Contacts

- Emily
- Anna

- Julia
- Isabella
- James
- Jacob
- Oliver
- Jack
- Isa
- Emma
- Ethan
- Eric
- Michael
- Daan
- Marc

Appendix B

Field Study

B.1 Consent form

Purpose

Welcome and thanks for participating in this user test. The purpose of this test is to evaluate a prototype of a gesture-based interface for smartphones. By this test we want to find out whether people would, and if yes how, they use this system in their daily routine. Furthermore, we want to learn what aspects of the system have to be modified in the final version.

Procedure

In this evaluation you will try a prototype of a gesture-based interface to control your smartphone. You will be provided with an introduction about the system. Then you will be supplied with the application for one week. After this week you will be invited for another session with an interview. Approximately 5 people will participate in this test.

Technical Risks

The application that will be used during the evaluation was developed and tested carefully. Thus, it is not expected that your mobile phone or the data on it suffers any harm. Nevertheless, I cannot guarantee that this software is free of errors and will function as expected. If you encounter any problems or if you have any questions send an email to Frieder Loch.

Risks

No risks that go beyond the risks of using a smartphone are expected during your participation.

Collection of Private Data

We log aspects of your phone usage to answer our research questions. However, we took measures to protect your privacy and the privacy of your contacts. In the following the data that is logged and the measures to protect your privacy are explained.

- **Phone calls.** The application logs when you initiate a call and who you are calling. Neither the name of the recipient nor his phone number will be stored in the logs. The application generates a random identifier instead.
- **Outgoing SMS.** The application logs when you send an SMS and who receives this SMS. Neither the name of the recipient nor his phone numbers will be stored in the logs. The application generates a random identifier instead. **The content of the SMS is not stored or even retrieved.**
- **Your activities with the application.** It is logged what functions of the application you use. This includes the addition and removal of gestures and to what contact you assign a gesture.

Note: To give you control over when and if data is being logged you can

- disable and enable the logging and
- erase the log file at any time.

Refer to the document that you received for further instructions about how to do this. The log files are stored are on your sdcard in the folder `gestures`.

Breaks and Withdrawal

You may end or adjourn the study at any time by disabling or by removing the application from your device.

Usage of the Results and Confidentiality

The results will be included in a report that will be presented to my supervisors at the university. Your name will not be included in the report nor will you be associated with any session data collected. I took, and I will take, all reasonable efforts to keep your results unidentifiable and keep the data confidential.

Compensation

Regrettably, I can offer no monetary compensation for your participation.

Questions

If you have any questions after or during the test, or if you are interested in the results, contact Frieder Loch at f.loch@student.utwente.nl.

B.2 Introduction of hierarchical gestures

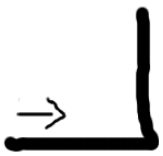
This document explains the application that you are going to test.

Structure

The system is controlled by gestures. All gesture are made of two parts: an action and a target. The system supports three actions: **Call, Send SMS,** and **Open.** In the following the gesture that are associated to these actions are given.

Call

Send SMS

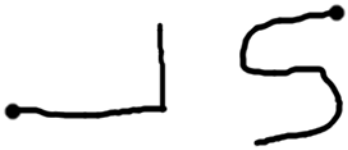


Open

To provide a complete command a second gesture is needed. This gesture specifies the target of the command. It is up to you to specify this gesture using the function “Add a Gesture” that is available in the main menu.

How to Gesture

I will explain how a gesture is carried out by giving an example. Let us suppose that the left gesture represents *Call* and the right gesture represents a contact in your address book, for instance *Frieder*.



To carry out the gesture to *Call Frieder* carry out both gesture in a row as it is shown in the following graphic (the red gesture is carried out after the black one).

You can carry out a gesture with this application in two ways:

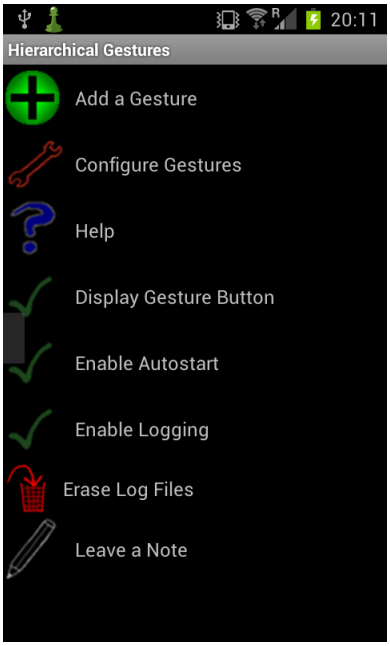
1. **Without a gap.** Carry out the first part of the gesture, rest with your finger on the screen until you feel a short vibration, then carry out the second part of the gesture.

