Fieldward and Pathward: Dynamic Guides for Defining Your Own Gestures

Joseph Malloch[1] **Carla F. Griggio**[1] **Joanna McGrenere**[2,1] **Wendy E. Mackay**[1]

[1]LRI, Univ. Paris-Sud, CNRS, Inria, Université Paris-Saclay, F-91400 Orsay, France

[2]University of British Columbia, Vancouver, Canada

{malloch, griggio, mackay}@lri.fr; joanna@cs.ubc.ca

ABSTRACT

Although users accomplish ever more tasks on touch-enabled mobile devices, gesture-based interaction remains limited and almost never customizable by users. Our goal is to help users create gestures that are both personally memorable and reliably recognized by a touch-enabled mobile device. We address these competing requirements with two dynamic guides that use progressive feedforward to interactively visualize the "negative space" of unused gestures: the *Pathward* technique suggests four possible completions to the current gesture, and the *Fieldward* technique uses color gradients to reveal optimal directions for creating recognizable gestures. We ran a two-part experiment in which 27 participants each created 42 personal gesture shortcuts on a smartphone, using *Pathward*, *Fieldward* or *No Feedforward*. The *Fieldward* technique best supported the most common user strategy, i.e. to create a memorable gesture first and then adapt it to be recognized by the system. Users preferred the *Fieldward* technique to *Pathward* or *No Feedforward*, and remembered gestures more easily when using the technique. Dynamic guides can help developers design novel gesture vocabularies and support users as they design custom gestures for mobile applications.

ACM Classification Keywords

H.5.2. Information Interfaces and Presentation (e.g. HCI): User Interfaces

Author Keywords

Dynamic guides; Gesture recognition; Personalized gestures; Progressive feedforward; Controlled experiment.

INTRODUCTION

Billions of mobile phones [23], laptops and tablets contain sensors that detect touch and can capture drawn 2D gestures. When used as input, gestures offer multiple advantages over traditional buttons, sliders and pull-down menus: they are fast to execute, and can be performed "eyes-free" in the dark or if the display is not visible. Gestures can efficiently access large sets of commands, and can be concatenated or otherwise combined into hierarchies or gestural grammars.

If gesture interaction is so useful, and touch capability is ubiquitous, why is gesture interaction on mobile devices mostly limited to pinch, drag and swipe? One possible reason is learnability [5]: Users remember the pinch gesture after seeing it once; complex gestures require more effort. Systems such as *Marking Menus* [11], *Octopocus* [5] and *Arpège* [7] provide in-context feedforward in the form of dynamic guides that reveal potential gestures or chords and associated commands. Experts perform known gestures quickly and efficiently, whereas novices pause to see the available gestures appear around their finger or fingers. *Octopocus* and *Arpège* also indicate the likelihood of each remaining gesture, based on previously performed input.

The above techniques help users learn *existing* gesture sets, but offer no support for creating or modifying them. Just as users prefer to create their own memorable banking PINs, rather than those issued by the bank, we believe that users will prefer to create their own gestures, associating them with symbols, objects, actions, or even "muscle memory" to facilitate their recall. However, creating personally memorable gestures addresses only half the problem: the system must *also* reliably recognize them [25].

We propose in-context dynamic guides that let users explore ideas for new, memorable gestures, and determine which ones the system can also recognize. We believe that mapping user-defined gestures to command shortcuts and scripts will increase the adoption of gesture-based interaction on mobile devices. For example, a user might associate a custom gesture with a script that snaps a picture and texts it to her boyfriend. Or, she might redefine a system-defined gesture, such as a question mark for "help", and access it from any application or hardware platform.

This paper describes the related work on recognizing and learning gestures, as well as customizing interfaces. We then describe the design and implementation of two in-context dynamic guides with different forms of progressive feedforward. *Pathward* is inspired by Octopocus's [5] path-based visualization of the "positive space" of *existing* gestures, but instead displays samples from the "negative space" of *possible* gestures. *Fieldward* displays a heatmap-like display of the *im-*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CHI 2017, May 06-11, 2017, Denver, CO, USA

© 2017 ACM. ISBN 978-1-4503-4655-9/17/05…$15.00

DOI: http://dx.doi.org/10.1145/3025453.3025764

mediate future of the user's gesture. We present the results of a two-part experiment that compares these two techniques and a no-feedforward condition. We discuss potential applications of our approach, with implications for design and directions for future research.

RELATED WORK

Recent advances in mobile phone and tablet technology make it possible to capture and recognize diverse forms of gesture input. From the user perspective, challenges include design of gesture sets, learnability, and personalization. Technical challenges include correctness, noise tolerance, computational cost, and ease of implementation.

Designing Gesture Sets

Current touch-based mobile platforms typically support *direct manipulation* gestures, such as pinching and dragging, to manipulate objects on the screen. Such gestures are usually simple, based on the metaphor of interacting with physical or virtual objects. Gestures may also be used to generate words on a soft keyboard [29] or to enter passwords on a mobile device [25]. *Gesture typing* is significantly faster than pressing keyboard keys: When entering passwords, users prefer drawing patterns to pressing buttons, even though button presses are easier for the system to recognize.

Several visual taxonomies ([24], [19]) lay out the features that distinguish different gestures on tabletop and mobile platforms. We are specifically interested in gestures that invoke commands on mobile devices. These may incorporate one or more strokes or segments – each of which may be straight or curved – and may include additional dimensions, such as pressure or temporal features.

A key challenge is to design coherent *sets* of gestures that successfully address this complexity. One approach is to use letter-shaped gestures that spell enough of the command name to identify it uniquely [13]. Another strategy is to use a *Marking Menu* [11], which organizes commands in a radial layout around the cursor. The user invokes a command by moving the cursor in the command's direction.

Although *Marking Menus* are up to three times faster than ordinary pull-down menus, they can only handle a limited number of items, usually between eight and 16. However, the same radial structure can also be applied to hierarchically organized commands, as in *Hierarchical Marking Menus* [12]. *Flower* [4], and *Wave* menus [3] also use a radial layout and support hierarchical organization, but use specific schemas to organize the gestures. Appert and Zhai [1] argue that designers should avoid complex, hard-to-visualize recognition algorithms.

Other researchers have involved users in the design of application gesture sets, asking them to generate and interpret "natural gestures" [27, 15] or as part of a participatory design session [21]. Researchers who compare the memorability of free-form gestures find that user-defined gestures are both easier to master [16] and easier to remember [17]. These studies suggest that users should play a role in defining gestures, but focus only on designing standardized gesture sets for future applications, rather than on how to let users create their own gestures in the field.

Learning Gesture Sets

Kurtenbach and Buxton [11] focused on the trade-off between designing for novices, who value discovery and learning, and experts, who value speed and robustness. Their *Marking Menu* offers an elegant solution – experts who already know the appropriate gesture just execute it, whereas novices pause to display a *Marking Menu*. Users see both visual *feedback* indicating the gesture they just performed, and *feedforward* indicating the directions of the remaining possible commands. Over time, novices learn the gestures associated with each command, without requiring conscious effort or a specific intent to learn. The system is forgiving – experts who forget a gesture may also pause and the menu appears to remind them.

