AN EXPLORATION OF MULTI-TOUCH INTERACTION TECHNIQUES

A Dissertation

by

SASHIKANTH DAMARAJU SRIRANGA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Tracy Anne Hammond
Committee Members,	James Caverlee
	Jinsil Hwaryoung Seo
	Weiling He
Head of Department,	Duncan M. Walker

August 2013

Major Subject: Computer Science

ABSTRACT

Research in multi-touch interaction has typically been focused on direct spatial manipulation; techniques have been created to result in the most intuitive mapping between the movement of the hand and the resultant change in the virtual object. As we attempt to design for more complex operations, the effectiveness of spatial manipulation as a metaphor becomes weak. We introduce two new platforms for multi-touch computing: a gesture recognition system, and a new interaction technique.

I present Multi-Tap Sliders, a new interaction technique for operation in what we call *non-spatial parametric spaces*. Such spaces do not have an obvious literal spatial representation, (Eg.: exposure, brightness, contrast and saturation for image editing). The multi-tap sliders encourage the user to keep her visual focus on the target, instead of requiring her to look back at the interface. My research emphasizes ergonomics, clear visual design, and fluid transition between modes of operation. Through a series of iterations, I develop a new technique for quickly selecting and adjusting multiple numerical parameters. Evaluations of multi-tap sliders show improvements over traditional sliders.

To facilitate further research on multi-touch gestural interaction, I developed mGestr: a training and recognition system using hidden Markov models for designing a multi-touch gesture set. Our evaluation shows successful recognition rates of upto 95%. The recognition framework is packaged into a service for easy integration with existing applications.

DEDICATION

To Amita, who helped push me out the door.

ACKNOWLEDGEMENTS

I would not be able to complete this degree without the help of my adviser Dr. Tracy Hammond, to whom I eternally grateful. I'd like to thank my mentor Dr. Andruid Kerne for introducing me to the field of human centered systems, and giving me immense encouragement to pursue my ideas, however hair-brained they initially sound.

My friends have been the singular source of motivation along the roller coaster ride of my graduate career. I'd like to thank Andrew Webb, Zachary Dougas Toups, William Hamilton and Yin Qu at the Interface Ecology Lab for their collaboration and support. Jon Moeller has been an amazing collaborator and a maker, without whose help I would not have been able to fabricate my hardware. I owe Abhinav Mathur and Nabeel Shahzad a round of drinks (and more) for pushing me through strenuous days and nights of hard work to meet my deadlines. My colleagues at the Sketch Recognition Lab Stephanie Valentine, Paul Taele, Manoj Prasad and Davind Turner have been amazing company with whom I've had the privilege of spending the last leg of my time here.

Finally, I'd like the thank my parents, for whom my graduation has always been a year and a half away. Their support has meant the world to me.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

1. INTRODUCTION

This research promotes the use of multi-touch beyond spatial manipulation. We describe a motivation from an analysis on commonly performed operation. A new domain of interaction for manipulating *non-spatial parameter spaces* is identified. We present a novel multi-touch interaction technique that merges the operations of selection and adjustment of numerical parameters.

Historically, *command line interfaces* (CLIs) constituted the first modality of human computer interaction. The user had to memorize commands acceptable to the computer. Commands were designed with ease of repetition by expert users as a priority. Shortened textual notations of described actions was the norm: *ls* for list, and *rm* for removing files. As computing resources improved, displays allowed the visual representation of complex content on the screen. Graphical user interfaces (GUIs) adopted the paradigm of Windows, Icons, Menu and Pointer (WIMP) for presenting and interacting with content. The *menu* emphasized recognition of operations from a visible set, over the recall of how to invoke a command from a vast number of options, as was the case in CLIs.

In 1983, Shneiderman [104] introduced the term *direct manipulation* to mean interaction with elements of the interface using a pointer. The pointer, controlled by the physical movement of a mouse by the human, operated on visual targets such as buttons or menu items. With multi-touch, the human can now interact with a computer by directly placing her hands on the visual. The pointer is no longer a required part of the interaction between the human and the computer.

This research investigates the design space of multi-touch interaction beyond direct spatial manipulation. The first approach is that of gesture recognition. I demon-

strate techniques to recognize arbitrary multi-touch gestures. Inspired by the use of multiple fingers of a single hand for quick command selection in keyboard shortcuts, I demonstrate how numerical parameters can be quickly selected and adjusted.

1.1 Contributions

The summarized contributions of this research are:

1. Motivating the design space of parameter adjustment for multi-touch interaction.

2. Demonstrating a new interaction paradigm for merging selection and adjustment of a small sets of parameters with touch interfaces.

3. A design approach emphasizing comfort of the human participant.

4. Evaluation presenting drawbacks and benefits of using touch for relative parameter adjustment.

5. Demonstrating two domains (photo post-processing and data analysis) that qualitatively benefit from use of the MultiTap Slider.

6. Identifying techniques for training and recognizing arbitrary multi-touch gestures.

1.2 Problem statement

Engaging in visual design tasks often requires users to adjust multiple parameters for an associated action. These are typically presented to the user as a set of sliders. Parameter adjustment requires the user to shift visual focus away from the target to acquire the *thumb* of the slider (the visual portion of the slider indicating the current value). Design requires iterations of the following sequence of subtasks: parameter

selection, adjustment, and observing the effect on the target. The duration of each iteration is further extended when the user needs to switch between different parameters. This leads to to-and-fro visual saccading between an interactive component and the target object, even if the interface is located close to the target.

Jacob et al. [54] present a dual classification of how humans perceive combinations of attributes: *integral* or *separable*. This describes aspects of a user's mental model, a cognitive term for describing how a user believes the system works. Attributes such as the x and y points of a location are perceived as an integral whole. Manipulation of location includes both the parameters. However, the tuple of attributes size and saturation are not perceived together, but are separable. I attempt to bring the user an experience that allows the integration of the multiple parameters of an operation as close at hand as possible.

These distractions of repeated saccading during tasks with high cognitive loads, such as visual design, lead to break downs in the user experience. This context is similar to that of visual analytics, wherein the visual is not a design, but a visualization of large sets of data. The problems describe above apply to interactions that require problem-solving and analysis of data. Our research attempts to develop new forms of interaction for manipulation in such multi-dimensional abstract parameter spaces, that do not have an obvious 2D or 3D spatial representation.

I ask the following questions to guide this research: What movements of the hand would map to movements of a particular parameter? How could we specify which parameter must be altered, and by how much? Can each of these movements of the hand be efficiently performed multiple times without little effort?

Operations in creative fields, such as post-processing of a photograph taken by a digital camera, are delicate balancing acts. The photographers take great care in making minute adjustments of a large number of parameters available to them. For

example, the tone curve consists of four parameters that can be adjusted: highlights, lights, darks and shadows. The values of each of such parameters are used to perform a complex set of mathematical calculations to alter the pixel color values.