2. **With a gap.** Carry out the first part of the gesture, lift your finger from the screen, carry out the second part of the gesture.

You have to start each gesture on the gray button on the left side of the screen. If you carried out a wrong gesture, press the “Leave Gesture Mode Button” and try again.

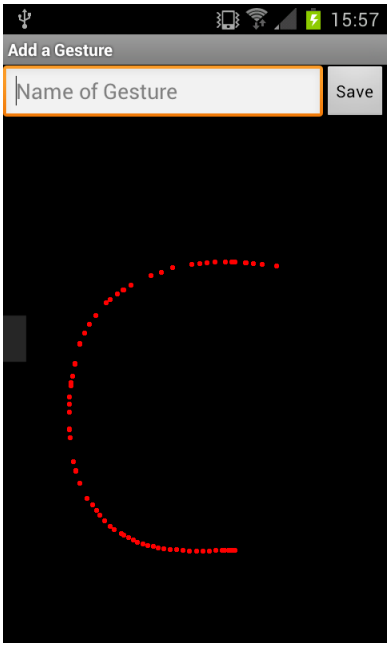
User Interface

The following picture shows the functions that are available from the main user interface of the application. Each of them will be explained later on.



Add a Gesture

Use this function to add a new gesture to the system. Draw the gesture on the black part of the screen and enter the name at the top. You can only use letters and numbers to name the gestures.



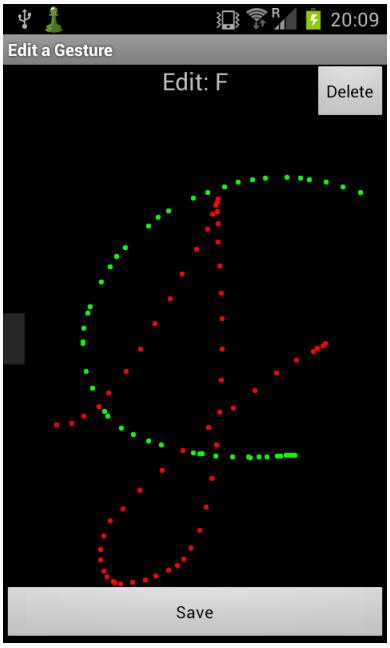
To improve the accuracy of the gesture recognition, keep the following recommendations in mind :

- You shouldn't use gestures that look similar to gestures that are already in the system. A diagonal line, for instance, is likely to be confused with the gesture for the action "Call" (see above).
- The direction of a gesture is relevant. Thus, a circle that you draw clockwise is distinct from a circle that you draw anti-clockwise.
- Draw the gestures big and slowly when you add them.
- Delete gestures that you don't need.

After having add a gesture, you need to assign it to a contact using the function "Configure Gestures" that will be explained in the following.

Configure Gestures

This function is used to control the assignment between a gesture and a contact. If you do a long-tap on the name of a gesture class you can modify its shape or delete it. The original shape of the gesture class is shown in red. The new shape of the gesture class is drawn on top of it in green.



Help

This function displays all gestures that are available and the contacts to which they are assigned. To display the following or the previous gesture you have to swipe to the left or to the right. The first two points of the gesture are drawn in green to indicate the direction of the gesture.

Display Gesture Button

Use this button to show or hide the gesture button.

Enable Autostart

Use this toggle to select whether the application should start automatically when the phone boots.

Enable Logging

As you know, it is logged how you use the application. Use this toggle to dis- or enable the logging temporarily or permanently.

Erase Log Files

If you want to erase all logs that were made so far, make use of this functionality. The logging will continue after the deletion of the log files if the "Enable Logging" toggle is enabled.

Leave a Note

This system is a prototype. Thus, it is likely that things do not function as they should, that things are confusing. and demand improvements. If you encounter such a thing let me know by entering a textual note to describe what happened and what you wish to be different.

In Case of Problems

If the application should crash, restart it. If this does not help reinstall the application, or contact me.

