Multi-Touch Gestures for Discrete and Continuous Control

Halla Olafsdottir, Caroline Appert

Multi-Touch Gestures for Discrete and Continuous Control

Halla Olafsdottir
halla@lri.fr

Caroline Appert
appert@lri.fr

Univ Paris-Sud, CNRS & Inria
F-91405 Orsay, France

ABSTRACT

Touchscreen interaction currently relies on a limited set of multi-touch gestures and a wide range of graphical widgets that are often difficult to manipulate and consume much screen real-estate. Many tasks remain tedious to perform on touchscreens: selecting text over multiple views, manipulating different degrees of freedom of a graphical object, invoking a command and setting its parameter values in a row. We propose a design space of simple multi-touch gestures that designers of user interfaces can systematically explore to propose more gestures to users. We further consider a set of 32 gestures for tablet-sized devices, by proposing an incremental recognition engine that works with current hardware technology, and empirically testing the usability of those gestures. In our experiment, individual gestures are recognized with an average accuracy of $\sim$90%, and users successfully achieve some of the transitions between gestures without the use of explicit delimiters. The goal of our contribution is to assist designers in optimizing the use of the rich multi-touch input channel for the activation of discrete and continuous controls, and enable fluid transitions between controls.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces - Graphical user interfaces.

General Terms

Human Factors, Experimentation, Performance

Keywords

Multi-Touch Gestures; Discrete Control; Continuous Control; Tablet.

1. INTRODUCTION

Multi-finger input offers a very expressive channel to interact with devices equipped with a tactile screen, by associating a given human gesture with a system control. In theory, the channel size is very large but in practice it is drastically reduced by human and system limitations. On the one hand, cognitive and motor resources limit the number of associations humans can memorize and the complexity of gestures they can perform. On the other hand, recognizing humans' intentions only from a sample of contact points is difficult.

Current touchscreen devices make an extensive use of single-finger slides and two-finger pinches for viewport navigation, but the use of other multi-touch gestures remains anecdotal. In the end, interaction still heavily relies on graphical widgets for many manipulations. For example, when selecting text over multiple views, users have to interlace finger slides (for adjusting the viewport) with manipulations of small graphical handles (for setting the selection range). Widgets on touchscreens not only reduce the size that can be dedicated to the content of interest, but also raise usability issues. For example, acquiring and precisely manipulating targets with fingers can be tedious on both small and large surfaces.

We believe that enabling users with a larger set of multi-touch gestures could both make interaction easier in scenarios like the text selection mentioned above, and make graphical presentations lighter by avoiding graphical widgets. The gestures should not only remain simple to execute, but they should also ideally be easy to input in a sequence, to provide fluid transitions between the different controls that need to be chained to achieve many tasks. For example, when selecting text, users have to be able to easily transition between navigating across views and adjusting the selection range. In other scenarios, they may want to pan a representation at different speeds to reach quickly and precisely a given paragraph in a PDF document or a geographical area in a map. They might also want to manipulate different degrees of freedom of a graphical object, as in a 3D docking task [20], or activate a command and immediately set the value of its parameters in a fluid manner [8].

In this paper we propose a design space of multi-touch gestures for interacting with devices equipped with a touchscreen. This design space is organized along dimensions that (i) make sense in terms of human anatomy, (ii) do not involve complex shapes and (iii) can be systematically explored. We introduce a recognition algorithm that can discriminate between a set of 32 gestures. These gestures were selected from our design space, and can be used to interact with a tablet-sized device. Our algorithm is incremental, i.e., it relies only on the most recent finger traces, to enable both early recognition and continuous control during gesture execution. By using only local geometrical features of the last points sampled, users are able to fluidly transition between different gestures without requiring explicit delimiters, like pausing or lifting fingers.

After a review of related work, we describe our multi-touch gesture design space. We then detail the incremental recognition engine we implemented to recognize these gestures, which works with existing hardware, without the need for additional sensors. We

Halla Olafsdottir & Caroline Appert.Multi-Touch Gestures for Discrete and Continuous Control. In AVI '14: Proceedings of the International Working Conference on Advanced Visual Interfaces, 8 pages, ACM, may 2014.

http://dx.doi.org/10.1145/2598153.2598169

report on a user study that evaluates both users' ability to perform the gestures, and the accuracy of the recognizer. In our study, participants first had to perform each gesture individually, and then execute two gestures in a row, without lifting their fingers. In this experiment, individual gestures were recognized with an average accuracy of $\sim$90% and users successfully achieved some of the transitions between two gestures in the sample we considered.

2. RELATED WORK

Multi-touch gestures not only provide advantages related to direct input [12]; they also offer a very high expressive power, as users can vary the number of fingers in contact and their individual trajectories. They are now commonly integrated in many mobile devices, usually dedicated to object manipulations such as rotate, scale and translate [20] and to the management of multiple views through multi-finger pinches and slides. Such gestures have now become so "natural" to users that they have found their way to desktop manipulations with commercial devices like multi-touch trackpads, and research prototypes like the Mouse 2.0 [26].

2.1 System and Gesture Recognition

Implementing multi-touch gestures on the system side is a challenge for interface developers. They have not only to identify the gestures of interest in a very large informal design space, but also to programmatically describe them and associate them with discrete and continuous controls. Tools such as Proton [14, 13] and Gesture Coder [17] allow developers to implement multi-touch gestures. They internally represent gestures as basic touch events organized into a state machine, where a callback can be associated with any state so as to implement continuous control. These projects focus on advancing the technical aspects of implementing any multi-touch gesture. They neither consider which sets of gestures make sense for users, nor enable transitions between different gestures within the same continuous stream of touch events.

Many recognition approaches have been proposed for single-touch input. The most famous algorithms are probably the $-family, which consists of recognition engines that are based on resampling and point-to-point comparison with gesture templates ($1 [28], Protractor [16], $N [2]). These algorithms are particularly appreciated because they are far easier to implement than statistical approaches that learn a sophisticated model from a large set of gesture samples. In this latter family, gestures can for example be represented as a sequence of strokes or angles to train Hidden Markov Models [1, 25], or as vectors of global geometric features to train covariance matrices [23].