Even though photographers may not fully comprehend the details of the implementation, they understand the resultant visual change. As their expertise with the software grows, their knowledge of the relationship between the several groups of parameters increases. However, this does not imply that given a photograph, an expert photographer can uniquely identify the desired parameter values without interaction.

There is no correct answer to these operations. The user interface exposes a set of knobs that feed into complicated algorithms that ultimately change pixels. Details are abstracted away from the user to simplify the experience, while delivering as much power from the algorithms as possible. Providing more knobs would overwhelm the user, where as providing fewer might take away control. The software attempts to find a balance between the two.

The operation demands play. There is a constant exploration of what the photograph could look like with different values. The process of adjustment is an open-ended task that benefits from such exploration. Each parameter is constrained to a fixed range. However, even with just four parameters, under the menu Tone Curve, the possible combinations are in the millions. The operation is purely subjective. The experience and design sense of the photographer can aid in identifying the satisfactory combination of the parameter values. This selection is the primary purpose of the interaction. Devoting more visual and cognitive faculties of the brain, will aid the user in making a qualitatively improved product.

2. RELATED WORK

Our research draws from several related fields in touch interaction. Recent interest in the field has increased considerably due ubiquity of input devices, and the low cost for entry to do-it-yourself multi-touch screens. To frame our research, we consider below the breadth of research in touch interaction.

The term *direct manipulation* was introduced [104] to mean interaction with elements of the interface using a cursor. The cursor, controlled by the physical movement of a mouse by the human, operated on visual targets. Changes to the target are continuously visible. Direct manipulation brings to the forefront the new ability to rapidly perform incremental, reversible operations while viewing their effect on the object.

Direct manipulation was a great paradigm for human computer interaction. At the time Shneiderman coined the term, the mouse and cursor seemed more direct than the CLIs. However, touch allows for a more *direct* manipulation than the mouse and cursor. The target of the manipulation with a cursor is physically removed from the human hand moving the mouse.

The *more* direct manipulation made possible with touch interaction raises several questions regarding existing designs of the menu and the pointer. What is the optimal on-screen placement of the menu? What should its visual representation be? How can the dexterity of the human hand be leveraged? Not only does this mean that the user can interact with several on-screen objects at the same time, but also that several users can now interact with the same screen, opening up the research field of single display groupware [106]. This research focuses on command selection using multi-touch interaction. We limit the survey to techniques applicable to interaction

requiring finger and hand contact with a surface.

We consider attributes of research to classify prior work. One attribute is the use of either direct manipulation techniques or explicit command selection. The former deals with mappings of hand movements to implicit selection and adjustment of spatial parameters. The latter category explicitly presents commands available for selection.

We present a discussion on the techniques and a comparison of their evaluation methods.

2.1 Sensing technologies

2.1.1 Input only multi-touch sensing

Westerman [112] developed a detailed finger tracking *capacitive sensing* input device. The device used a combination of heuristic algorithms to detect fingers and the palm. A keyboard layout printed on the device allowed it to function as a keyboard, mouse, and gestural input device. It distinguishes individual fingers with fine granularity.

Westerman developed a technique using a selection of fingers (*chords*) combined with a movement to distinguish the intent of interaction. An iconic representation of the chords and their movements was illustrated. Mappings between the combinations of movement-chords to actions in a computer are presented as a form of gestural interaction. However, the only evaluation of the device was self-reported by the author himself over a period of 6 weeks.

The device was sold by the company FingerWorks. Production was ceased when acquired by Apple Inc. This technology was the 'revolutionary' new innovation for the Apple iPhone and iPod, when used as a transparent overlay on an LCD display. This innovation has made popular the idea of computing with only a touch-screen,

sans keyboard. Today's Apple MacBooks and the Apple *Magic trackpad* use a similar capacitive sensing technology to sense multiple fingers on the trackpad. Unlike the FingerWorks device, only rudimentary gestures such as multi-finger swiping and pinch-to-zoom are implemented on the trackpad.

2.1.2 Integrated display input devices

The first commercially available multi-touch sensing systems such as DiamondTouch [26] table and SmartSkin [98] initially served as a platform for multi-user interaction, rather than specifically multi-touch interaction. The front-projected hardware supported the detection of the coarse shape of the contact with the screen, allowing disambiguation of palm, open and closed hand, and single finger interaction.

The DiamondTouch table (see Figure 2.1), a product of the Mitsubishi Engineering Research Lab (MERL) served as the foundation for a decade of touch-based table top collaboration. The table enabled associating a touch point on the screen to a particular user. A specialized stylus could also be used for precise input on the surface. Although we constrain this survey to single-user interaction, we consider important work performed on the DiamondTouch table, that has an impact on multi-touch command selection.

Han developed a cheap rear-projected vision based system to detect touch using frustrated total internal reflection (FTIR) of infra-red light [38]. The infra-red region of the light spectrum is used for touch sensing, while the visual is left for display. The inexpensive, easy to fabricate FTIR systems enabled a large influx of multi-touch interaction research. This technique of using infra-red light to capture touch input has been extended by amateur hardware enthusiasts and researchers.

Variations on the vision based sensing techniques include placing the camera above the plane of interaction (inverted FTIR [28]), and illuminating the interactive

Figure 2.1: A view of the capacitive sensing DiamondTouch table [26], with a top-projected display being used collaboratively.

surface from below (diffused illumination) and using a lightplane [108]. The vision based techniques suffer from the requirement of having the infra-red lighting, camera and, projection components occupying a significant amount of space, either in front of or behind the screen. However, they also make it possible to sense more features of the touch contact. Postures can be detected [20] over the tabletop display.

Moeller and Kerne present ZeroTouch[80], in which the infra-red lighting and sensing components are compressed into a frame that can be mounted onto any display. By reducing costs, and improving the form factor of the sensors, technologies like the ZeroTouch will hasten the pace of adoption of large screen multi-touch in mainstream computing.

The iPhone from Apple [47] was the first commercial display device to support multi-touch input via capacitive sensing. The 3.7" screen could physically support

only a couple of fingers at a time. The introduction of the iPad [47] with a 9.7" screen, and the light-weight device brought whole handed multi-touch input into the mainstream market. Microsoft develops a larger coffee-table form factor touch screen, the PixelSense [49]. The table uses diffused illumination and multiple cameras to detect the shape of the contact. The Surface also features additional functionality such as object and fiducial (geometric barcodes) detection.

2.2 Support for legacy interfaces

This section discusses important research that tackles issues in the real-world adoption of multi-touch technologies. Specifically, the *fat finger* problem and the lack of a *tracking state* of the hand above the screen, analogous to the *mouse over* state. Not only does the human finger visually occlude the pixels that it is attempting to target, but it also lacks the single-pixel precision of the mouse cursor.

2.2.1 Fat fingers

Early research [94] raises the issue of moving from a cursor capable of selecting a single pixel to a finger that occludes several pixels at once, and the input approximates the contact to one of those pixels. Sears and Shneiderman present *take-off* [103] as a promising technique, wherein a cursor is presented to the user with an fixed offset to their finger. Zoom-pointing [10] is developed using graphical scaling of the target area, increasing the motor control space available to the user. Albisson and Zhai [3] develop multiple techniques to improve the precision of the finger on a screen, using 2D levers, an interface that allows precise manipulation of the cursor position. Rubbing and tapping [92] on the screen has also been used to improve the precision of interaction touch-screen displays.