Bibliography

Alpern, M., & Minardo, K. (2003). Developing a car gesture interface for use as a secondary task. In *CHI '03 extended abstracts on human factors in computing systems* (pp. 932–933). New York, NY, USA: ACM.

Anderson, D., Bailey, C., & Skubic, M. (2004). Hidden Markov model symbol recognition for sketch-based interfaces. In *AAAI fall symposium* (pp. 15–21). Menlo Park, CA: AAAI Press.

Android 4.0 compatibility definition. (2012). `http://source.android.com/compatibility/4.0/android-4.0-cdd.pdf`. (accessed on July 18, 2012)

Android fragmentation visualized. (2012). `http://opensignalmaps.com/reports/fragmentation.php/`. (accessed on June 29, 2012)

Appert, C., & Zhai, S. (2009). Using strokes as command shortcuts: cognitive benefits and toolkit support. In *Proceedings of the 27th international conference on human factors in computing systems* (pp. 2289–2298). New York, NY, USA: ACM.

Bach, K. M., Jæger, M. G., Skov, M. B., & Thomassen, N. G. (2008). You can touch, but you can't look: interacting with in-vehicle systems. In *Proceedings of the twenty-sixth annual SIGCHI conference on human factors in computing systems* (pp. 1139–1148). New York, NY, USA: ACM.

Bau, O., & Mackay, W. E. (2008). Octopocus: a dynamic guide for learning gesture-based command sets. In *Proceedings of the 21st annual ACM symposium on user interface software and technology* (pp. 37–46). New York, NY, USA: ACM.

Blattner, M. M., Sumikawa, D. A., & Greenberg, R. M. (1989). Earcons and icons: Their structure and common design principles. *Human–Computer Interaction*, *4*, 11–44.

Bragdon, A., Nelson, E., Li, Y., & Hinckley, K. (2011). Experimental analysis of touch-screen gesture designs in mobile environments. In *Proceedings of the 2011 annual conference on human factors in computing systems* (pp. 403–412). New York, NY, USA: ACM.

Bragdon, A., Uguray, A., Wigdor, D., Anagnostopoulos, S., Zeleznik, R., & Feman, R. (2010). Gesture play: motivating online gesture learning with fun, positive reinforcement and physical metaphors. In *ACM*

international conference on interactive tabletops and surfaces (pp. 39–48). New York, NY, USA: ACM.

Bragdon, A., Zeleznik, R., Williamson, B., Miller, T., & LaViola, J. J., Jr. (2009). Gesturebar: improving the approachability of gesture-based interfaces. In *Proceedings of the 27th international conference on human factors in computing systems* (pp. 2269–2278). New York, NY, USA: ACM.

Brewster, S., & Brown, L. M. (2004). Tactons: structured tactile messages for non-visual information display. In *Proceedings of the fifth conference on australasian user interface - volume 28* (pp. 15–23). Darlinghurst, Australia: Australian Computer Society, Inc.

Brewster, S., Räty, V.-P., & Kortekangas, A. (1996). Earcons as a method of providing navigational cues in a menu hierarchy. In *Proceedings of hci on people and computers xi* (pp. 169–183). London, UK: Springer.

Brown, L. M., Brewster, S., & Purchase, H. C. (2005). A first investigation into the effectiveness of tactons. In *Proceedings of the first joint eurohaptics conference and symposium on haptic interfaces for virtual environment and teleoperator systems* (pp. 167–176). Washington, DC, USA: IEEE Computer Society.

Brown, L. M., Brewster, S. A., & Purchase, H. C. (2006). Tactile crescendos and sforzandos: applying musical techniques to tactile icon design. In *CHI '06 extended abstracts on human factors in computing systems* (pp. 610–615). New York, NY, USA: ACM.

Burke, J. L., Prewett, M. S., Gray, A. A., Yang, L., Stilson, F. R. B., Coovert, M. D., et al. (2006). Comparing the effects of visual-auditory and visual-tactile feedback on user performance: a meta-analysis. In *Proceedings of the 8th international conference on multimodal interfaces* (pp. 108–117). New York, NY, USA: ACM.

Callahan, J., Hopkins, D., Weiser, M., & Shneiderman, B. (1988). An empirical comparison of pie vs. linear menus. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 95–100). New York, NY, USA: ACM.

Castellucci, S. J., & MacKenzie, I. S. (2008). Graffiti vs. unistrokes: an empirical comparison. In *Proceedings of the twenty-sixth annual SIGCHI conference on human factors in computing systems* (pp. 305–308). New York, NY, USA: ACM.