A few approaches have focused on a recognition that is either incremental or based on local gesture features. For single-touch input, Octopocus [4, 3] adapted a posteriori recognition engines to produce dynamic guides that help users to discover and learn gestures. In a similar spirit, Kristensson and Denby [15] proposed a probabilistic approach to enable continuous recognition of users' partial input on the basis of a template-based algorithm. While these approaches enable incremental recognition, they do not yet propose fluid transitions between different gestures that are part of the same execution stream. Achieving such transitions requires considering local geometrical properties like Motion-Pointing [5] and CycloStar [18] do. These techniques successfully identify a specific oscillatory movement using only a limited number of recent gestures points. They can be used to select an item in a set [5] or to transition between pan and zoom controls [18].

2.2 Human and Gesture Execution

The number of multi-touch gestures users can find on commercial devices is rather small in comparison with the potential richness of such input. The common multi-touch gestures are either simple rectilinear slides or finger pinches. Recently, few applications allow users to rotate an object by moving at least one of the fingers in contact along a circular trajectory. The very small size of the vocabulary of multi-touch gestures is actually quite surprising, especially when compared with single-point gestures that can be used to, e.g., input any alphabetical character with systems like Graffiti or Unistroke [7]. This may be because cognitive and motor human factors limit the number and the complexity of shapes users are able to memorize and execute.

Few studies have focused on understanding which gestures are guessable [27] and easy to memorize [21]. In their "guessability" experiment, Wobbrock et al. [27] observed that user-defined gestures are easier to memorize than pre-defined gestures. However, the commands for which participants had to define gestures were inspired by mouse-based interfaces. This may have swayed participants towards defining gestures that mimic mouse use. This study also found that some gestures elicit little inter-user agreement, suggesting the need for on-screen widgets or pre-defined gestures. Nacenta et al. [21] similarly concluded that pre-defined gestures may be needed to complement user-defined ones. Pre-defined gestures are usually better recognized by the system; their mapping with controls are more consistent across applications, and they can be transferable among users in a collaborative setting. Our design space and its associated recognition engine will help designers implement such pre-defined multi-touch gestures.

Designing multi-touch gestures requires taking anatomical properties and constraints into account. In particular, the fingers cannot be considered as independent entities. Studies have shown that when one finger is moved the other ones will inadvertently move to some degree [9, 29, 30]. This *enslaving* is caused both by peripheral factors such as muscles shared between fingers, and by central factors such as overlapping cortical representations [24]. Studies have established that the thumb and index fingers are the most independent, followed by the little, middle and ring fingers [9, 22, 29, 30]. Fingers are also more enslaved to their immediate neighbor than to the other fingers (proximity effect) [30]. This is easily demonstrated by attempting to move the ring finger in isolation. The thumb is different from the other fingers as it sits in a different plane and is controlled by its own separate muscles. It has three joints, it can be extended 60°, flexed until it touches the palm, abducted 45° and adducted until it touches the index finger. This combination of features, unique for humans, equips us with opposable thumbs [19].

3. DESIGNING MULTI-TOUCH GESTURES

Our main objective is to define a design space of pre-defined multi-touch gestures that take advantage of the human hand's versatility, are limited to simple shapes, and can be recognized continuously using only basic finger traces. We want interface designers to be able to use this design space to identify and test a system of gestures in a systematic manner. To enable this systematic exploration, our classification is more structured and detailed than the only taxonomy for multi-touch gestures proposed so far [27].

3.1 Design Space

Our multitouch gesture design space is defined along four dimensions: *Contact Point (CP)*, *Constraint*, *Reference* and *Shape*.

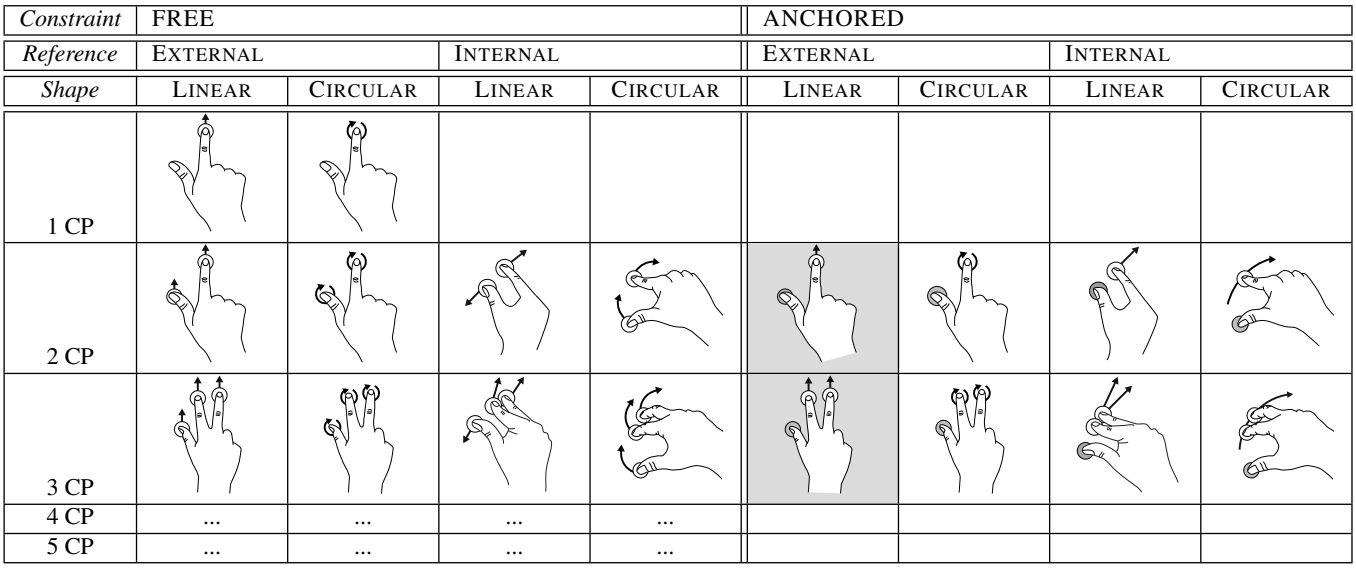

Figure 1: Design space for multi-touch gestures. Pictures show a subset of this space by illustrating only gesture classes that involve from 1 to 3 contact points (CP) and that feature at most one anchor (the thumb).