Traditional user interface components such as buttons and checkboxes become difficult to operate on touch screens when packed densely. Attempting to change the

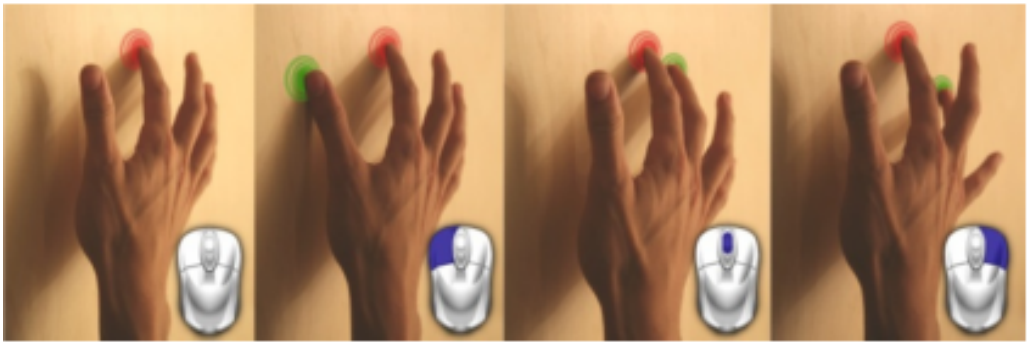

Figure 2.2: The SDMouse technique developed by Matejka et al. [78].

state of one checkbox with a finger could accidentally trigger an adjacent checkbox. Moscovich [84] presents sliding widgets as a novel technique to overcome this problem. The finger is considered as an area input, instead of a singular point on screen. A directional swipe, instead of a tap, is used to click on a button, or alter the state of the checkbox. A list of such items require either left-to-right or right-to-left swipes, alternately. This prevents the inaccurately placed finger from accidentally activating nearby interface components. Twelve participants are recruited for laboratory experiment comparing traditional push buttons with sliding buttons. Their results show that accidental activations are reduced by the use of sliding widgets.

2.2.2 Mouse emulation

In the interest of supporting general purpose use of computers using the touch modality, researchers developed techniques to emulate the mouse, and its buttons. The fluid DTMouse [29] describes a mapping using multiple fingers to emulate mouse move while avoiding occlusion (mouse cursor is moved at the mid point of two fingers) and mouse drag (a third finger toggles drag mode). Benko et al. [11] introduce a menu for a bimanual operation that facilitates magnification, controlling speed of cursor movement (indirect manipulation of the cursor) and snapping the cursor to

items.

Matejka et al. [78] develop a new design (SDMouse) for multi-finger chorded mouse emulation. Through evaluation they show an improvement over the DTMouse techniques. Figure 2.2 shows the final design of emulation of the mouse functions : move, drag, scroll and right-click. They present an exhaustive experimental study with the following independent variables: four different techniques, two tasks (click and drag), two mouse buttons (left or right), target distance and size. From their results, we see the mouse outperforms all touch techniques. However, amongst the touch techniques compared, theirs was the fastest.

2.2.3 Touch feedback

Unlike the mouse, many touch based systems cannot sense above-the-surface interaction, i.e, a mouse-over (hover) state. There exist specialized hardware systems designed to overcome this deficiency [44, 108]. However, in our research, we focus on developing techniques that do not require the additional hardware. This implies that the only states that can be detected are click and drag. The missing state prevents the system from showing the user which element is currently being acted upon. Additionally, providing visual feedback during for normal and unexpected behaviour is the application's responsibility. The operating system does not notify the user if the hardware is faulty, or which interface component is going to process the input.

Wigdor et al. introduce Ripples [115], a visual feedback mechanism that shows an animation in response to touch interaction (see Figure 2.3). The fluid, dynamic feedback shown around the finger, represents whether or not an on-screen element has *captured* the touch. Input from a captured touch is only sent to a single interface component, until the finger is lifted. This prevents accidental activation elsewhere

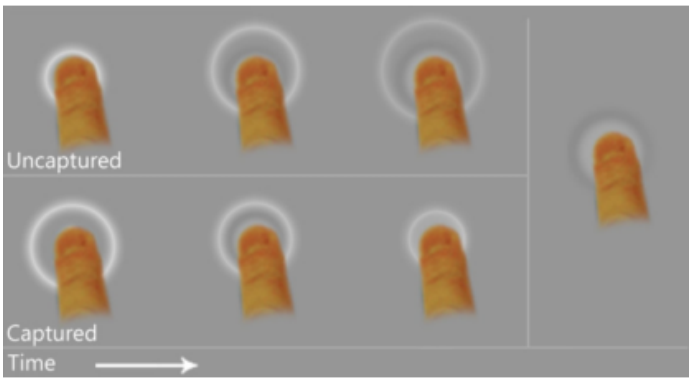

Figure 2.3: Sequence showing *ripples* [115], an animation of a circle around a finger contact, based on whether it is over an active interface element (captured) or not (uncaptured).

on the screen. A *tether* is shown from the contact point to the interface component that has captured the particular input.

2.3 Direct spatial manipulation

Researchers coined the term *natural user interfaces* (NUIs) [1, 116] to describe new techniques that allow for fluid manipulation of virtual items. The goal of such techniques is to make the resulting interaction seemingly obvious to the user, that is, the techniques should *do the right thing* for a given context. Over the past decade, new sensing technologies have made this field rich and quick-paced. Although initially developed for the touch modality, the term NUI has also been applied for full-body and voice interaction with devices like Microsoft XBox Kinect [50].

Following the paradigm of direct manipulation, we consider research in multi-touch interaction focused on altering the spatial attributes of elements. Traditionally, the term direct manipulation was used for graphical buttons that could be clicked to perform a command. With touch, manipulation is more direct.

A user can drag photos on a canvas by dragging a finger. Placing a second finger

on the canvas enables performance of three operations with an single movement of the hand: rotation, translation and scaling. This simple demonstration brings the natural hand movements that we use for physical objects to interact with the digital world. These manipulations of spatial parameters received the majority of attention in initial research and development on multi-touch.

We use the term *direct spatial manipulation* to specifically identify techniques that alter the spatial appearance of a visual. Direct spatial manipulation makes the selection of commands of rotation, translation and scaling *implicit*. These commands are selected through the use of *mappings*, a relationship between the movement of the human hand across the surface to a command or operation executed on an item on screen. The result of these manipulations is the adjustment of visual parameters of items, namely their X, Y location, their angle of orientation, and their scale.

The following section presents the prior work in two parts: (1) techniques for translate-rotate-scale for two and three dimensional objects, and (2) using physics to facilitate more natural mappings for operations on multiple elements.