Dix, A., Finlay, J., Abowd, G., & Beale, R. (2004). *Human–computer interaction*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Ecker, R., Broy, V., Butz, A., & De Luca, A. (2009). pieTouch: a direct touch gesture interface for interacting with in-vehicle information systems. In *Proceedings of the 11th international conference on human–computer interaction with mobile devices and services* (pp. 22:1–22:10). New York, NY, USA: ACM.

Frankish, C., Hull, R., & Mirgan, P. (1994). *Recognition accuracy and user acceptance of pen interfaces* (Tech. Rep.). Bristol, UK: HP Laboratories Bristol.

Freeman, D., Benko, H., Morris, M. R., & Wigdor, D. (2009). ShadowGuides: visualizations for in-situ learning of multi-touch and whole-hand gestures. In *Proceedings of the ACM international conference on interactive tabletops and surfaces* (pp. 165–172). New York, NY, USA: ACM.

Goldberg, D., & Richardson, C. (1993). Touch-typing with a stylus. In *Proceedings of the INTERACT '93 and CHI '93 conference on human factors in computing systems* (pp. 80–87). New York, NY, USA: ACM.

Grossman, T., Dragicevic, P., & Balakrishnan, R. (2007). Strategies for accelerating on-line learning of hotkeys. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 1591–1600). New York, NY, USA: ACM.

Grossman, T., Fitzmaurice, G., & Attar, R. (2009). A survey of software learnability: metrics, methodologies and guidelines. In *Proceedings of the 27th international conference on human factors in computing systems* (pp. 649–658). New York, NY, USA: ACM.

Gyung, M., & Cho. (2006). A new gesture recognition algorithm and segmentation method of korean scripts for gesture-allowed ink editor. *Information Sciences*, *176*(9), 1290–1303.

Hankinson, J. C. K., & Edwards, A. D. N. (1999). Designing earcons with musical grammars. *SIGCAPH Comput. Phys. Handicap.*, 16–20.

Hinckley, K., Baudisch, P., Ramos, G., & Guimbretiere, F. (2005). Design and analysis of delimiters for selection-action pen gesture phrases in scriboli. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 451–460). New York, NY, USA: ACM.

Hirotaka, N. (2003). Reassessing current cell phone designs: using thumb input effectively. In *CHI '03 extended abstracts on human factors in computing systems* (pp. 938–939). New York, NY, USA: ACM.

Höysniemi, J., Hämäläinen, P., Turkki, L., & Rouvi, T. (2005). Children's intuitive gestures in vision-based action games. *Commun. ACM*, *48*(1), 44–50.

IEEE. (2008). *Utility conventions.* `http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap12.html`. (accessed on July 18, 2012)

IEEE. (2012). *Shell command language.* `http://pubs.opengroup.org/onlinepubs/9699919799/utilities/V3_chap02.html#tag_18_09_01`. (accessed on July 18, 2012)

Intent |Android developers. (2012). `http://developer.android.com/reference/android/content/Intent.html`. (accessed on June 26, 2012)

Introna, L. D. (1997). Privacy and the computer: Why we need privacy in the information society. *Metaphilosophy*, *28*(3), 259–275.

Isokoski, P. (2001). Model for unistroke writing time. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 357–364). New York, NY, USA: ACM.

Karlson, A. K., Bederson, B. B., & Contreras-Vidal, J. L. (1998). Studies in one-handed mobile design: Habit, desire and agility. In *Proceedings of the 4th ERCIM workshop on user interfaces for all.*

Kjeldskov, J., & Graham, C. (2003). A review of mobile hci research methods. In L. Chittaro (Ed.), *Human–computer interaction with mobile devices and services* (Vol. 2795, p. 317-335). Springer Berlin / Heidelberg.

Kline, P. (2000). *Handbook of psychological setting.* New York, NY, USA: Routledge.

Kurtenbach, G., & Buxton, W. (1993). The limits of expert performance using hierarchic marking menus. In *Proceedings of the INTERACT '93 and CHI '93 conference on human factors in computing systems* (pp. 482–487). New York, NY, USA: ACM.

Kurtenbach, G., & Buxton, W. (1994). User learning and performance with marking menus. In *Proceedings of the SIGCHI conference on human factors in computing systems: celebrating interdependence* (pp. 258–264). New York, NY, USA: ACM.