In the ***Contact Point*** dimension, the number of contact points involved in a gesture is defined. For a single-handed interaction, its values ranges from 1 to 5. To be compatible with current technology, our design space does not consider finger identification. For example, a 2-CP gesture my involve the thumb and the index finger, middle and ring finger or any other two finger combination. ***Constraint*** refers to the behavior of the contact points, that are either active or static. A gesture is *free* when all contact points are active. It is *anchored* when it has at least one static and one active CPs. Our design space only considers gestures with at least one active CP. Chord gestures, where all the CPs are static, have previously been carefully studied [6]. The ***Reference*** dimension reflects if an invariant point serves as a reference for gesture execution or not. Gestures are *internal* if there is a reference, being the centroid of contact points for free gestures or the anchor digit for anchored gestures. The ***Shape*** dimension captures the gestures' form. When creating the design space, we consciously decided to include only simple shapes: *linear* or *circular*.

The design space, illustrated in Figure 1 with a subset of gestures, is obtained by crossing the values of the above dimensions. In practice, the space is reduced by two main constraints. Both anchored and internal free gestures cannot be defined when the gesture involves only one contact point. Nevertheless, the design space still contains 18 free and 116 anchored gestures. This number is derived by identifying all possible finger combinations that yield discernible patterns of contact points. For example, a 3-finger anchored gesture can have one or two anchor points which can be either adjacent or divided, resulting in a total of six combinations for these gesture classes.

However, not all gestures are feasible. Strong enslaving of middle and ring fingers [29, 30] will, for example, make any gesture where those two do not act in unison very difficult or even impossible. The form factor of the device also affects what proportion of the design space can be used. Large devices such as tabletops allow for gestures with up-to-five contact points, or even the use of both hands. On the other hand, using more than three fingers on a tablet is often cumbersome. This number is reduced even further for small and very small devices such as smartphones and watches.

The full design space is quite large. As a first evaluation, we opt to study further a subset of gestures for a tablet-sized device, GS_{tablet}. We explicitly choose gestures that are the least challenging from the perspective of finger-coordination, have two or three contact points and a maximum of one anchor point (second and third lines of Figure 1). The gestures may involve the thumb, index, middle and/or ring fingers; but participating fingers are always adjacent to each other. The anchor point is always the thumb, as it is one of the most independent fingers [9, 29], in particular when acting in parallel to the other ones [22]. This set consists of 16 gesture classes. Within each gesture class, we consider different *directions*. Linear external gestures are along one of the four cardinal directions (NORTH, EAST, SOUTH, WEST). Linear internal gestures go either TOWARDS or AWAY from the gesture's reference point. Circular gestures can be either clockwise (CW) or counterclockwise (CCW). From this set of 40 gestures, we excluded the 8 anchored external linear gestures based on user feedback collected during informal preliminary tests (grayed out in Figure 1). These gestures are actually both uncomfortable and difficult to perform. In the end, GS_{tablet} consists of 32 gestures.

The range of motion of gestures in the design space is not uniform. Limiting factors can be anatomical, such as the length of the involved fingers and the flexibility of the hand. The size of the device may also limit the gesture amplitude. Free external linear gestures can, for example, only be as long as the size of the screen, while free internal circular gestures are limited by how much fingers and wrist can ab- and adduct, as well as by shoulder movement. External circular gestures are, on the other hand, infinite in range, as users can perform as many rotations as they wish. Gestures that have a limited range may suit well for *discrete* controls to, e.g., replace a button. Gestures that can be repeated for an arbitrary duration can be used for *continuous* controls to, e.g., replace a slider.

3.2 Recognition Engine

We now describe the recognition engine we have implemented on a Samsung Galaxy SII tablet (resolution 59 pixels per cm) to

discriminate the different types of gestures introduced above. To enable continuous control, our algorithm is incremental. It analyzes the finger traces in a recognition loop that starts as soon as one finger touches the surface. To detect static fingers, the loop runs at a frequency of 40Hz (period 25ms), independently from the frequency at which the system delivers events.

In order to avoid unstable recognition due to local noise in the different finger traces, we filter the recognition results by introducing a short lag (100 ms). A gesture is considered as reliably recognized for the first time if it has been recognized at least four times in a row by the recognition loop. It then remains the active gesture until any other gesture has been reliably recognized.

The loop treatment starts by looking at whether the gesture is *anchored or free*. A finger is anchored if its trace is bounded to a square of 50-pixel (85mm) over the last 500ms. Otherwise, it is free. A gesture is anchored as soon as one finger in contact is anchored. The algorithm then computes the values of other dimensions by using two local geometrical features: the most recent individual finger traces and the polygon formed by the finger contact points (i.e., the contact envelope). In the following, P_d refers to the polygon formed by the points pt_d of each finger trace. For a given finger, pt_d is the point located d pixels away from its current point pt_0 along its trace (i.e., distance path = d).

3.2.1 Anchored gestures

During a circular internal gesture, each free finger moves along a circle centered on the anchor location. Our algorithm considers all points of one free finger trace over the last 200 pixels (339mm), and looks at how much their distance to the anchor varies. It computes distances to the anchor for all captured points since pt_{200}. It considers a gesture as *circular internal* if the standard deviation over these distances is lower than 20 pixels. Considering only one finger trace is sufficient, as in the case of other possible anchored gestures (internal linear and external circular) all free fingers see their distance to the anchor vary.

During a linear internal gesture, all free fingers either get close to, or away from, the anchor along individual linear movements. Our algorithm considers polygons P_{50} and P_0, and the different individual finger traces over the last 50 pixels (85mm). For both polygons, it computes the vertex-centroid distance. It also computes the straightness of each individual 50-pixel trace. A trace between pt_{50} and pt_0 is considered as straight if the ratio between the length of the $pt_{50}pt_0$ segment and the distance path separating pt_{50} from pt_0 is above 0.99. A gesture is recognized as *linear internal* (i) if the vertex-centroid distance has changed by at least 15 pixels and (ii) if all free finger traces are straight. As we do not consider *linear external* gestures, the gesture is recognized as *circular external* otherwise.