2.3.1 Translate-rotate-scale

An early motivation for touch interaction was co-located collaboration. Wu and Balakrishnan [123] used the DiamondTouch table to explore techniques using two fingers for rotation in the context of a collaborative application (RoomPlanner) to decide the layout of furniture in a room. Their evaluation notes the use of the thumb and index finger as the most convenient for fine adjustments.

Kruger et al. [62], perform a study to understand the effects of orientation of elements in a co-located tabletop collaboration context. Participants, seated around a table, are asked to solve a puzzle that requires moving and rotating pieces of paper. A video analysis of their performance reveals that orientation of pieces affected

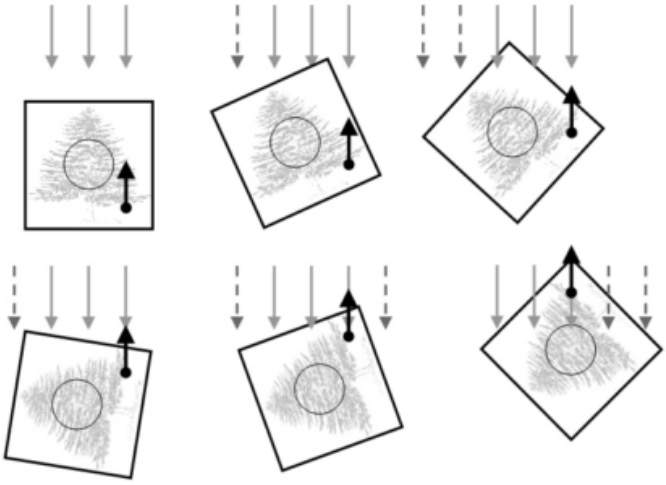

Figure 2.4: Kruger et al. [63] present a single touch technique for integrated rotation and translation. Touches in the circle perform only translation.

comprehension, coordination and communication amongst teams. Performance improved when elements were oriented towards each participant. When passing to another person, re-orientation of the element helped improve coordination.

They use this result as a motivation for research and development of a single-contact interaction technique [63] that uses simulated force to integrate the rotation along with the translation of an object (see Figure 2.4). The tasks for evaluation required element targeting and passing to members across a table. Their metrics were of accuracy and qualitative user feedback, compared against a baseline technique that uses corners of elements as rotation handles. The unified technique they developed was shown to outperform the baseline technique. They extend their work to use multiple fingers for the context of rotation of 3D virtual objects [39]. Through a series of passing and docking tasks, they show quantitative and qualitative evidence that using more fingers allows for a more natural and flexible interaction.

Davidson and Han [24] explore the novel input dimension of pressure to explore layering of virtual items along with the development of a peeling metaphor to access

occluded elements. Reisman and Han [96] also develop new bimanual multi-finger interactions for 2D and 3D object manipulation, without reporting on an evaluation of their performance.

Although integrating rotation, translation and scaling results in a more natural behaviour, the multi-touch interaction modality makes it difficult to specify only one of the three operations. Scaling an object with two fingers will cause a small amounts of rotation. When performing a rotation operation, the location of the visual changes. For design tasks, precision is necessary. Nacenta et al. design and compare four techniques to improve performance of the integrated techniques [88]. The evaluation tasks required participants to perform combinations of rotation, translation and scaling of objects. Compared to a baseline of an unconstrained manipulation, where the finger movements map directly to the object, all of their techniques were shown to reduce errors, but not improve time for completion.

With interest in interacting in more than two dimensions increasing, researchers attempt to identify the impact for constricting the degrees of freedom in mappings. In an analysis of an experiment involving 3D manipulation tasks, Martinet [76] finds that separating rotation from translation results in higher coordination, thus improved control on the individual actions. By using a different number of fingers, the user can choose to either rotate or translate the object. A single finger performs only translation. Rotation of an object requires the use of more than one finger. Instead of using sophisticated techniques like in [88], number of fingers in contact with the screen makes selection of operations easier, and avoids accidental performance of other operations.

Figure 2.5: 'Bumptop', a physically based desktop prototype from Agarawala and Balakrishnan [2].

2.3.2 Physics simulations

Building on the success of using touch for direct spatial manipulation, researchers extend the techniques from single to multiple concurrent elements. By considering multiple elements, the set of commands that the user would like to perform increases. The new commands include: a) selecting and forming groups of elements, b) adding and removing elements from groups, c) affecting multiple elements simultaneously.

Agarwala and Balakrishnan [2] improve on the direct spatial manipulation paradigm by introducing gravity, friction, piles and gravity for organizing groups of elements. Though the project initially was designed for the pen, the implementation also supported touch. It used two fingers for spreading of piles, and flicks to add items to piles. Figure 2.5 shows the collection of items arranged into messy and neat piles. A

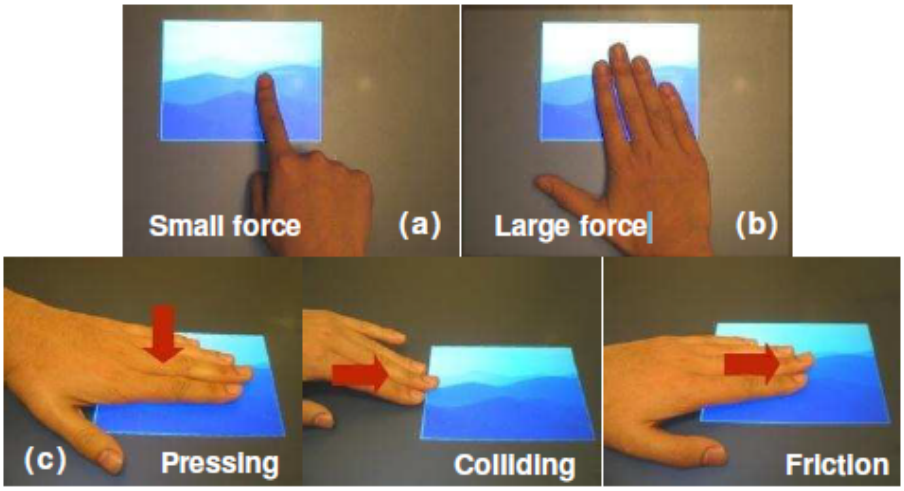

Figure 2.6: Cao et al. [18] demonstrate mappings of contact postures to varying physical force.

study with six users was performed, wherein they explored the system while thinking aloud. Qualitative feedback showed that the users were excited, learnt the techniques easily and appreciated the smooth interface.

ShapeTouch [18] introduces more dimensions of input by identifying properties of the finger contacts, and using it to distinguish the action on pile of cards. One finger translates the top card of a pile, while two fingers spreads the pile out, the addition of a third finger moves the pile as a whole. An initial exploratory user evaluation showed that users liked the ability to use hands instead of fingers, and that they could interact with multiple controls at once.

Wilson et al. [119], explore strategies to aid in the control of virtual elements within a physically based system. They describe three different strategies: joints, proxies and particles. These techniques presented strategies for representing the contact of the hand on the surface within the virtual world. A physical simulation applied on the virtual elements interacted with the representation of the contacts.