Octopocus [5] also provides dynamic guides around the cursor when the user pauses, but is not restricted to radial or other structured layouts. It can handle any arbitrarily shaped gesture, so gesture set designers can include more meaningful gestures, e.g. drawing a *?* for *help*. We hope to provide similar dynamic guides that support creating, not just learning gestures.

Personalizing Gestures

Long et al.'s [14] study of how users design their own gesture sets found that users lack understanding of the recognizer and often add strokes that are too similar to previously defined gestures. Horvitz [10] developed a mixed-initiative system [10] that suggests modifications when a user's gesture is too similar to an existing gesture. Oh and Findlater [18] studied personalized gesture-creation, and developed a design space for procedural modification of existing gestures. Unfortunately, few commercial systems support user-directed gesture creation. Some tablets use multi-finger gestures to perform pre-specified functions, e.g. dragging four fingers to switch applications, but users can only disable this, not modify it.

Gesture Recognition

Rubine's Classifier [20] is probably the most well-known gesture-recognition technique, followed by Dynamic Time Warping (DTW) [22] and Hidden Markov Models (HMM) [26]. The $1 Recognizer [28] facilitates compact and easily-understandable implementation. In addition to identifying a template match, approaches such as the *Gesture Variation Follower* [6] also output continuous measures of deviation from the template. Gesture-Typing interfaces [29] are a special case of gesture recognition. With an installed base of over 100+ million devices, Google's Android Keyboard [8] is probably the most highly used gesture recognizer. We do not focus on gesture recognition, per se, but rather seek recognizer-agnostic techniques.

DESIGN CHALLENGE

Our goal is to help users create personally meaningful gestures that execute commands on mobile phones and tablets. Such gestures must satisfy two competing requirements: Users want gestures that are easy to remember, whereas the system must reliably distinguish gestures from each other. Although techniques such as *Marking Menus* [11] and *Octopocus* [5]

explicitly support learning gestures, neither addresses how to help users *create* or *redefine* those gestures.

As a starting point, we assume that professional designers will design an "optimal" set of gestures, test them for compatibility and memorability, and then hard-code the gesture set into the system, just as they do for keyboard shortcuts. Systems with a small space of pre-existing gestures, such as *Marking Menus*, could be modified via simple configuration dialogs, e.g., *Maya*'s [2] *Marking Menu Editor*. Alternatively, users could modify the mappings via a configuration window that displays pre-existing gestures and their corresponding commands, as in *Scrybe* [9]. However, neither approach helps users add new, personally meaningful gestures to the system.

Users *can* configure shortcut keys, which benefit from a discrete input space where each key can be identified unambiguously. It is easy to verify that mapping the same key combination to two different commands will cause a conflict, and other conflicts are not possible. Gesture input, however, is different. Every time the user performs a particular gesture, it varies, sometimes significantly. The system must compare each gesture to a set of predefined templates, using a specified distance metric.

If the performed gesture is "close" to a gesture template and "far away" from all others, the recognizer can safely assume that the close gesture is the intended one, and execute the associated command. However, errors can occur when gesture templates are very similar to each other. For example, a user may intend to draw a straight line, but curve the line, thus matching an "arc" gesture instead. To avoid this, the set of mapped gestures must be distant enough from each other to prevent accidental misidentification.

Designing and adding new gestures to the system presents several challenges. Each new gesture must be compatible with the existing gesture set, but the distance metrics may be confusing or uninformative. Because two arbitrary shapes likely vary along multiple dimensions, users who lack intimate knowledge of the recognition technique and implementation will have difficulty intuiting how the recognizer works, even when provided with a scalar distance score. The representation of distance between two gestures may be non-linear, depending on both the technique used and any implementation optimizations. Users may even be confused by the system's representation of a single gesture, since it may have been temporally or spatially resampled, normalized, or warped. Recognizer features, such as scale, rotation, and direction sensitivity, may also result in surprising collisions.

DESIGN SOLUTION

We created two dynamic guides that display graphical *feedforward* around the user's finger. Both techniques support interaction between the user and the system's gesture recognizer – each indicates which future directions will result in recognizable gestures. Users can explore this *negative space* of possible gestures to see if gestures they find memorable are also easy for the system to recognize.

Both guides use color to indicate the suitability of a gesture or direction. A gradient from blue to red indicates increasing proximity to existing gestures in the *recognizer space*[1]. Blue gestures are unique; purple ones are ambiguous. Users may draw any gesture they like – neither dynamic guide constrains the user's drawing in any way. The techniques simply reveal information about the uniqueness of each gesture from the recognizer's perspective, and suggest possible completions.

We treat the recognizer as a "black box" with no recognizer-specific heuristics, which lets us support multiple implementations. Both dynamic guides are *recognizer-agnostic*, able to accommodate subsequent improvements to the recognizer, or even changing it entirely, without lowering effectiveness.

Pathward Dynamic Guide

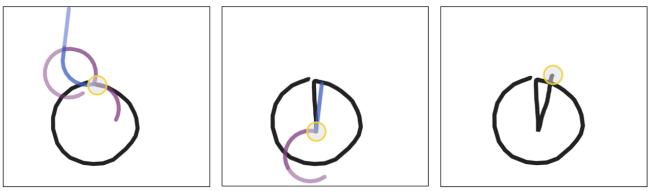

Figure 1. *Pathward* technique: a. The user draws a black circle: three possible completions appear at the touch point (yellow circle). b. The user ignores them and draws downward: two possible completions appear. c. The user follows the blue line upward to create a unique gesture.

The *Pathward* technique is inspired by the *Octopocus* [5] dynamic guide, but with a twist. *Octopocus* shows the *positive space* of existing gesture-command pairs, and how to continue from the current touch point to correctly execute each gesture. We adapted this approach to display samples of possible recognizable gestures, drawn from the *negative gesture space* revealed by the recognizer. Since the number of possible gestures is prohibitively large, samples are constructed by concatenating a random selection of pre-generated gestural atoms consisting of simple lines and arcs. The recognizer evaluates each sample against the existing gesture set, and the set of samples is optionally pruned based on the recognition scores.

Like *Octopocus*, the *Pathward* technique calculates the offset from the gesture's starting point, and advances the displayed gesture suggestions by an equivalent path length (see Fig.1). The remainders of the gesture suggestions are shown originating from the current touch point, indicated by the yellow circle annotations (which are not seen by user). As the user draws, the shapes formed by the drawn gesture and each remaining suggestion are repeatedly passed to the recognizer for evaluation, and the colors of the suggested paths are updated to reflect their evolving recognition scores.

Fieldward Dynamic Guide

The second dynamic guide offers a more holistic view of the negative gesture space, in the form of a dynamic heatmap. While *Pathward* suggests four pre-evaluated gesture completions, *Fieldward* evenly samples the entire display to complete the gesture with vectors originating from the current touch-point (see Fig.2). The score for each completion defines the color at the corresponding point in the background, resulting

[1]The same colors were used for Synaptics Scrybe [9], avoiding problems with the most common types of color blindness.