3.2.2 Free gestures

During a circular internal gesture, each free finger moves along a circle centered on the centroid of the polygon's contact envelope, while the relative position of free fingers remains constant. Our algorithm considers both polygons P_{200} and P_0 and computes their angle of reference. It recognizes a *circular internal* gesture if this angle has changed by more than $\frac{\pi}{6}$. Checking this rotation criterion is enough to discriminate this type of gesture from other gestures: neither linear internal gestures nor linear/circular external gestures involve such a rotation of the contact envelope.

During a linear internal gesture, all free fingers get either close to, or away from, the centroid of the current contact envelope P_0. Our algorithm considers polygons P_{50} and P_0 and recognizes a *linear internal* gesture if the difference between the vertex-centroid distance is higher than 15 pixels (25mm) between P_{50} and P_0. It eventually discriminates between *circular external* and *linear external* gestures by looking at whether all finger traces are arcs over their last 100 pixels (170mm) or not. A trace between pt_{100} and pt_0 is considered as an arc if (i) it is not straight (according to the straightness criterion mentioned above for anchored linear internal) and (ii) it is not a corner (the ratio between the mahalanobis distance from pt_{100} to pt_0 and the distance path separating pt_{100} from pt_0 is either below 0.9 or above 1.1).

4. EXPERIMENT

We evaluate the recognizer with a user experiment. The experiment consists of two phases that are always presented to participants in the same order. In the first phase, we test individual gestures. In the second phase, we evaluate transitions between pairs of gestures. The two phases, which involve the same group of participants, are run in sequence with a 15-minute break between them.

To facilitate reporting, we identify a gesture with a short code value that consists of values along all dimensions of the design space, separated by symbol '_'. For example, F_3_INT_CIRC_CW refers to a free (F) gesture using three (3) contact points, that is internal (INT), circular (CIRC) and performed clockwise (CW).

4.1 Participants

Twelve volunteers (8 men and 4 women), 27 to 42 years old, participated in the experiment. All are right-handed and have normal or corrected-to-normal vision. Eleven participants use touchscreen devices on a regular basis.

4.2 Apparatus

We run the experiment on a Galaxy Tab II multi-touch tablet. The tablet has a 10.1 inch display with a resolution of 1280x800 pixels. It runs the Android 4.0 operating system.

4.3 Setup

Participants are first informed about the purpose and procedure of the experiment. They are asked to wash and dry their hands carefully to minimize screen friction and facilitate sliding when performing the gestures. They also complete a participant information survey. During experimental tasks, they sit on a chair, and hold the tablet in a landscape orientation with their left hand, interacting with it using their right hand. They are instructed to hold the tablet comfortably, but are otherwise free to choose their hold.

4.4 Phase 1: Individual Gestures

This first phase evaluates recognition accuracy and user performance for individual gestures of the design space. To complete the task, each gesture has to be continuously recognized for a certain distance or duration.

Discrete gestures, which are limited by either the size of the tablet or anatomical constraints, need to be stably recognized for 1.7 cm (100 pixels). *Continuous* gestures, which do not have such limitations, need to be stably recognized for 1000 ms to validate that users are able to maintain them for a substantial amount of time.

Table 1 lists the individual gestures we test in Phase 1. It consists of the 32 gestures of the GS_{tablet} set described earlier. Involved fingers may be the thumb, index, middle and ring fingers. In total, the experiment takes approximately 20 minutes to complete.

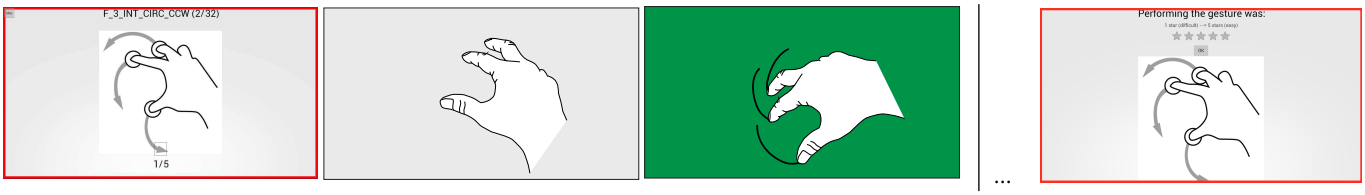

Figure 2: (Left) Individual gesture task ; (Right) Rating screen after the five repetitions

4.4.1 *Task and Procedure*

Figure 2-(Left) illustrates a task scenario. At the beginning of each trial, an illustration, similar to the ones in Figure 1, is displayed on the tablet. The involved fingers are indicated with white circles, placed under the fingertips. In some trials, one of the circles is gray. Prior to the experiment participants are informed that this is the convention used to show that the finger must act as an "anchor", i.e., it has to be kept static while the other fingers move in the direction indicated by the arrows. Participants are asked to perform the gesture, using the same set of fingers. They are allowed to observe the image for as long as they need, but as soon as their fingers touch the screen, the image is replaced by the fingers' traces. They can also perform the gestures anywhere on the screen, and adjust the orientation of their hand if needed. A change in background color provides feedback to participants about their performance: the background remains gray until a gesture class is recognized, and turns green (resp. red) if the recognized gesture class is correct (resp. incorrect).

Each gesture is repeated five times in a row, with the first two repetitions considered as practice trials. The presentation order of the 32 gestures is randomized for each participant. Thus, the design is: 12 participants x 32 gestures x 5-trial series (5 repetitions) = 1920 tasks (768 practice and 1152 measures). After a gesture has been repeated five times, participants rate, on a 5-point Likert scale: how easy/difficult it was for them to perform it, regardless of how well it was recognized (Figure 2-right).

4.4.2 *Results*

Using data collected during this experiment, we compute the mean recognition accuracy of our algorithm for the 32 classes of gestures and analyze the types of errors participants make. A trial is considered successful if the first class recognized for a sufficiently long period of time or distance is the correct one. Table 1 summarizes the recognition score and the participants' mean rating for each gesture class. Its presentation is divided into the two general gesture categories: *Continuous* gestures, which are oscillatory circular gestures (i.e., external circular gestures) users can repeat indefinitely, and *Discrete* gestures. Gestures that belong to different classes of the design space are separated by horizontal lines.