The techniques are evaluated with six users using positioning, sorting and steering tasks. They present quantitative results in terms of task completion time, as well a discussion on how the different techniques may be combined.

2.4 Gestural interaction

Research in human-computer interaction deals with the unambiguous understanding and processing of human intent. A goal of *natural user interface* [1, 116] is to the human to express their intent comfortably, using actions we commonly use in the world. The mappings to non-spatial operations are not self evident. We consider research on techniques for more complex operations. Sensing technologies additionally can sense different parts of the hand that make contact with the screen, such as palms, or the side of the hand. We describe research that explores the use of these additional dimensions of sensing. Another framework of analysing emerging interaction paradigms is that of the *reality based interfaces* (RBI)[53] that puts forward the main theses of transferring skills from the real world : naive physics, awareness of the body, environment and social circumstances. They discuss the tradeoffs and limitations of designing with the RBI approach.

We return to the RoomPlanner application [123], wherein an open hand is used for sweeping virtual items (see Figure 2.1). Here the posture of the hand is identified as a signifier of intent. Instead of using a finger to point to a single element, the open hand affects multiple elements. This resulting user experience resembles the movement to sweep items off a physical desktop. Command invocation at the target through stylus interaction was describe by Rubine [100].

Wu et al. [124], use the DiamondTouch table to explore possible mappings of the hand for activities beyond the literal. Their implementation, a photo browser application, enables annotation, wiping (erasing), cut/copy-pasting, and the expansion

and collapse of piles.

They define phases of multi-touch interaction as registration (identifying the posture of the hand), relaxation (the user can relax their hand posture once identified) and re-use (contextually redefining mappings of a gesture). The non-dominant hand is used to establish the context of interaction, while the dominant performs the operation.

Figure 2.7 illustrates how the hand postures are re-used across multiple operations. A qualitative survey through a post-study questionnaire established the user comfort with the techniques. Users report that the pixel-accurate selections were difficult to perform.

Wigdor et al. [113] use the terms *rock* and *rails* to define the hand postures of the closed fist and vertical open respectively. They use these static gestures to aid in the precise manipulation of elements on a canvas. The non-dominant hand performs these postures, while the dominant hand positions the elements. This technique demonstrates a combination of gestural techniques with direct spatial manipulation techniques of rotation, translating and scaling.

The gestures enable an improved precision of the manipulation. A qualitative evaluation presented eight participants with randomly sized objects, that were to be re-arranged to a final prescribed layout. All but two participants preferred the rock and rails techniques over traditional methods.

2.4.1 On the design of gestures

Considering that most commands lack a literal obvious real-world counterpart, their association to a hand movement is an important interaction design topic. One process of development of this mapping is described by Wobbrock, Morris and Wilson [83, 120]. They use a *participatory design* approach to elicit a gesture vocabulary

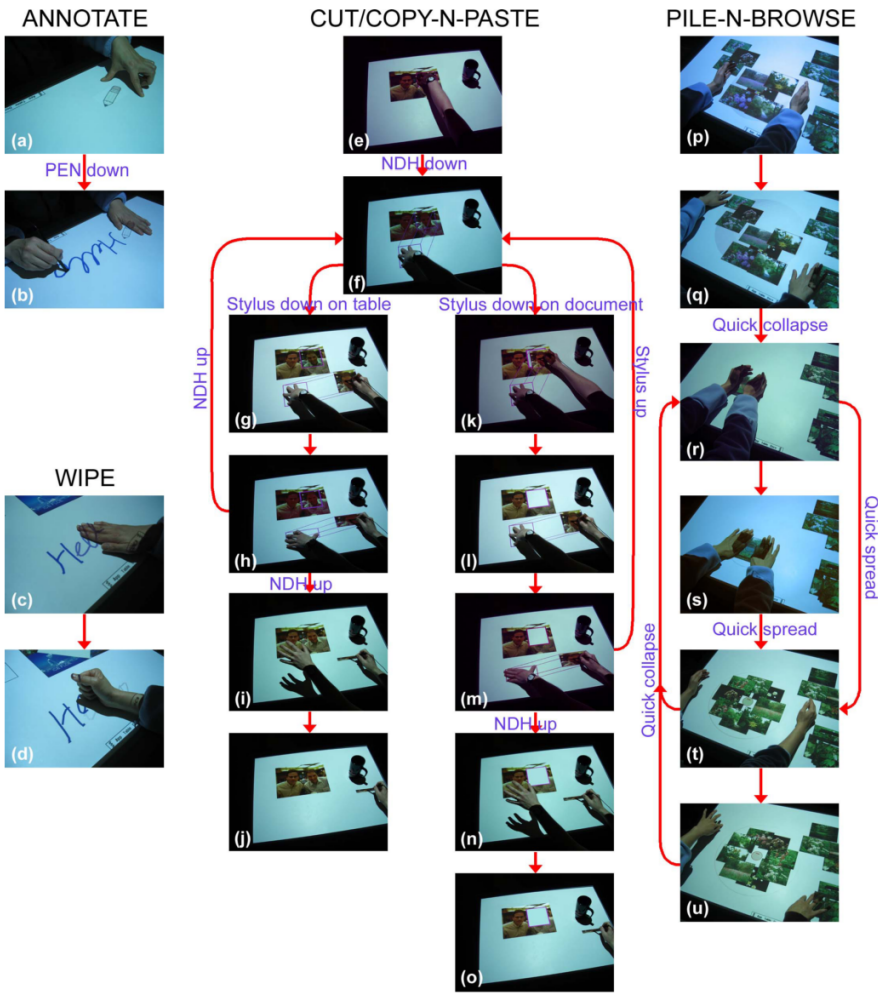

Figure 2.7: A state diagram showing phases of a gestural interaction [124] . Hand postures register the command, user relaxes them during operation and reuses them contextually for other commands.

from the users, without first imposing a rigid set of pre-defined gestures.

A qualitative study is performed with simple graphical shapes on screen, and the user is simply asked to perform a set of commands. The set includes non-trivial commands such as *duplicate, activate menu, accept, reject, minimize* and *help*. They prescribe design guidelines on the selection gesture mappings. The resultant gesture vocabulary is said to be a *simpler* than the typical complex gestures that are developed by HCI researchers.

One of their reported findings states that their users did not consider number of fingers to be of importance in the gesture. However, this contradicts findings by other researchers [18, 23, 39, 96], whose evaluations show that number of fingers can be beneficial to the selection of separate operations.

2.4.2 Learning & discoverability of mappings

As we improve our understanding of gestural interaction, the vocabulary of hand postures available to the designer increases. By combining the different postures, with dynamic changes over the duration of their performance can lead to movements that are difficult to describe.

Octopocus [7] developed for mouse and pen input introduces the concepts of dynamic guides and *feedforward* visualizations for single-stroke gestures. Based on the trajectory of the cursor, the visualization is modified to show possible gestures. The *feedback* was the ink left by the stroke the user has performed, while the feedforward showed possible paths available to the user.