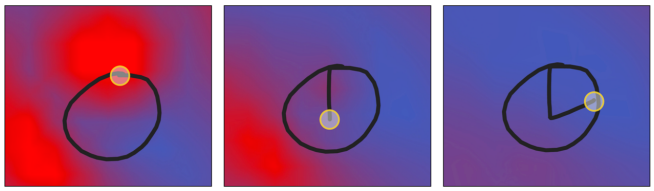

Figure 2. *Fieldward* technique: a. The drawn circle ends in a red zone, indicating the gesture is not unique. b. The user draws downward; then c. moves upward, ending in a blue zone to define a unique gesture.

in a color field that indicates the zones in which a gesture would be unique (blue), ambiguous (purple), or collide with existing gestures (red). An OpenGL shader is used to display even gradients among the sampled points.

Pathward vs. *Fieldward*: Advantages and Disadvantages

The *Pathward* and *Fieldward* techniques address different sets of trade-offs. One advantage of the *Pathward* technique is its ability to represent suggested gestures of arbitrary complexity and length, including curves, loops, and closed figures. This is especially important for users who do not think to explore certain aspects of the gesture design space. For example, our pilot tests revealed that many users prefer drawing closed paths and figures, but we cannot tell whether they explicitly rejected open figures or simply did not consider them.

A major disadvantage of *Pathward* technique is that, as observed with *Octopocus* [5], users can only interpret a limited number of gestures before becoming confused by "visual clutter". Some of this can be mitigated by varying the opacity of the suggested gestures, using high opacity only for the sections that immediately follow the touch point, with low opacity for the remainder of each gesture suggestion. Even so, we chose to limit the number of suggested gestures to four, to ensure that the user could easily distinguish among them.

An additional drawback is the arbitrary nature of the suggested gestures. Although we could certainly develop a statistical description of each user's gestures in order to make personalized suggestions, it is not clear whether the user would be better served with stylistically similar or different suggestions. The *Pathward* suggestions are thus limited to a technical exploration of the negative gesture space from the system's perspective, where each suggested shape is arbitrary and is not pre-associated with any meaning.

The *Fieldward* technique avoids several disadvantages of the *Pathward* technique. First, since feedforward is displayed as background gradients instead of discrete objects, an arbitrarily high-resolution sampling of the gesture space does not result in visual clutter, nor does it distract from the drawn gesture. In practice, sampling resolution is limited only by computational cost and the need to run the technique at interactive framerates.

In addition, users may be confused or frustrated if *Pathward* suggests gestures that they judge to be stylistically or cognitively inconsistent with the problem domain, the set of existing gestures, or the user's personal preferences. By contrast, the *Fieldward*'s ambient display avoids suggesting any particular gesture style, shape, or other visual object.

However, these advantages are offset by certain costs. The most important is the temporal limitation: unlike *Pathward*, the *Fieldward* technique can only display one linear step into the future of the current gesture. The benefits of *Fieldward*'s high spatial resolution are matched by a corresponding loss in the *order* (cf. polynomial expressions) of the suggestions, since it cannot show the effects of projected curves, loops or even additional line segments beyond the first one. Also, users must remember that recognizable gestures should *end* in a blue zone, but may traverse red zones in the process.

EXPERIMENT DESIGN

We ran an experiment to compare the *Pathward* and *Fieldward* techniques, with a *No Feedforward* control condition. Although these techniques can work on any touch display, we focus on mobile devices, which offer compelling applications for personalized gestures. Our research questions include:

1. *Can dynamic guides help participants create memorable gestures that are also easy for the system to recognize?*
2. *Do participants prefer Fieldward or Pathward?*
3. *Which strategies do participants use to define personal gestures, and are they affected by the choice of technique?*

Methodological Challenges

The experimental design was challenging, causing us to run a series of pilot studies before settling on the two-part approach described in the Method section below. We ran the first pilot on a tablet, but found that participants used the large screen space to create very long gestures, making them more likely to be unique. We decided to switch to smartphones, since the smaller screen size significantly increased the difficulty of the task, which also increased the need for a dynamic guide. Inspired by situations that require one-handed interaction, such as standing on a crowded bus or carrying groceries, we asked participants to draw each gesture with the thumb of one hand.

Another problem stemmed from our initial use of "invocable" guides, in which the feedforward only appears if the user hesitates. Some pilot test users never invoked feedforward, making it impossible to compare techniques. For the experiment, we decided to turn on the dynamic guide at touch down, to ensure consistency across conditions; in a real application, we would restore the delay so that experts could proceed with no guide.

We designed the dynamic guides so users could choose the desired level of "recognizability" of their gestures, that is, how sloppily the gesture could be performed and still be recognized correctly. In the first pilot study, participants applied different criteria to determine if recognition was "good enough", and many did not understand what "recognizable" actually meant; for the experiment, we chose a specific recognition threshold.

Early pilot tests were excessively long, up to two hours, and exhausted the participants. So we sought alternatives that would reduce the time to an hour while ensuring sufficient task difficulty. We explored starting with an existing gesture set, to which participants added new gestures. The task was immediately difficult, since each stored gesture increases the likelihood of a collision. Unfortunately, providing users with existing gestures influences their strategies for creating new

gestures in subtle and unmeasurable ways. In the experiment, we asked participants to create the gesture set from scratch, starting with six gestures in the initial practice block.

Early pilot studies used a within-participants design, in which users were shown a command name and asked to generate an "easy-to-remember" gesture, register it and later recall it. We counter-balanced the feedforward techniques for order, and tried various block sizes, ranging from 1 to 12, rotating through each of the three techniques: *Fieldward*, *Pathward* and *No Feedforward*. Unfortunately, we found that the two feedforward techniques were highly confounded: a technique from a previous block sometimes revealed subtle information about gestures already present in the gesture set, and influenced participants in unmeasurable ways. For example, a participant in the *No Feedforward* condition might use a gesture suggestion from the previous *Pathward* condition. This carry-over meant that a within-participant design could not untangle the effect of the current condition from previous conditions.

Another problem was that participants learned at different rates, and faced problems generating gesture ideas at different points in the study. Many participants avoided using dynamic guides when the gesture set was almost empty, since gestures were recognized reliably anyway. They started to use guides only after they "got stuck", usually after about a dozen gestures. Other participants developed strategies that worked for a few trials, after which they had to come up with a new strategy.

The high variability in learning and gesture-creation strategies, independent of the techniques, led us to consider a between-participants experimental design. Exposing three groups of participants, each to a different technique (*Fieldward*, *Pathward* and *No Feedforward*), provides a "clean" comparison of the characteristics of each technique. However, it does not allow us to determine which technique users might prefer had they been exposed to all of the techniques.

Finally, we considered alternative control conditions, for example displaying the existing gestures. However, providing information about registered gestures only in the control condition would strongly bias the participants; and providing it in all conditions would weaken our ability to measure the effects of the feedforward techniques. Further, since we deal with multiple examples of each user-defined gesture, there is also no single "correct" version to display.

Method

Based on the above considerations, we decided on a two-part experimental design. Part One treats the feedforward technique – *Pathward*, *Fieldward*, or *No Feedforward* – as a between-participants factor, whereas Part Two treats them as a within-participants factor. We believe this offers a reasonable compromise: The between-participant blocks let us compare performance measures across techniques without confounding or learning effects; the within-participant blocks let participants try all three techniques when the size of the gesture set is large enough that the task consistently benefits from a dynamic guide. We collect all the same data for both Parts One and Two, but the analysis for Part Two focuses on the qualitative data, where participants compare the techniques to each other.