The mean recognition score across all gestures is 89% $\pm$ 31% (the size of the standard deviation stems from both inter-user and inter-gesture variability). The recognition score per participant ranges from 77% to 95% and from 65% to 100% per gesture class. Cochran's Q test reveals a significant effect of gesture class on recognition score ($Z = 2.7$, $p = 0.007$). Table 1 shows that 19 out of 32 gestures have a recognition score above 90%. We analyze below the two main categories of errors participants made, to identify what features of the gestures the recognizer confuses. We then outline guidelines to improve the recognizer.

The first category of errors concerns external circular gestures. ***For free external circular gestures, most errors occur when the recognizer confuses either LIN and CIRC gestures (12 out of 32) or recognizes the gesture incorrectly as anchored (8 out of 23).*** This type of error likely results from the between-user variability in the circle's size. Some participants draw consistently-larger circles than others. A small section of a large circle may be misinterpreted as a straight line, while a small section of a small circle can be within the tolerance limits of the bounding box used to classify a finger as an anchor. Introducing a short lag in the algorithm to consider a trace larger than 100 pixels (∼20mm) might resolve the confusion between LIN and CIRC. Regarding misclassifications as anchored gestures, we could introduce a post-treatment at the end of the recognition algorithm. This post-treatment would consist of decreasing the size of the bounding box of the anchored finger trace to make the anchored criterion less tolerant after an anchored external circular gesture has been recognized. If this revised criterion is violated, the recognition result would be changed to free external circular. ***For anchored external circular gestures, most recogni-***

Gesture	Reco Score	Rating
continuous		
F_2_EXT_CIRC_CW	88.5%	4.2
F_2_EXT_CIRC_CCW	88.5%	3.7
F_3_EXT_CIRC_CW (*)	95%	3.6
F_3_EXT_CIRC_CCW	85%	4
A_2_EXT_CIRC_CW	75%	3.4
A_2_EXT_CIRC_CCW (*)	91.5%	3.6
A_3_EXT_CIRC_CW	80%	3.6
A_3_EXT_CIRC_CCW	81.5%	2.9
discrete		
F_2_EXT_LIN_NORTH (*)	98.5%	4.5
F_2_EXT_LIN_EAST (*)	100%	5
F_2_EXT_LIN_SOUTH (*)	93.5%	5
F_2_EXT_LIN_WEST (*)	95%	4.9
F_2_INT_LIN_TOWARDS (*)	93.5%	4.2
F_2_INT_LIN_AWAY (*)	91.5%	4.3
F_2_INT_CIRC_CW	81.5%	3.1
F_2_INT_CIRC_CCW	75%	3.6
F_3_EXT_LIN_NORTH (*)	98.5%	4.5
F_3_EXT_LIN_EAST (*)	100%	4.6
F_3_EXT_LIN_SOUTH (*)	91.5%	4.7
F_3_EXT_LIN_WEST (*)	96.5%	4.6
F_3_INT_LIN_TOWARDS	73.5%	3.6
F_3_INT_LIN_AWAY	81.5%	3.6
F_3_INT_CIRC_CW	78.5%	2.4
F_3_INT_CIRC_CCW	65%	2.8
A_2_INT_LIN_TOWARDS (*)	96.5%	4.3
A_2_INT_LIN_AWAY (*)	95%	4.2
A_2_INT_CIRC_CW (*)	91.5%	4.1
A_2_INT_CIRC_CCW (*)	95%	4.1
A_3_INT_LIN_TOWARDS	88.5%	3.8
A_3_INT_LIN_AWAY (*)	96.5%	3.8
A_3_INT_CIRC_CW (*)	100%	4.2
A_3_INT_CIRC_CCW (*)	90%	4

Table 1: Recognition score and mean qualitative rating for gestures tested in Phase 1. Gestures marked with a (*) have a recognition score higher than 90%.

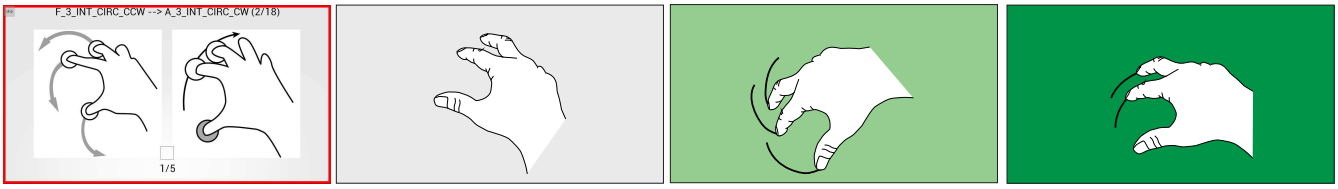

Figure 3: Gesture transition task

tion errors arise because of a confusion between INT and EXT (36 out of 38). To address this confusion, our algorithm should probably be less tolerant in the allowed variability of the distance between a moving finger and an anchored one. This could be done by increasing the minimum trace length (currently 200 pixels), or by reducing the 20-pixel variability threshold to avoid introducing an overly-long lag.

Free internal gestures are the second category of poorly recognized gestures. ***For free internal circular gestures, most recognition errors occur when the gestures are mis-recognized as anchored (88 out of 96)***. Moving the thumb to the same degree as the other fingers when performing such gestures may be difficult, causing the algorithm to consider it as an anchor. First, the thumb is shorter than the other fingers and may be limited in its range of motion along the circular trajectory (especially at the end of the movement where other fingers typically adduct). Second, it has a higher number of degrees of freedom and may be more sensitive to screen friction. This type of recognition error could be reduced by introducing a post-treatment to make the anchored criterion more severe, as already discussed above. If this less-tolerant criterion is violated once an anchored internal gesture has been recognized, the gesture is re-classified as free internal.