Multi-touch interaction has a more complex problem. The posture of the hand may not be as interaction designer expects (for example a vertically placed palm instead of a closed fist). Improper hand posture can make some gestures difficult to perform. Visual representation of posture as well as performance of the gesture is

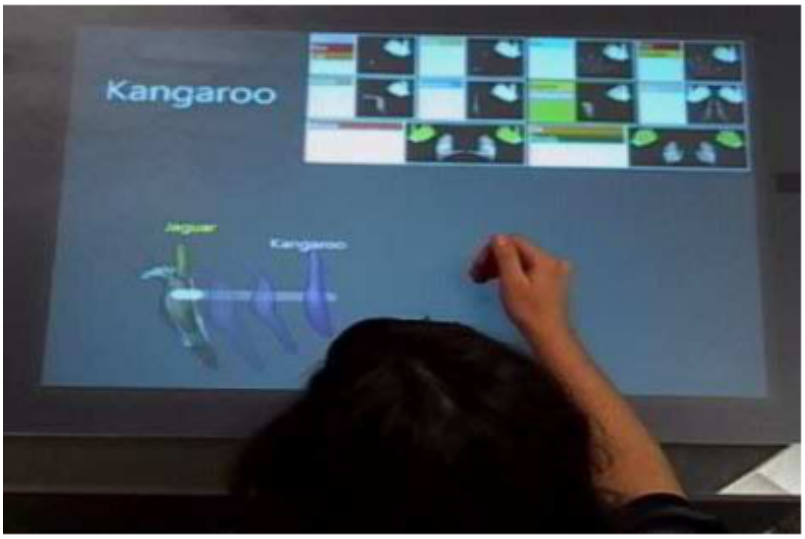

Figure 2.8: Illustration of the visual presented to the user from [33]. The *feedforward* mechanism shows two gestures from this starting point: Jaguar and Kangaroo.

non-trivial.

ShadowGuides [33] introduces a new visualization that is aimed at teaching new gestures to the participant. The Microsoft PixelSense [49] is used as the hardware device, providing accurate shape of the contact. Figure 2.8 shows the visualization. The feedforward visualization is displayed adjacent to the user's hand. A detailed menu shows the set of gestures, along with the part of the hand that is used to perform each gesture (see top right). They begin with a description of the taxonomy of surface gestures. The three axes of classification are registration pose (number of fingers or shapes), continuation pose (whether the pose changes during gesture performance), and movement (whether the hand stays in the same place).

A set of gestures are designed to demonstrate parts of their taxonomy. These gestures are mapped to artificial items (animals) for the benefit of a user study. The study shows that fewer memory and performance errors are seen when using their technique as compared to the baseline of video demonstrations. However, the paper

lacks a discussion on the design and mapping of gestures to real-world operations. Using such complex gestures in a real-world application requires an unambiguous understanding of the mappings not just by the designer, but by the human participant.

Using play to alter the perceived cost of learning, Gesture Play [14] describes a physical simulation with springs, dials, and props to teach specific gestures to the user.

2.4.3 Gesture recognition

Gesture recognition research is not by itself novel, its application to touch has typically meant borrowing from sketch recognition [99, 70, 69] to adapt to single-stroke recognition algorithms [121, 102]. Multi-stroke recognizers have also been developed, but only if the strokes were performed sequentially, not simultaneously as is the case with multi-finger gestures [72]. With regards to commercially prevalent multi-touch gestures, the implementations usually are hard coded. For example with the pinch-to-zoom gesture: Event callbacks and heuristics are used to determine if the two fingers are placed on the same graphical element. Subsequent moves of the fingers are then used to calculate the zoom amount. This approach is infeasible when the goal is to establish a learn-by-example gesture system.

Yamoto and Ishii developed methods for using hidden Markov models to identify human action using video as the input medium [125]. Morimoto et al. developed more contextualized work, using HMMs to classify facial movements as Yes or No gestures [82]. Wilson and Bobick developed parametric HMMs for gesture recognition [117], to allow an operation to be used by an interactive application.

New research describes mechanisms to recognize a range of multi-touch gestures. Proton++[61] prescribes a regular expression approach to define the details of a new

gesture. A tablature of basic primitives with touch events (up, down move) and directional components can be used in a specified syntax. Gesture Coder [71] on the other hand can learn-by-example. A specialized state machine is automatically learned from a set of input samples. Both of these research products allow for parameterization of the gesture, through specialized means. It is also important to note that the parameters of the gestures in these multi-touch recognition systems are designed to be used for spatial manipulations. Both systems are designed to allow for free-form movement during the gesture to map to operations such as panning a map.

2.5 Explicit command selection

Techniques described in Sections 2.3 and 2.4 are feasible when a clear mapping exists between the command and the movement that the human must perform to invoke it. Changes to spatial attributes are a great fit for direct manipulation techniques. The mappings from body movements to changes in the visual are clear and explicit. The experience from using the techniques can be considered natural to the human.

The question of what mappings are obvious is not easy to answer. Although mappings can be made, there isn't a clear deterministic design process that allows for the production or *choreography* of the gestural mappings to an arbitrary set of commands. What may be considered natural or intuitive for one person, may not be for another. The context of the mappings may have a similar effect, the gesture may be more natural when mapped to a different operation in a different context. This leads to an ambiguity in selecting the most obvious gesture. This subjective variance in human opinions makes designing a vocabulary of gestures for complex actions a difficult problem.

Considering more complex commands such as picking a color, fetching mail, altering volume, or creating a new folder in a file system. It is interesting to note that related research, shadow guides [33], avoids to topic of gesture mapping by selecting completely arbitrary context for the set of gestures: a list of animals.

As the size of the command set for an application increases, the discoverability of the gestures becomes less obvious to the user. The cognitive overhead of memorizing the gestures increases. Taking a clue that signified the paradigm shift from CLIs to GUIs, this overhead can be reduced by encouraging recognition over recall: visualize possible commands in a clear and consistent manner to the human.

We call this form of command selection *explicit*. The user must be able to *navigate* to a commands from within a menu, making available commands *visible*. Traditional menus are functional, and can be operated by touch interfaces. Although research presented in section 2.2 could be used for alleviating some of the problems, we need new fluid multi-touch techniques for command selection.

The introduction of a visualization of the menu in some form at the finger contact. We describe two categories of menus, 1) stroke-based marking menus, that may or may not have a visual representation of operations at all times, and 2) visual menus that emphasize a consistent visual representation during their operation.

2.5.1 Marking menus

Stylus based interaction has been more prevalent than touch, and hence has a considerable depth of research dedicated to accessing commands via a menu. Kurtenbach and Buxton develop *marking menus*[64], that use direction of a pen-stroke to select menu items. Each directional stroke selects items from subsequent levels in the menu hierarchy. Chaining strokes together allows selection from a deeply nested menu. The motivation of this technique is that the strong spatial memory of humans

aids in the repeated selection of menu items.