Participants

We recruited 27 participants (15 men, 12 women) ages 22-40. Three were left-handed; one right-handed participant preferred using her left hand for the experiment.

Hardware and Software

We used two LG Nexus 5 (4.95" display) and two LG Nexus 5x (5.2" display) smartphones. The *Pathward* and *Fieldward* techniques are implemented in Java and integrated into an application for the Android platform (see Fig. 3). Touch events (*down*, *move*, *up*) from the Android system are used to build a vector that describes the current gesture. With each *move* event, gesture completion candidates are appended to the user's gesture and the stock Android gesture recognizer is used for calculating distances between each candidate gesture and the existing template gestures. This implementation supports only single-touch, unistroke gestures.

Depending on its configuration for sensitivity to rotation and direction of gestures, the stock Android recognizer uses different techniques for calculating inter-gesture distance. We used the default configuration (sensitive to both rotation and sequence) where the recognizer returns a score based on the inverse of the minimum cosine distance between two-dimensional gesture representations that have been pre-rotated and normalized. Due to optimizations in the recognizer and the particular internal representation, the scores are difficult to understand intuitively. However our pilot test results indicated that a score lower than 1.5 is sufficient for adding a new gesture with no collisions.

Procedure

Task description

A key application for dynamic guides is to help users create custom gesture-command combinations for their mobile phones. We envision a scenario in which a phone manufacturer provides a set of gestures for accessing common commands. For example, a *?* gesture might be assigned to *Help*. Users could then redefine these gestures, e.g. assigning a thunderbolt gesture to *Get Weather Report*. Users could also create new gesture shortcuts for personally useful commands, e.g. *Call Mom*, or *Post this photo on Facebook*. These gestures can be performed eyes-free on a watch or smartphone; such as sending a custom message in reply to a call during a meeting.

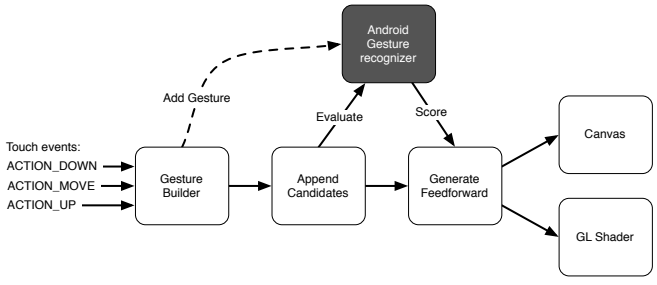

Figure 3. Simplified system diagram: With each *move* event, completion candidates are appended to the user's gesture and passed to the recognizer for evaluatino. Recognition scores are used to generate feedforward using an OpenGL shader (*Fieldward*) or Canvas functions (*Pathward*). Successful gestures are added to the recognizer's internal store.

Command Category	Command Examples
Communication: Family	Call Mom Group-chat Friends
Communication: Social-Media	Facebook share Check #TrafficAlert
Context: Media	Dropbox PDF Repeat Song
Context: Location	Show location Navigation On
Shortcuts: Apps	Tweet photo Take Selfie
Shortcuts: OS	Airplane-mode On Dismiss Notifications

Table 1. Left: Trial commands are drawn from six command categories. Right: Examples of specific trial commands.

Based on this scenario, we designed an ecologically valid task for the experiment. Table 1 (left column) shows three categories of common mobile phone commands: *communication*, *context* and *shortcuts*, each with two subcategories. We created six phrases from each subcategory, consisting of a verb-noun combination, such as *Publish video*, and a longer description, (*Publish current video on Youtube*) (see Fig.4a).

Trial description

Each trial begins by presenting the participant with a brief command, e.g. *Call Mom*, initially shown in black (see Fig.4b). The participant designs a gesture associated with that command that they can easily remember but that is still recognizable by the system. Participants may create any gesture they like, except for letters and numbers.

In the *Pathward* or *Fieldward* conditions, the associated feedforward technique is activated on touch down, and disappears on touch up. No feedforward appears in the *No Feedforward* condition. All three conditions provide recognition feedback by changing the color of the currently drawn command. If the command is recognizable, i.e. with an inter-gesture distance score above 1.5, the command label turns red, otherwise it turns blue. If the participant approves the gesture, she can register it, by re-drawing the same gesture two more times, without feedforward. This ensures that the recognizer has three good samples of the gesture. Each recording increments the number in the gray "REGISTER #/3" box (see Fig.4).

If the participant is not satisfied with the gesture at this stage, she can try a new gesture, as many times as she likes. The trial ends when the participant has successfully registered three recognizable samples of the gesture. If the participant cannot successfully register a second or third sample of the gesture, she can press a button to move on to the next command.

Experiment overview

Figure 5 shows the design of the overall experiment, which is divided into two parts. Each part consists of a practice block, followed by three experimental blocks. Each block consists of six trials in which the participant is presented with a command and asked to create a new gesture. Each experimental block contains an example from each of the six categories in Table 1, counterbalanced for order within and across blocks.

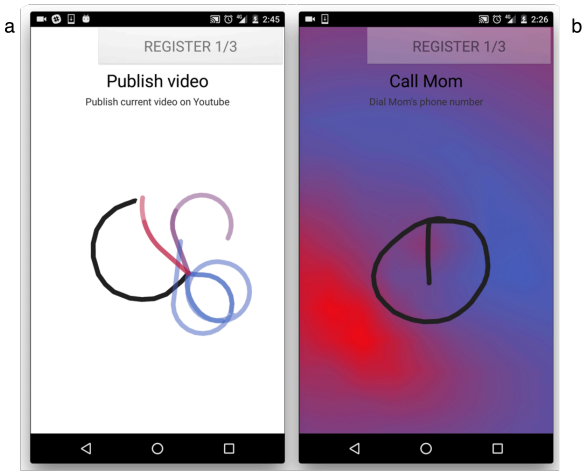

Figure 4. Dynamic guides help users create recognizable gestures. a. *Pathward* suggests four paths: *blue* paths successfully complete the user's *black* gesture, *red* paths collide with existing gestures. b. *Fieldward* displays an interactive heatmap: linear completions ending on *blue* are recognizable, and on *red* cause collisions. *Purple* paths and areas are ambiguous.

Part One: Between-Participants Design

Part One uses a between-participants design with one factor: TECHNIQUE (*Path*, *Field*, *None*). Participants are randomly assigned to one of three groups, each with a different feedforward technique.

All participants begin with practice block *a*, which serves as the initial practice session. The experimenter first explains the general goals of the study and then demonstrates how to generate and record a gesture that is recognizable. Participants are asked to hold the phone with only one hand and perform all gestures with their thumb; they are invited to keep their free hand busy (e.g. by holding a pen) to ensure they only use one hand to perform the gestures. The experimenter asks the participant to copy and register four new gestures, the first two with *No Feedforward* and the others with the assigned technique – *Path*, *Field*, *None* – explaining how the technique works. Participants are then asked to create and register two of their own gestures using the assigned feedforward technique. The practice block poses the easiest gesture-creation task, since the gesture set is initially empty, making the first few gestures easily distinguishable. The gesture set in subsequent experimental blocks includes these six preliminary gestures.