4.5 Phase 2: Gesture Transitions

Users often chain or interlace commands and parameter adjustments. Examples include opening a map application and getting to the current location, scrolling while adjusting the text selection range when editing a document, or turning silent mode off and adjusting the volume. The second phase of our experiment evaluates recognition accuracy and user performance when transitioning between a pair of gestures. Participants are asked to perform two gestures sequentially, without lifting their finger off the tablet. The first gesture of a pair is termed the *source gesture* and the second one the *destination gesture*. As the number of combinations of two gestures is far too large to be exhaustively tested, we picked two sample groups of representative transitions. This phase evaluates two groups of transitions, (Group-a) between gestures of the same class and (Group-b) between gestures of different classes.

The 21 gesture pairs in Group-a are listed in Table 2. Both the source and the destination gestures have the same value along the *Constraint*, *Reference* and *Shape* dimensions, but not necessarily the same number of *Contact Points*. When transitioning from source to destination gestures, participants need to change direction and may also be required to add or subtract a contact point by placing or lifting a finger.

Group-b consists of 18 gesture pairs, listed in Table 3. In this group, the source and destination gestures will take different values along a single dimension, which can be *Constraint*, *Reference* or *Shape*, while keeping the values of the other dimensions the same. This method yields 3 types of gesture transitions: (1) transitioning from anchored to free gestures or vice versa, (2) transitioning from internal to external gestures or vice versa, and (3) transitioning from linear to circular or vice versa. We end up with 3×6 gesture pairs, as anchored external linear gestures remain excluded. In this group, the number of contact points is always 3, involving the thumb, index and middle fingers. We randomly assigned a direction to each of the gestures within a pair but excluding cases where two consecutive discrete gestures have identical directions (such as A_2_INT_LIN_TOWARDS and F_2_INT_LIN_TOWARDS) as this type of transition is impossible to perform.

4.5.1 Task and Procedure

At the beginning of each trial, participants familiarize themselves with both gestures of a pair individually, by completing two repetitions of the first experimental task (Figure 2-(Left)) for both source and destination gestures. They then perform five repetitions of the transition task, illustrated in Figure 3. Images of both gestures are shown simultaneously on the screen to inform participants that they should perform them in a sequence and without lifting all their fingers off the tablet. The background color gives participants feedback about the recognition. The screen turns light green when the source gesture has been recognized. Participants can then transition to the destination gesture. The background turns dark green when the destination gesture is recognized. If an incorrect gesture class is recognized while performing the source gesture, the color of the background turns red and participants need to lift their fingers and restart the task.

Following a correct recognition of the source gesture, the experimental program logs incorrect recognition results, but does not alert participants. The task ends as soon as the destination gesture has been properly recognized. If participants lift their fingers off the tablet, they have to restart the trial regardless of their progress. Prior to the experiment, we did not know whether participants would be able to transition between gestures without making any error. To save participants from too much frustration and fatigue, we decided to internally log the errors that occurred during the transition instead of having participants restart the trial. As incremental recognition during noisy transition phases is very challenging, the goal of this experiment is more to investigate if our approach is usable in at least some cases and propose guidelines on how to improve those that are less usable.

Participants first practice two repetitions of both individual gestures and then perform five repetitions of the gesture pair. As in Phase 1, the first two repetitions of a pair are considered practice trials. The order of gesture pairs within a group are randomized, but the presentation order of Group-a and Group-b is counterbalanced across participants. Each group takes approximately 15 minutes and participants rest for 5-10 minutes between the two groups. The design of this experiment is: 12 participants $\times$ 39 gestures $\times$ 9-trial series (2 $source$ + 2 $destination$ + 5 $source \rightarrow destination$) = 4212 tasks. $12 \times 39 \times 3 = 1404$ are actually measured. As in Phase 1, we collect participants' perception of task difficulty using a 5-point Likert scale presented after each 9-trial series.

continuous → continuous

Source	Destination	Reco Score	Rating
F_2_EXT_CIRC_CW	F_2_EXT_CIRC_CCW	100%	4.1
A_2_EXT_CIRC_CCW	A_2_EXT_CIRC_CW	100%	4.3
F_2_EXT_CIRC_CCW	F_3_EXT_CIRC_CW	86.1%	4.1
A_2_EXT_CIRC_CW	A_3_EXT_CIRC_CCW	91.6%	4.2
F_3_EXT_CIRC_CCW	F_2_EXT_CIRC_CW	88.9%	4
A_3_EXT_CIRC_CW	A_2_EXT_CIRC_CCW	86.1%	4.1

discrete → discrete

Source	Destination	Reco Score	Rating
A_2_INT_LIN_AWAY	A_2_INT_LIN_TOWARDS	80.5%	3.9
F_2_INT_LIN_TOWARDS	F_2_INT_LIN_AWAY	94.5%	4.3
F_2_INT_CIRC_CCW	F_2_INT_CIRC_CW	83.5%	3.1
A_2_INT_CIRC_CW	A_2_INT_CIRC_CCW	80.5%	4.4
F_2_EXT_LIN_NORTH	F_2_EXT_LIN_EAST	100%	4.2
F_2_INT_LIN_AWAY	F_3_INT_LIN_TOWARDS	83.5%	3.6
A_2_INT_LIN_TOWARDS	A_3_INT_LIN_AWAY	88.8%	3.7
F_2_INT_CIRC_CW	F_3_INT_CIRC_CCW	41.5%	3.3
F_2_EXT_LIN_WEST	F_3_EXT_LIN_SOUTH	97.2%	4
A_2_INT_CIRC_CCW	A_3_INT_CIRC_CW	80.5%	3.8
A_3_INT_CIRC_CCW	A_2_INT_CIRC_CW	72.2%	4.2
F_3_INT_CIRC_CW	F_2_INT_CIRC_CCW	58.5%	3
F_3_INT_LIN_AWAY	F_2_INT_LIN_TOWARDS	88.8%	3.9
A_3_INT_LIN_TOWARDS	A_2_INT_LIN_AWAY	75%	4
F_3_EXT_LIN_SOUTH	F_2_EXT_LIN_NORTH	64%	4