Initially, a visual radial menu shows upto eight top level menu items, located in segments of the circle. A first directional stroke selects the top level item, which prompts eight subsequent menu items to be shown. A second stroke performs the final selection. Thus with two strokes, a selection from upto 64 elements can be made. The visual menu can is presented with a delay. When the user pauses at a certain stage, the available commands are presented in a circular menu around the cursor.

Through repeated use, the user learns the spatial location of menu items, and the strokes required to access them. The length of the stroke need not be finely controlled, allowing for a less constrained movement of the hand during operation.

Expert users can perform the strokes for an item from memory, instead of waiting for the menu to be displayed. Another crucial aspect of the marking menu is the lack of visual focus required to operate it. The strokes can be performed anywhere on the screen while the user can keep her visual focus on the target being manipulated. This technique has a quantifiable performance gain over regular menus, that requires precise spatial selection of a menu item. An evaluation [91] comparing marking menus with the toolglass [13] and hot keys shows several benefits of marking menus over the toolglass.

Variations on marking menus [64, 65, 129, 128] have been shown to have a high performance with both the mouse and the pen. The summary of research into stroke based gestures is beyond the scope of this paper. All of these techniques could be applied to single-finger interaction. These techniques are aimed at facilitating a smooth learning curve for a novice user to become an expert. The novice begins by using the visual menu. Repetitions of item selection during normal use encourages learning. Their evaluation tasks involve numerous menu item selections using comparable

Figure 2.9: A selected set of multi-touch *chords* for top level item selection from Lepinski et al. [68]. When each of these are combined with a directional stroke in one of eight directions, upto 64 items can be selected.

techniques. The measurements of time to completion and rate of errors in selection are used as the golden rule for technique adoption.

Lepinski et al. [68] translate this work from the domain of stylus / pen based interaction to multi-touch. While regular marking menus use directional strokes for each menu level, they explore the use of hand chords for the top level menu selection. A chord is identified by a detecting which combination of fingers make contact with the screen. Figure 2.9 shows the final set of chords selected for the user study. Finger disambiguation techniques are introduced to help differentiate the fingers of a single hand. Their evaluation shows the performance (in terms of time and error) of the multi-touch marking menu to be better than the single touch marking menu.

However, from personal experience, attempting to quickly alter between the chords repeatedly is physically difficult, and induces undue stress in the hand. Their evaluation neglects to raise these experiential issues that are extremely relevant to their techniques adoption in the real-world.

Bailly et al. [5] extend the marking menus to bimanual interaction. Instead of using hand chords, they use only the number of fingers to encode the selection of the first level. The use of the second hand allows for a larger selection. For selection at the next menu level, they use two techniques, directional strokes and simple multi-finger tapping.

The motivation behind this set of techniques is that with some training, a human can remember spatially, and kinaesthetically (with muscle memory) the mappings between a hand movement and a command. A short study shows that their techniques initially *do not* outperform the baseline of selection from a standard menu bar. Their reasoning is that the low pixel density of their hardware makes the menu bar larger and easier to select, but higher resolution displays might show different results. Later trial blocks show a convergence in performance of the three techniques. An important finding from this research is the ability for the human to encode information with the number of fingers in contact with the screen.

2.5.2 Visual menus

Considering the pace of adoption of the multi-touch modality in research, it is surprising that, to our knowledge, there have only been a handful of publications that develop novel techniques for presenting visual menus to the user. Considering human perceptual mechanism for organization of the menu [19].

Stacked Half Pie menus [40] use a single touch to navigate a hierarchy, visualized as a semi-circle. A goal of the design is to reduce the distance that the hand must traverse to access items of the menu. Stating the concern of occlusion of the screen by the hands, they place the menu at the bottom of the screen, towards the human. Qualitative reports from an exploratory evaluation show that users had difficulty in understanding components of the design.

Banovic et al. use multiple fingers in their single-handed pie menus [6]. Activated by the thumb, the menu can be accessed using the little, ring, middle or index fingers. Figure 2.10 illustrates the three target selection techniques that are compared on the basis of speed and accuracy for menu item selection. The items in the menu are not hierarchical, but are placed in the partial pie menu. Their evaluation fails to

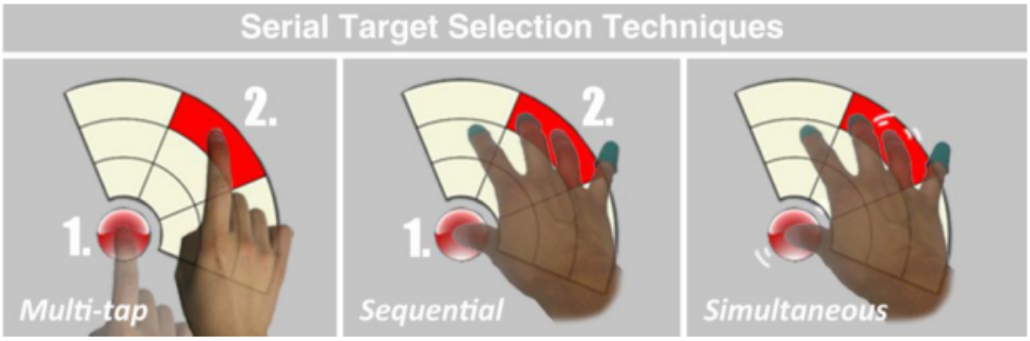

Figure 2.10: Banovic et al. [6] present strategies for selection of menu items in a partial pie menu.

mention the ease of completion of the selection tasks by the participants. From the description of the interaction needed to operate the menu, the human must contort their palm into uncomfortable positions to maintain precision and control.

Attribute gates [107] allow the user to set multiple attributes of an element (given a discrete set of options for each) using a single stroke with co-location tabletop collaboration as the context. Ring menus [59], uses both hands and chords to display available commands.

The most relevant research to our own work is the dissertation of Malik [74]. The focus of his research is the exploration of single-handed multi-finger interaction. In this work, he introduces the use of the thumb to select from a discrete set of items (see Figure 2.11), and also manipulate a slider over a continuous range of values. Several strategies for visualizing the menu are discussed and compared. He presents studies identifying the comfortable human range of motion of the index finger and the thumb. He also demonstrates the use of number of fingers to select a specific operation. However, these new widgets were only an exploratory illustration. They lacked a concrete implementation and evaluation. The quantitative reports are limited to studies focused on gathering performance data of finger movements,