Next, each group of participants receives three blocks of the same technique: A is assigned *Fieldward*, B is assigned *Pathward*, and C is the control *No Feedforward* technique. Each participant performs three experimental blocks of six trials, to create a total of 18 new gestures. At the end of each block, participants are tested on their ability to remember the last six gestures they created in order to prevent them from simply creating random scribbles. Commands are presented in random order and the participant is asked to draw the associated gesture, with no feedforward. The message "OK!" appears if the recognizer matches the gesture with the right command; otherwise "wrong gesture" appears.

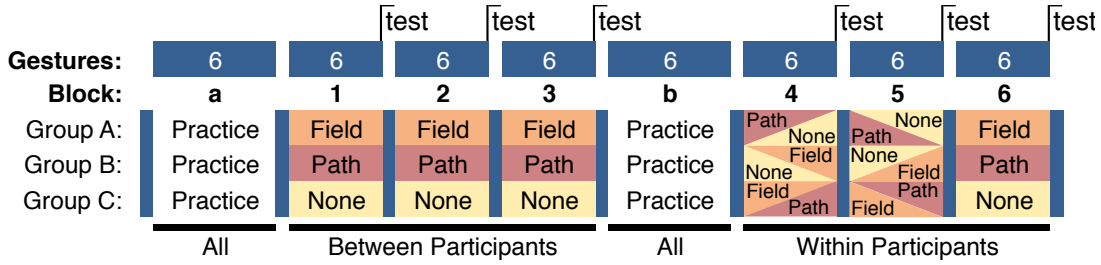

Figure 5. Experiment design: Experimental blocks 1-3 are allocated *between* participants. Experimental blocks 4-6 are allocated *within* participants, with blocks 4 and 5 counter-balanced with a latin square across participants. Block 6 repeats the technique in blocks 1-3. Each experimental block includes six trials that produce six new gestures, followed by a test.

At the end of Part One, each participant has added a total of 24 gestures to their gesture set: six from the practice block and 18 from the three experimental blocks. After finishing block 3, participants answer a questionnaire, and may take a break before beginning Part Two.

Part Two: Within-Participants Design

Participants are assigned the two remaining feedforward techniques from Part One in blocks 4 and 5, counter-balanced for order across participants according to a Latin square. Block 6 presents the same technique as in blocks 1-3. This produces an ABCA design that lets participants compare the three techniques at a point when the task has become reliably difficult.

All participants begin with practice block *b*, which introduces the two remaining feedforward techniques. This practice block uses a different gesture set, to prevent confounding with the participant's existing gesture set. The first three trials of the practice block show one new technique, the last three show the other. The experimenter explains each new technique and asks the participant to use it to copy and register a new gesture. Participants then create two additional gestures, using the same technique. When practice block *b* is finished, the six gestures are deleted and the 24 gestures from Part One are restored.

Each participant performs three experimental blocks of six trials to create a total of 18 additional gestures. At the end of each block, participants are tested on their ability to remember the last six gestures they created. At the end of the experiment, each participant has created a total of 36 gestures under experimental conditions, plus the six created during practice block *a*. After finishing block 6, participants answer a questionnaire.

Data Collection

During the practice and experimental blocks, the smartphone runs a screen recorder that saves video of every performed gesture. The experiment log includes timestamps of the start and end of each trial, as well as the touch events for every gesture tried by the participant, the gesture length in pixels, the drawing time in milliseconds, and the list of inter-gesture distance scores from the recognizer. From this raw data we can extract the number of *recognizable* gestures performed, the number of *non-recognizable* gestures performed (FAILED ATTEMPTS), the number of *recognizable* gestures that users performed but decided not to register (USER-REJECTED GESTURES), and the number of successful recall tests (RECALL ACCURACY).

At the end of Part One, participants are given a questionnaire with a list of all 18 commands and are asked to describe the strategies they used to make their gestures memorable. During Part Two, participants see a list of 6 commands at the end of each block. As in Part One, they describe the strategies they used to make their gestures memorable. They then answer four Likert-style questions to assess each technique (see Quantitative Self-Reported Measures). At the end of Part Two, they choose the most helpful technique and explain their choice.

We took observational notes during all sessions and debriefed participants after the final trial, with a particular focus on what the participants liked and disliked about the techniques and about their strategies for creating memorable gestures. We consulted their written strategy descriptions to supplement our understanding when necessary. The salient themes presented in the Qualitative Results section emerged from regular discussions of the data among the research team, with frequent checks back to the source data.

RESULTS

Quantitative Performance Measures

We collected a total of 2916 experimental trials (27 PARTICIPANT x 3 TECHNIQUE x 36 TRIALS). As described earlier, we restrict the statistical analyses to the 1458 trials of between-participants' data from Part One. Similarly, we analyse only the questionnaire responses from Part Two.

Using ANOVA, we first determined that there were no unwanted significant effects from the control variables (command category, command examples). We then ran a 3x3 mixed analysis of variance with factors TECHNIQUE × BLOCK, followed with Tukey HSD tests for post-hoc comparisons when warranted.

FAILED ATTEMPTS: We use FAILED ATTEMPTS to examine gesture creation over time (see Fig. 6). We found a main effect of BLOCK ($F_{2,48} = 8.617$, $p = .0006$). Block 1 (mean=2.69) produced significantly fewer failed attempts of creating a new recognizable gesture than Block 2 (mean=4.36) and than Block 3 (mean=5.019) (both $p < .05$). In terms of TECHNIQUE, across all blocks, participants using the *Field* technique (mean=3.05) failed fewer trials than those using *Path* (mean=3.92) and *None* (mean=5.10). However, the differences were not significant ($F_{2,24} = 1.504$, $p = .242$).

GESTURE LENGTH: We found a main effect of BLOCK ($F_{2,48} = 15.27$, $p < 0.0001$) revealing that gestures get longer

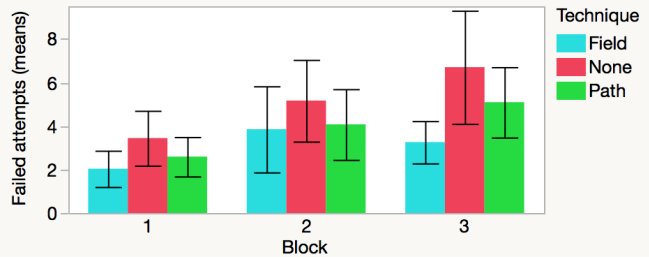

Figure 6. FAILED ATTEMPTS **represents a proxy measure for the probability of gesture collisions in the recognizer and thus task difficulty. Bars indicate 95% confidence intervals.**

over time. Those created in Block 1 (mean=2219 pixels) are significantly shorter than those in both Blocks 2 (mean=2587 pixels) and 3 (mean=2766 pixels) (both comparisons $p < .05$). A main effect of TECHNIQUE ($F_{2,24} = 5.53$, $p = .01$) reveals that gestures created with *Field* (mean=2949 pixels) are significantly longer than those created with *None* (mean=2338 pixels) and *Path* (mean=2285 pixels) (both $p < .05$). However, an interaction effect between BLOCK and TECHNIQUE ($F_{4,48} = 4.14$, $p = .0058$) clarifies that gestures created with *Field* only begin to get significantly longer than those with *None* and *Path* starting in Block 2 (both $p < .05$).