Table 2: Gesture transitions tested in Phase 2 - Group-a

continuous → continuous

Source	Destination	Reco Score	Rating
A_3_EXT_CIRC_CCW	F_3_EXT_CIRC_CCW	97.2%	3.5
F_3_EXT_CIRC_CW	A_3_EXT_CIRC_CW	97.2%	3.4

discrete → discrete

Source	Destination	Reco Score	Rating
F_3_INT_CIRC_CW	F_3_INT_LIN_TOWARDS	38.9%	2.6
A_3_INT_CIRC_CCW	A_3_INT_LIN_AWAY	36.1%	3
F_3_INT_LIN_AWAY	A_3_INT_LIN_TOWARDS	58.3%	2.7
A_3_INT_LIN_AWAY	F_3_INT_LIN_TOWARDS	83.3%	3.6
F_3_INT_LIN_AWAY	F_3_EXT_LIN_EAST	94.5%	3.1
A_3_INT_LIN_TOWARDS	A_3_INT_CIRC_CW	58.3%	2.1
A_3_INT_CIRC_CCW	F_3_INT_CIRC_CW	25%	2.8
F_3_INT_CIRC_CCW	A_3_INT_CIRC_CW	72.2%	2.6
F_3_EXT_LIN_EAST	F_3_INT_LIN_AWAY	83.3%	3.7
F_3_INT_LIN_AWAY	F_3_INT_CIRC_CCW	97.2%	3.1

discrete → continuous

Source	Destination	Reco Score	Rating
F_3_EXT_LIN_SOUTH	F_3_EXT_CIRC_CCW	86.1%	3.4
A_3_INT_CIRC_CCW	A_3_EXT_CIRC_CW	77.7%	3.6
F_3_INT_CIRC_CW	F_3_EXT_CIRC_CW	72.2%	2.5

continuous → discrete

Source	Destination	Reco Score	Rating
A_3_EXT_CIRC_CW	A_3_INT_CIRC_CW	88.8%	2.8
F_3_EXT_CIRC_CCW	F_3_INT_CIRC_CW	66.6%	2.7
F_3_EXT_CIRC_CW	F_3_EXT_LIN_WEST	91.6%	4

Table 3: Gesture transitions tested in Phase 2 - Group-b

4.5.2 Results

As for Phase 1, we group results according to the *discrete* vs. *continuous* control properties of the gestures (Table 2). In Experiment 2, we end up with four categories. *discrete* × *discrete* pairs of gestures can be used to invoke two commands sequentially (e.g., "open mail app" followed by "compose message"). *discrete* × *continuous* gesture pairs can be used to invoke a command and set its parameter (e.g. "brightness" followed by value setting). *continuous* × *discrete* pairs of gestures can be used to adjust a selection and invoke a contextual command (e.g. selecting text and "copying" it). *continuous* × *continuous* gesture pairs can be used to chain two continuous controls (e.g. scale a map and adjust its orientation).

All trials collected have 100% recognition scores for the source gesture in the pair, since participants had to redo the trial if a recognition error occurred (although we did log the number of such errors). If the destination gesture is *discrete*, the trial is correctly recognized as soon as the destination gesture class is correct. If the destination gesture is *continuous* the trial is correct when the destination gesture class has been recognized for at least 500ms.

Participants transition between two *continuous* gestures with a recognition accuracy of 93% ± 6%. This is a surprisingly good result, as recognition scores for individual *continuous* gestures was 83% ± 6%. Participants appear to have benefited from learning in the first phase, as recognition accuracy for the source gesture in this second phase has increased to 89.6% ± 12% when this gesture is *continuous* (i.e., external linear).

On the opposite, transitioning from a *continuous* to a *discrete* gesture is more problematic for the three transition types of Phase 2. As free internal gestures are limited by anatomical constraints, they can be uncomfortable when started in an inappropriate posture. This means that participants may have repositioned their hand during the transition in order to be able to perform the destination gesture. As they are instructed to keep their fingers in contact with the surface, repositioning probably introduces noisy traces that get misinterpreted by the incremental recognition process.

Transitioning along the *Form* dimension during free external gesturing is probably more promising, as the recognition score for F_3_EXT_CIRC_CW to F_3_EXT_LIN_WEST suggests (91.6%, last line of Table 3). We plan to test a larger sample of transitions to further validate this interpretation.

Transitioning from a *discrete* gesture leads to very contrasting results. Participants frequently end *Discrete* gestures at the end of their range of motion, and often in a position that is uncomfortable to start another gesture. This is especially true for Group-b, where the gestures of a pair are of different classes. Internal linear gestures that tend to spread out users' fingers (AWAY direction), however, exhibit much better recognition scores, ranging from 80.5 to 97.2%, if we exclude transitioning from free to anchored gestures (F_3_INT_LIN_AWAY → A_3_INT_LIN_TOWARDS). When transitioning between *discrete* gestures of the same class (Group-a), conditions in which the same number of fingers is kept in contact are better recognized (91.3% ± 9.4%) than those where fingers are added (81.3% ± 18.3%) or subtracted (76.2% ± 12.3%). Changing the number of contact points in internal circular gestures appears to be particularly difficult. When this type of transition is excluded, recognition scores are tilted even further in favor of adding a finger (88% ± 6%) in comparison with subtracting (79.2% ± 10.3%).

5. CONCLUSION

We propose a design space of multi-touch gestures that are inspired by the versatility of the human hand and can be continuously recognized by using geometrical features and the number of contact points. This design space includes variations on existing multi-touch gestures, such as pinches or slides, and proposes new ones. By combining the results of our experiment with known constraints about finger coordination, we can say with confidence that the design space contains at least 88 gestures that are feasible from both a recognition and an execution perspective. Based on human anatomical properties and device form factor, we identify two main categories of gestures that are suitable for either *discrete* or *continuous* control. The structure of this design space allows user interface designers to systematically identify and test gesture sets. We explore 32 gestures that can be used on a tablet-sized device by de-

signing a recognition engine for this gesture set and running a user study to evaluate both recognition accuracy and user performance. Our algorithm only considers local geometrical properties of the most recent gesture traces to enable continuous control as soon as possible after the user starts gesturing, and enable fluid transitions between different gestures. In our experiment, the overall recognition score for the 32 gestures is $\sim 90\%$. When participants transition between two gestures, the results are more contrasted. Out of 39 transitions, 11 of them are recognized with an accuracy greater than 90%, but 12 others with an accuracy lower than 75%. We argue that these scores represent a "worst case scenario", as the recognizer chose one gesture among the 32. In the context of a real application, only a subset would likely be used.