Kurtenbach, G., Moran, T. P., & Buxton, W. (1994). Contextual animation of gestural commands. *Computer Graphics Forum*, *13*(5), 305–314.

Kurtenbach, G., Sellen, A., & Buxton, W. (1993). An empirical evaluation of some articulatory and cognitive aspects of "marking menus". *Journal of Human Computer Interaction*, *8*(1).

Lee, S., & Zhai, S. (2009). The performance of touch screen soft buttons. In *Proceedings of the 27th international conference on human factors in computing systems* (pp. 309–318). New York, NY, USA: ACM.

Li, Y. (2010a). Gesture search: a tool for fast mobile data access. In *Proceedings of the 23nd annual ACM symposium on user interface software and technology* (pp. 87–96). New York, NY, USA: ACM.

Li, Y. (2010b). Protractor: a fast and accurate gesture recognizer. In *Proceedings of the 28th international conference on human factors in computing systems* (pp. 2169–2172). New York, NY, USA: ACM.

Li, Y. (2012). Gesture search: Random access to smartphone content. *Pervasive Computing*, *11*(1), 10–13.

Lin, M.-W., Cheng, Y.-M., & Yu, W. (2008). Using tactons to provide navigation cues in pedestrian situations. In *Proceedings of the 5th nordic conference on human–computer interaction: building bridges* (pp. 507–510). New York, NY, USA: ACM.

Loue, S. (2002). Ethical issues during and after the study. In *Textbook of research ethics* (p. 113-169). Springer US.

Lund, A. M. (2001). *Measuring usability with the use questionnaire.* `http://www.stcsig.org/usability/newsletter/0110_measuring_with_use.html`. (accessed on February 24, 2012)

MacKenzie, I. S., & Soukoreff, R. W. (2002). Text entry for mobile computing: Models and methods, theory and practice. *Human–Computer Interaction*, 147–198.

MacKenzie, I. S., & Zhang, S. X. (1997). The immediate usability of graffiti. In *Proceedings of the conference on graphics interface '97* (pp. 129–137). Toronto, Ont., Canada, Canada: Canadian Information Processing Society.

Maguire, M. (2001). Context of use within usability activities. *International Journal of human–computer Studies*, *55*(4), 453–483.

Morris, M. R., Wobbrock, J. O., & Wilson, A. D. (2010). Understanding users' preferences for surface gestures. In *Proceedings of graphics interface 2010* (pp. 261–268). Toronto, Ont., Canada, Canada: Canadian Information Processing Society.

Moyle, M., & Cockburn, A. (2003). The design and evaluation of a flick gesture for 'back' and 'forward' in web browsers. In *Proceedings of the fourth australasian user interface conference on user interfaces 2003 - volume 18* (pp. 39–46). Darlinghurst, Australia, Australia: Australian Computer Society, Inc.

Musicxml 3.0 specification. (2011). `http://www.makemusic.com/musicxml`. (accessed on May 14, 2012)

Nielsen, J. (1992). The usability engineering life cycle. *Computer*, *25*(3), 12–22.

Nielsen, J. (1993). *Usability Engineering* (First ed.). San Francisco, CA, USA: Morgan Kaufmann.

Nielsen, J. (2005). *Ten usability heuristics.* `http://www.useit.com/papers/heuristic/heuristic_list.html`. (accessed on July 12, 2012)

Nielsen, J. (2009). *Powers of 10: Time scales in user experience.* `http://www.useit.com/alertbox/timeframes.html`. (accessed on April 23, 2012)

Nielsen, M., Störring, M., Moeslund, T., & Granum, E. (2004). A procedure for developing intuitive and ergonomic gesture interfaces for hci. In A. Camurri & G. Volpe (Eds.), *Gesture-based communication in human–computer interaction* (Vol. 2915, pp. 105–117). Springer Berlin / Heidelberg.

Norman, D. A. (2002). *The design of everyday things.* New York, NY, USA: Basic Books.

Ormrod, J. E. (2004). *Human learning* (Fourth ed.). Upper Saddle River, NJ, USA: Merrill/Prentice Hall.

Pachal, P. (2012). *Here's how bad android's fragmentation problem is.* `http://mashable.com/2012/05/16/android-fragmentation-graphic/`. (accessed on June 19, 2012)

Pielot, M., Poppinga, B., & Boll, S. (2010). Pocketnavigator: vibro-tactile waypoint navigation for everyday mobile devices. In *Proceedings of*

the 12th international conference on human computer interaction with mobile devices and services (pp. 423–426). New York, NY, USA: ACM.