USER-REJECTED GESTURES: Participants using the *Field* technique (mean=1.2) rejected more gestures compared to the *None* (mean=.7) technique. Participants in the *Path* condition fell in between (mean=.9). However, these differences were only at trend level ($F_{2,24} = 3.6$, $p = .07$). This requires follow-up research.

RECALL ACCURACY: We found no significant differences ($F_{2,24} = .26$, $p = .77$) in recall accuracy among *Field* (mean=1.97), *None* (mean=2.08) and *Path* (mean-1.86).

Quantitative Self-Reported Measures

In Part Two, participants were asked to rate four statements on a 5-point Likert scale, from strongly disagree to strongly agree. The statements asked whether the current technique helped them to: A) think of new gestures, B) discover recognizable gestures, C) discover memorable gestures, and D) adapt my memorable gestures to make them recognisable. Three questions resulted in significant differences across technique, based on analysis using a Friedman test:

Question/Technique	*Field*	*Path*	*None*
A) Think of new gestures	4	3	3
B) Discover recognizable gestures	4	3	2
C) Discover memorable gestures	3	2	3
D) Adapt my memorable gestures to make them recognizable	4	3	3

Table 2. Medians of the answers from the 5-point Likert-scale questions about how much each technique helped on the listed aspects.

B. DISCOVER RECOGNIZABLE GESTURES: Participants showed significantly stronger agreement ($p = .019$) for *Field* compared to *None*; *Path* was in between but was not significantly different from the other two: $\chi^2(2) = 9.16, p = .01$.

C. DISCOVER MEMORABLE GESTURES: Participants showed significantly stronger agreement ($p = .016$) for *None* compared to *Path*; *Field* was in between but not significantly different from the other two: $\chi^2(2) = 10.54, p = .005$.

D. ADAPT MEMORABLE GESTURES TO MAKE THEM RECOGNIZABLE: Participants showed significantly stronger agreement for *Field* compared to *Path* ($p = .008$) and to *None* ($p = .005$): $\chi^2(2) = 15.02, p = .001$.

Finally, participants picked the technique they most preferred for creating memorable gestures. *Field* was most popular (15), although many liked *None* (9). Very few preferred *Path* (3).

Qualitative Results and Discussion

In general, we found the qualitative findings were consistent and complementary to the quantitative results.

Participants value memorability over recognizability

The most common strategy for creating gestures consisted of thinking of a memorable shape and then, if necessary, tweaking it to make it recognizable. For example, in the *Fieldward* condition, P11 drew a smiley face for *Play CandyCrush*, and extended the line representing the face until it showed blue under her thumb. Some users did not mind that the extra segment had no particular meaning – they just expected to memorize it. For example, P4 used *Fieldward* to design a gesture for *Group-chat friends* (see Fig.7): *I started with a circle (remembering my circle of friends) and then went to a [blue] corner to make it distinguishable from other gestures.*

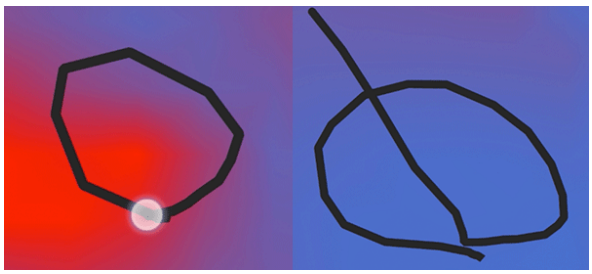

Figure 7. P4 first tried a circle for *Group-chat Friends*, and used *Fieldward* to continue the line towards a blue zone .

By contrast, other participants clearly wanted the extra segment to have meaning. For example, P11 using *Pathward* to design a gesture for *Next Episode* said: *Drawing a play button (triangle) [alone] didn't work. Then I used the [triangle] button followed by a path to add a little circle. In my mind that meant 'next thing'* (see Fig.8).

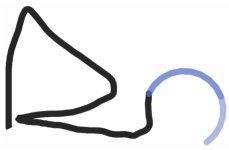

Figure 8. P11 added his own meaning ("the next thing") to the add-on segment suggested by *Pathward* for the *Next Episode* command.

Participants who had a particular gesture in mind would often extend it to make it recognizable, which explains why the *Fieldward* technique produced the longest gesture lengths. *Fieldward* supports this strategy particularly well: when the task got the hardest (final block), *Fieldward* produced the longest gestures but at the same time had the fewest failed

attempts. That *Fieldward* was also the preferred technique, despite the longer gesture lengths, indicates that participants do not necessarily favour shorter gestures. However, this may also be an artifact of the experimental setting. Over time, if users must frequently execute gestures that are long, they might decide to redefine them to make shorter versions.

In the questionnaire, participants ranked *None* highest for helping find memorable gestures, in contrast to *Fieldward* that ranked highest for discovering recognizable gestures. This suggests a tension between efficiently finding recognizable gestures and remembering them: *Though it takes several times to figure out a recognizeble gesture, it would be easier for me to remember the gestures I created independently* (P21).

Fieldward is preferable to Pathward

Participants appreciated the open-ended, flexible nature of *Fieldward*, which supported thinking of a gesture first, then considering recognizability. They also felt it gave them greater creative control. P8 observed that: *Fieldward is free enough to help you to create a figure and remember it. (...) It's hard to follow a path and remember the figure you made.* P7 noted this limitation with *Pathward*: *I get that the lines [in Pathward] wanted to help me, but ... but they help you 'step by step', and I had a complete gesture idea in my mind.* Note that professional gesture designers might feel differently if their goal is to create coherent sets of gestures, such as with *Flower Menus*. The *Pathward* technique could be adapted to follow patterns specific to a particular goal or application.

From a usability standpoint, *Pathward* suffered more from occlusion than *Fieldward*. Given the number of radiating paths on the small screen, participants sometimes found it difficult to see the available paths, or even the one they were actively trying to follow. P11 said: *The path was difficult to see and follow sometimes, as it was occluded by my drawing. As the Fieldward was on the background, the feedback was easier to understand.*

However, the *Fieldward* technique was not entirely immune to occlusion. Occasionally, a small red zone might be hidden by the finger. If the surrounding zone is blue, the user has the false sense that the gesture is recognizable, and only realizes the problem on touch up, when the gesture is rejected. This could be corrected easily by displaying the colors hidden below the finger as a small halo that surrounds it.

Feedforward techniques act as temporary scaffolding

Although the recognizer was treated as a black box, some participants were able to use the feedforward techniques to discern aspects of how it works. Some participants picked up "tricks" from each technique, such as copying *Pathward*'s tendency to add a curl to a gesture, or backtracking to retrace the most recent segment, as revealed by *Fieldward*. Some participants treated feedforward as a temporary scaffolding, after which they felt sufficiently confident to no longer need it. For example, P7 used *No Feedforward* for *Check #TrafficAlert*: *I represented a Traffic mess by doing a lot of circles on top of each other. I remembered from using Fieldward that it looked like a feasible option.* P10 said: *In the beginning, the Fieldward was helpful to understand how to create recognizable gestures, but once I learned, I could do fine with No Help [No Feedforward].* These participants also indicated a preference for *No Feedforward* at the end of the study.