Our future work will first focus on improving the recognition of individual gestures based on collected data. Our experiment does not indicate that any particular gesture from our set should be excluded, as none was rated below 3 when participants were asked to rate how difficult (1) or easy (5) it was for them to perform each respective gesture. The experiment we report here is only a preliminary step as we consider a subset of 39 transitions that we chose without considering anatomical constraints related to starting a gesture directly after the end of another. Previous studies [10, 11] have shown that multi-touch gestures can be uncomfortable when started in some specific postures. We will use this literature to guide us in identifying the most user-friendly transitions. Simultaneously, we will work on technical aspects to propose the best mapping between gestures and user interface controls.

6. REFERENCES

[1] D. Anderson, C. Bailey, and M. Skubic. Hidden markov model symbol recognition for sketch-based interfaces. In *AAAI Fall Symposium*, 15–21, 2004.

[2] L. Anthony and J. O. Wobbrock. A lightweight multistroke recognizer for user interface prototypes. *GI '10*, 245–252. CIPS, 2010.

[3] C. Appert and O. Bau. Scale detection for a priori gesture recognition. *CHI '10*, 879–882. ACM, 2010.

[4] O. Bau and W. E. Mackay. Octopocus: A dynamic guide for learning gesture-based command sets. *UIST '08*, 37–46. ACM, 2008.

[5] J.-D. Fekete, N. Elmqvist, and Y. Guiard. Motion-pointing: Target selection using elliptical motions. *CHI '09*, 289–298. ACM, 2009.

[6] E. Ghomi, S. Huot, O. Bau, M. Beaudouin-Lafon, and W. E. Mackay. Arpège: learning multitouch chord gestures vocabularies. *ITS '13*, 209–218. ACM, 2013.

[7] D. Goldberg and C. Richardson. Touch-typing with a stylus. *CHI '93*, 80–87. ACM, 1993.

[8] F. Guimbretière and T. Winograd. Flowmenu: Combining command, text, and data entry. *UIST '00*, 213–216. ACM, 2000.

[9] C. Hager-Ross and M. Schieber. Quantifying the independence of human finger movements: comparisons of digits, hands, and movement frequencies. *Journal of Neuroscience*, 20(22):8542, 2000.

[10] E. Hoggan, M. Nacenta, P. O. Kristensson, J. Williamson, A. Oulasvirta, and A. Lehtiö. Multi-touch pinch gestures: Performance and ergonomics. *ITS '13*, 219–222. ACM, 2013.

[11] E. Hoggan, J. Williamson, A. Oulasvirta, M. Nacenta, P. O. Kristensson, and A. Lehtiö. Multi-touch rotation gestures: Performance and ergonomics. *CHI '13*, 3047–3050. ACM, 2013.

[12] J. Karat, J. E. McDonald, and M. Anderson. A comparison of menu selection techniques: touch panel, mouse and keyboard. *International Journal of Man-Machine Studies*, 25(1):73 – 88, 1986.

[13] K. Kin, B. Hartmann, T. DeRose, and M. Agrawala. Proton++: A customizable declarative multitouch framework. *UIST '12*, 477–486. ACM, 2012.

[14] K. Kin, B. Hartmann, T. DeRose, and M. Agrawala. Proton: Multitouch gestures as regular expressions. *CHI '12*, 2885–2894. ACM, 2012.

[15] P. O. Kristensson and L. C. Denby. Continuous recognition and visualization of pen strokes and touch-screen gestures. *SBIM '11*, 95–102. ACM, 2011.

[16] Y. Li. Protractor: A fast and accurate gesture recognizer. *CHI '10*, 2169–2172. ACM, 2010.

[17] H. Lü and Y. Li. Gesture coder: A tool for programming multi-touch gestures by demonstration. *CHI '12*, 2875–2884. ACM, 2012.

[18] S. Malacria, E. Lecolinet, and Y. Guiard. Clutch-free panning and integrated pan-zoom control on touch-sensitive surfaces: The cyclostar approach. *CHI '10*, 2615–2624. ACM, 2010.

[19] K. L. Moore. *Clinically oriented anatomy*. Williams and Wilkins, 1992.

[20] M. A. Nacenta, P. Baudisch, H. Benko, and A. Wilson. Separability of spatial manipulations in multi-touch interfaces. *GI '09*, 175–182. CIPS, 2009.

[21] M. A. Nacenta, Y. Kamber, Y. Qiang, and P. O. Kristensson. Memorability of pre-designed and user-defined gesture sets. *CHI '13*, 1099–1108. ACM, 2013.

[22] H. Olafsdottir, V. M. Zatsiorsky, and M. L. Latash. Is the thumb a fifth finger? a study of digit interaction during force production tasks. *Experimental brain research*, 160(2):203–213, 2005.

[23] D. Rubine. Specifying gestures by example. *SIGGRAPH '91*, 329–337. ACM, 1991.

[24] M. H. Schieber. Constraints on somatotopic organization in the primary motor cortex. *Journal of neurophysiology*, 86(5):2125–2143, 2001.

[25] T. M. Sezgin and R. Davis. Hmm-based efficient sketch recognition. *IUI '05*, 281–283. ACM, 2005.

[26] N. Villar, S. Izadi, D. Rosenfeld, H. Benko, J. Helmes, J. Westhues, S. Hodges, E. Ofek, A. Butler, X. Cao, and B. Chen. Mouse 2.0: multi-touch meets the mouse. In *UIST '09*, 33–42. ACM, 2009.

[27] J. O. Wobbrock, M. R. Morris, and A. D. Wilson. User-defined gestures for surface computing. *CHI '09*, 1083–1092. ACM, 2009.

[28] J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. *UIST '07*, 159–168. ACM, 2007.

[29] W. S. Yu, H. van Duinen, and S. C. Gandevia. Limits to the control of the human thumb and fingers in flexion and extension. *Journal of neurophysiology*, 103(1):278–289, 2010.

[30] V. M. Zatsiorsky, Z.-M. Li, and M. L. Latash. Enslaving effects in multi-finger force production. *Experimental Brain Research*, 131(2):187–195, 2000.