Plamondon, R., & Srihari, S. (2000). Online and off-line handwriting recognition: a comprehensive survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *22*(1), 63 -84.

Qian, H., Kuber, R., & Sears, A. (2009). Towards identifying distinguishable tactons for use with mobile devices. In *Proceedings of the 11th international ACM SIGACCESS conference on computers and accessibility* (pp. 257–258). New York, NY, USA: ACM.

Raskin, J. (2000). *The humane interface: New directions for designing interactive systems*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.

Richter, H., Ecker, R., Deisler, C., & Butz, A. (2010). Haptouch and the 2+1 state model: potentials of haptic feedback on touch based in-vehicle information systems. In *Proceedings of the 2nd international conference on automotive user interfaces and interactive vehicular applications* (pp. 72–79). New York, NY, USA: ACM.

Roth, V., & Turner, T. (2009). Bezel swipe: conflict-free scrolling and multiple selection on mobile touch screen devices. In *Proceedings of the 27th international conference on human factors in computing systems* (pp. 1523–1526). New York, NY, USA: ACM.

Roudaut, A., Bailly, G., Lecolinet, E., & Nigay, L. (2009). Leaf menus: Linear menus with stroke shortcuts for small handheld devices. In T. Gross et al. (Eds.), *Human–computer interaction – interact 2009* (Vol. 5726, pp. 616–619). Berlin, Heidelberg: Springer.

Rubine, D. (1991). Specifying gestures by example. In *Proceedings of the 18th annual conference on computer graphics and interactive techniques* (pp. 329–337). New York, NY, USA: ACM.

Saffer, D. (2009). *Designing Gestural Interfaces* (First ed.). Sebastopol, CA, USA: O'Reilly.

Sears, A., & Arora, R. (2002). Data entry for mobile devices: an empirical comparison of novice performance with jot and graffiti. *Interacting with Computers*, *14*(5), 413 - 433.

Sezgin, T. M., & Davis, R. (2005). Hmm-based efficient sketch recognition. In *Proceedings of the 10th international conference on intelligent user interfaces* (pp. 281–283). New York, NY, USA: ACM.

Shneiderman, B., Plaisant, C., Cohen, M., & Jacobs, S. (2010). *Designing the user interface: Strategies for effective human–computer interaction* (Fifth ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Tullis, T., & Albert, B. (2008). *Measuring the User Experience* (First ed.). Morgan Kaufmann.

Vatavu, R.-D., Vogel, D., Casiez, G., & Grisoni, L. (2011). Estimating the perceived difficulty of pen gestures. In *Proceedings of the 13th IFIP TC*

13 international conference on human–computer interaction - volume part ii (pp. 89–106). Berlin, Heidelberg: Springer.

Wikipedia. (2012a). *Mnemonic.* `http://en.wikipedia.org/wiki/Mnemonic`. (accessed on July 16, 2012)

Wikipedia. (2012b). *Smartphone.* `http://www.en.m.wikipedia.org/wiki/smartphone`. (accessed on July 10, 2012)

Williamson, J., & Murray-Smith, R. (2002). *Audio feedback for gesture recognition* (Tech. Rep.). Glasgow, Scotland, UK: Department of Computing Science, University of Glasgow.

Wobbrock, J. O., Aung, H. H., Rothrock, B., & Myers, B. A. (2005). Maximizing the guessability of symbolic input. In *CHI '05 extended abstracts on human factors in computing systems* (pp. 1869–1872). New York, NY, USA: ACM.

Wobbrock, J. O., Morris, M. R., & Wilson, A. D. (2009). User-defined gestures for surface computing. In *Proceedings of the 27th international conference on human factors in computing systems* (pp. 1083–1092). New York, NY, USA: ACM.

Wobbrock, J. O., Wilson, A. D., & Li, Y. (2007). Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual acm symposium on user interface software and technology* (pp. 159–168). New York, NY, USA: ACM.

Zeleznik, R., & Miller, T. (2006). Fluid inking: augmenting the medium of free-form inking with gestures. In *Proceedings of graphics interface 2006* (pp. 155–162). Toronto, Ont., Canada, Canada: Canadian Information Processing Society.