Different strategies adopted to support memorability

We observed several interesting strategies, beyond starting with a memorable gesture and tweaking it. A few participants created their own grammar that they applied to subsets of related commands. For example, P3 drew a curl gesture to represent sharing (*Forward parents*), and a winky eye to represent a picture (*Take selfie*). He then re-used both gestures as components of other compound commands involving either pictures or sharing something (*Tweet Photo*) (see Fig.9). P3 noted: *I developed a kind of system of images that makes sense for me, and I tried to keep it consistent during the process.*

Figure 9. P3 developed his own gesture grammar on the fly. a. *Forward parents*: Envelope, with a curl to indicate sharing. b. *Take selfie*: Winky eye for "picture". c. *Tweet Photo*: Winky eye with a curl gesture for sharing a picture.

In a few cases, participants selected a gesture suggested by a *Fieldward* technique and then – after the fact – assigned a meaning to the result. For example, P20 described his strategy for *Take Selfie*: *I followed the blue [field] and found a gesture that looked like a four, so that helped me to remember it.* Sometimes participants tried and missed drawing a particular gesture. Rather than discarding it, they would designate a meaning and keep it. For example, P3 wanted to draw a network of three connected circles for *Facebook share*. After drawing it, reminded him of male genitals. He still decided to keep the gesture, since he was not very fond of *Facebook*.

Figure 10. Different mnemonics for *Close all* include: a. Cross (common for Western users) b. Crossed circle (common for Asian users) c. Four "swipe right" gestures (Android gesture to close apps) d. "W" (MacOS keyboard shortcut, *<CMD>-W*, to close windows).

We also observed several cultural differences (see Fig. 10). For example, most Western users represented *Close all* with an 'X', whereas most Asian users represented it with a crossed circle. Some gestures reflected previous interactions with computers or smartphones. For example, P13 described his gesture for *Close all*: *4 swipes to the right as if I was closing the apps on Android*. For the same command, P7 based her gesture on a related keyboard shortcut: *In my computer I close apps with <CMD>-W so I did something similar to a W.*

For commands involving other people, participants created gestures that represent their personal relationships. For example, *Call Mom* evoked a diverse set of gestures: P24: *When I call mom I'm very happy. I did a circle for my face, and then*

another curve for my smile. P5: *It's a bag because my mom likes bags.* P27: *A bow, that was the Charades code for my Mom when I was young.*

The results with respect to the research questions posed earlier can be summarized as follows:

1. *Can dynamic guides help participants create memorable gestures that are also easy for the system to recognize?*

   Yes. Both techniques help users discover recognizable gestures, with *Fieldward* helping significantly more than *No Feedforward* and *Pathward* in between. *Fieldward* is significantly more helpful than *Pathward* for adapting memorable gestures to make them recognizable, with *No Feedforward* in between.

2. *Do participants prefer Fieldward or Pathward?*

   Participants clearly preferred *Fieldward* for most qualitative and quantitative measures. Surprisingly, some participants preferred *No Feedforward* to *Pathward*, which was only preferred when participants sought ideas for new gestures.

3. *Which strategies do users use to define personal gestures, and are they affected by the choice of technique?*

   Participants prefer to create a gesture they find memorable, and then adapt it to make it recognizable, with the help of the dynamic guide. In some cases, the dynamic guide can also inspire new strategies for creating gestures, or reveal aspects of how the recognizer works.

Discussion

Fieldward preferred, but No Feedforward also well liked
While *Fieldward* was the most preferred technique, *No Feedforward* also had reasonable support (9/27). Some of the users who preferred *No Feedforward* were those who no longer desired or needed the feedforward scaffolding, as noted above. For others it seemed that they simply preferred complete autonomy over the process. Creating gestures with no support (*No Feedforward*) resulted in more failed trials (initial attempts) , but also gestures that were shorter in length. This appears to have been a reasonable trade-off for some users. For example, P5 preferred *No Feedforward* and explained: *While using no help, I adapted my gestures by trying variations on how to draw the same shape. Adapting the gesture with the field lead [sic] to finding a recognizable gesture similar to my idea faster, but it lead [sic] to more complicated gestures.*

Need to support individual differences
We observed many individual differences in strategies and preferences, indicating the lack of a one-size-fits-all solution. However, users do not need one: providing options for different types of dynamic guides would allow users to take advantage of them only only when needed.

Gesture creation in the wild
If users had gesture-creation capabilities on their devices today, we doubt that many would create 42 gestures in one sitting, as imposed by this experiment. Instead, they would probably add gestures as needed, and modify them later if they were difficult to remember. Although we did not see an effect of technique on immediate recall accuracy in our study, more research "in the wild" is needed to determine their effects on gesture recall over time. Similarly, it will be interesting to see which users take advantage of gesture creation. Will they be mostly "power users" (who create their own shortcuts), or will this be adopted by a broader audience? Finally, how important is gesture length in the wild? We expect that users will not notice length for infrequent gestures, but will probably optimize for length for frequent ones. Even so, the ability to create personally meaningful gestures likely outweighs slightly greater lengths. A field deployment is necessary to clarify these issues.

Hybrid approach of designers and users working together
Although the experiment focused on participants who are novices with respect to understanding gesture recognition, the feedforward techniques themselves may also be of interest to app designers who want to incorporate gesture-based interaction. Designers could generate an initial set of command-gesture pairs and then allow users to personalize those gestures. For example, imagine an app designed for a noisy concert setting that relies on eyes-free, one-handed interaction to vote for the current band or song. Users could add personal gestures to perform personalized actions suited to that setting.

CONCLUSION AND FUTURE WORK

Our goal is to help users create their own gestures on mobile devices, such that they are both personally memorable and reliably recognized by the system. We introduced a novel type of dynamic guide that visualizes the negative space of possible gestures as the user interacts with the system. We implemented two novel feedforward techniques *Fieldward* and *Pathward*, which address different design trade-offs.

Participants strongly preferred the *Fieldward* technique, which offers a more open-ended picture of the immediate future relative the current state of the user's gesture. In contrast, *Pathward* provides shorter, more efficient gestures, that are easy to recognize, but not always easy for the user to remember.

We conducted both a between-participants and a within-participants experiment to compare these two techniques with a non-feedforward condition. We found that users discovered a wide variety of strategies for creating memorable gestures in partnership with the system, especially when using the *Fieldward* technique.

In the future, we would like to conduct a longitudinal study to see how people evolve their gestures over time. We also plan to investigate extensions to our implementations for supporting multi-touch and multi-stroke gesture input. The two feedforward techniques serve as the foundation for a series of techniques that will allow app developers to enable rich end-user personalizations to their mobile apps, and end users to create their favorite gesture shortcuts within particular app or across different applications.

ACKNOWLEDGMENTS

This work was partially supported by European Research Council (ERC) grant n° 321135 CREATIV: Creating Co-Adaptive Human-Computer Partnerships. We thank Theophanis Tsandilas for his invaluable statistics help, and Thibault Raffaillac for his implementation of early prototypes.

REFERENCES

1. Caroline Appert and Shumin Zhai. 2009. Using Strokes As Command Shortcuts: Cognitive Benefits and Toolkit Support. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 2289–2298. DOI: http://dx.doi.org/10.1145/1518701.1519052

2. Autodesk Inc. 2016. Autodesk Maya 2016 Documentation. (2016). Available: http://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=GUID-9C43FD30-D92F-4035-9820-4C3FFDD8E211, Accessed April 12, 2016.

3. Gilles Bailly, Eric Lecolinet, and Laurence Nigay. 2007. Wave Menus: Improving the Novice Mode of Hierarchical Marking Menus. In *Human-Computer Interaction - INTERACT 2007 (LNCS)*, Cécilia Baranauskas, Philippe Palanque, Julio Abascal, and Simone Diniz Junqueira Barbosa (Eds.), Vol. 4662. Springer-Verlag, Berlin / Heidelberg, 475–488. DOI: http://dx.doi.org/10.1007/978-3-540-74796-3_45

4. Gilles Bailly, Eric Lecolinet, and Laurence Nigay. 2008. Flower Menus: A New Type of Marking Menu with Large Menu Breadth, Within Groups and Efficient Expert Mode Memorization. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI '08)*. ACM, New York, NY, USA, 15–22. DOI: http://dx.doi.org/10.1145/1385569.1385575

5. Olivier Bau and Wendy E. Mackay. 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 (UIST '08)*. ACM, New York, NY, USA, 37–46. DOI: http://dx.doi.org/10.1145/1449715.1449724

6. Baptiste Caramiaux, Nicola Montecchio, Atau Tanaka, and Frédéric Bevilacqua. 2014. Adaptive Gesture Recognition with Variation Estimation for Interactive Systems. *ACM Trans. Interact. Intell. Syst.* 4, 4, Article 18 (Dec. 2014), 34 pages. DOI: http://dx.doi.org/10.1145/2643204

7. Emilien Ghomi, Stéphane Huot, Olivier Bau, Michel Beaudouin-Lafon, and Wendy E. Mackay. 2013. Arpège: Learning Multitouch Chord Gestures Vocabularies. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces (ITS '13)*. ACM, New York, NY, USA, 209–218. DOI: http://dx.doi.org/10.1145/2512349.2512795

8. Google Inc. 2016. Google Keyboard. (2016). Available: https://play.google.com/store/apps/details?id=com.google.android.inputmethod.latin, Accessed April 12, 2016.

9. Matthew Guay. 2010. Superpower Your Touchpad Computer with Scrybe. *How-To Geek* (May 2010). Available: http://www.howtogeek.com/howto/16587/superpower-your-touchpad-computer-with-scrybe/, Accessed April 12, 2016.

10. Eric Horvitz. 1999. Principles of Mixed-initiative User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. ACM, New York, NY, USA, 159–166. DOI: http://dx.doi.org/10.1145/302979.303030

11. Gordon Kurtenbach and William Buxton. 1991. Issues in Combining Marking and Direct Manipulation Techniques. In *Proceedings of the 4th Annual ACM Symposium on User Interface Software and Technology (UIST '91)*. ACM, New York, NY, USA, 137–144. DOI: http://dx.doi.org/10.1145/120782.120797

12. Gordon Kurtenbach and William Buxton. 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 (CHI '93)*. ACM, New York, NY, USA, 482–487. DOI: http://dx.doi.org/10.1145/169059.169426

13. Yang Li. 2010. Gesture Search: A Tool for Fast Mobile Data Access. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology (UIST '10)*. ACM, New York, NY, USA, 87–96. DOI: http://dx.doi.org/10.1145/1866029.1866044

14. Allan Christian Long, Jr., James A. Landay, and Lawrence A. Rowe. 1999. Implications for a Gesture Design Tool. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. ACM, New York, NY, USA, 40–47. DOI: http://dx.doi.org/10.1145/302979.302985

15. Dan Mauney, Jonathan Howarth, Andrew Wirtanen, and Miranda Capra. 2010. Cultural Similarities and Differences in User-defined Gestures for Touchscreen User Interfaces. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems (CHI EA '10)*. ACM, New York, NY, USA, 4015–4020. DOI: http://dx.doi.org/10.1145/1753846.1754095

16. Meredith Ringel Morris, Jacob O. Wobbrock, and Andrew D. Wilson. 2010. Understanding Users' Preferences for Surface Gestures. In *Proceedings of Graphics Interface 2010 (GI '10)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 261–268. http://dl.acm.org/citation.cfm?id=1839214.1839260

17. Miguel A. Nacenta, Yemliha Kamber, Yizhou Qiang, and Per Ola Kristensson. 2013. Memorability of Pre-designed and User-defined Gesture Sets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 1099–1108. DOI: http://dx.doi.org/10.1145/2470654.2466142

18. Uran Oh and Leah Findlater. 2013. The Challenges and Potential of End-user Gesture Customization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 1129–1138. DOI: http://dx.doi.org/10.1145/2470654.2466145

19. Halla Olafsdottir and Caroline Appert. 2014. Multi-touch Gestures for Discrete and Continuous Control. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces (AVI '14)*. ACM, New York, NY, USA, 177–184. DOI: http://dx.doi.org/10.1145/2598153.2598169

20. Dean Rubine. 1991. Specifying Gestures by Example. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '91)*. ACM, New York, NY, USA, 329–337. DOI: http://dx.doi.org/10.1145/122718.122753

21. Jaime Ruiz, Yang Li, and Edward Lank. 2011. User-defined Motion Gestures for Mobile Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 197–206. DOI: http://dx.doi.org/10.1145/1978942.1978971

22. H. Sakoe and S. Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26, 1 (Feb 1978), 43–49. DOI: http://dx.doi.org/10.1109/TASSP.1978.1163055

23. Statistica. 2014. Number of smartphone users worldwide from 2014 to 2019 (in millions). (2014). Available: http://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/, Accessed April 11, 2016.

24. Craig Villamor, Dan Willis, and Luke Wroblewski. 2010. Touch Gesture Reference Guide. (April 2010). Available: http://static.lukew.com/TouchGestureGuide.pdf, Accessed April 12, 2016.

25. Emanuel von Zezschwitz, Paul Dunphy, and Alexander De Luca. 2013. Patterns in the Wild: A Field Study of the Usability of Pattern and Pin-based Authentication on Mobile Devices. In *Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '13)*. ACM, New York, NY, USA, 261–270. DOI: http://dx.doi.org/10.1145/2493190.2493231

26. A. D. Wilson and A. F. Bobick. 1999. Parametric hidden Markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21, 9 (Sep 1999), 884–900. DOI: http://dx.doi.org/10.1109/34.790429

27. Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. 2009. User-defined Gestures for Surface Computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 1083–1092. DOI: http://dx.doi.org/10.1145/1518701.1518866

28. Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. 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 (UIST '07)*. ACM, New York, NY, USA, 159–168. DOI: http://dx.doi.org/10.1145/1294211.1294238

29. Shumin Zhai and Per Ola Kristensson. 2012. The Word-gesture Keyboard: Reimagining Keyboard Interaction. *Commun. ACM* 55, 9 (Sept. 2012), 91–101. DOI:http://dx.doi.org/10.1145/2330667.2330689