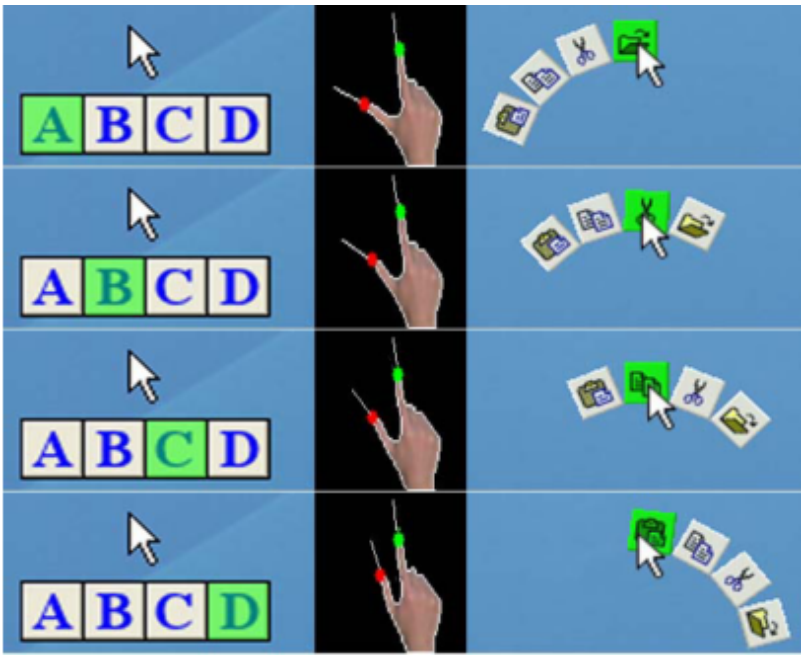

Figure 2.11: Malik [74] illustrates the use of thumb for selection of discrete items.

without the context of the interaction.

The above research fails to compare against a baseline menu, making their evaluation difficult to analyse in a generic context. With the recent spurt of such research, evaluations comparing the newly developed techniques are required. A lack of standardisation in the evaluation makes comparisons difficult.

2.6 Parameter adjustment

The marking menus described in 2.5.1 as well as the prototypes of visual menus considered in 2.5.2 are used for *discrete* item selection. We now consider *parameterized* operations that not only require a selection of a command, but also the adjustment of a parameter associated with the command, for example adjusting brightness. The selection of *brightness* is the command. Further, the user also needs to specify a *continuous* numerical value to complete the operation of brightness ad-

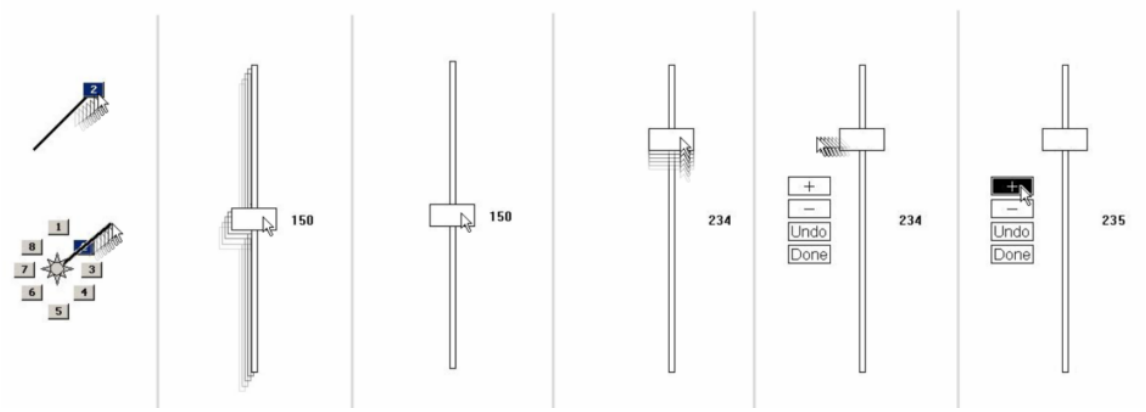

Figure 2.12: FaST sliders merge a marking menu with a traditional slider, presented at the target location. Additional buttons are displayed on hover for increased accuracy and undo capability.

justment. The techniques for command selection by themselves are insufficient to provide a fluid experience to complete parameterized operations.

Pook et al. introduce *control menus* [93], an evolution of marking menus, to perform adjustment of a parameter over a continuous numerical range. In a single stroke, the control menu allows selection of a command as well as the adjustment of a single numerical parameter. A pre-determined threshold distance distinguishes selection of the command and the manipulation of the continuous numerical parameter associated with the command.

In the same year Guimbretière and Winograd present *flow menus* [37] as an extension of control menus. Flow menus perform selection of a command by requiring re-entry into a fixed circular region, as opposed to the threshold distance method of the control menu. This difference has the added advantage of reducing accidental command activation. For continuous numerical adjustment, the menu is operated like a knob, with circular clockwise movement to increase and counter-clockwise to decrease the value.

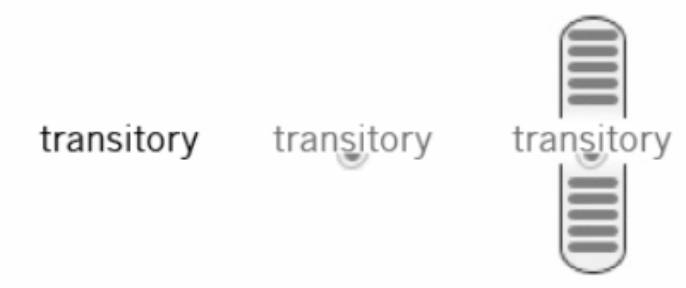

Figure 2.13: The three stages of activation of the in-context slider by Webb and Kerne [111] for parameterization of an interest value over a target.

A laboratory study by Guimbretière et al. [35] present a comparison of the control and flow menus. The study had two additional techniques for comparison: (1) a movable tool palette presented as a traditional panel of items, and (2) a two-handed toolglass that allows participats to use one hand to independently move a transparent color palette. Participants were asked to connect a set of 12 points, after selecting a specific color for each point. The results showed that techniques which merge selection of the command with adjustment of the parameter outperformed the other techniques.

FaST Sliders are designed for in-context, precise adjustment of parameters for mouse input [79], focussing more on precise adjustment of a parameter rather than selection. Parameters are selected through a marking menu. Figure 2.12 shows the stages of operation of the fast slider, with selection via marking menu, followed by coarse adjustment by sliding, and fine adjustment through the use of buttons. The hover state of the mouse is used to fluidly chain the selection of the parameter with its adjustment. A click on the slider thumb or on the done button ends the manipulation. SlideBar [21]

Webb and Kerne present a mouse based in-context slider, activated at the target, to express interest on a term, image or text clipping [111]. A transitory slider is

visualized over a visual in three stages, (see Figure 2.13). This technique is designed for a single parameter on the target, and can be performed with one less click. The slider is fluidly operated by hovering (a) over the target, then (b) over the half-navel presented and finally by clicking on a desired value. A user study uses a baseline of a traditional slider in a popup modal window. The task requires participants to rate words, text and images. Results show that participants were faster with the in-context slider and found it easier to use.

symTone by Latulipe et al. [66] utilizes the tone reproduction curve visualized over the target image as an affordance for manipulation. This technique may not be feasible for parametric spaces of other graphical effects.

Engaging in visual design tasks often requires users to adjust multiple parameters for an associated action, such as those for image level (brightness, contrast, saturation and opacity) or drop shadow (blur radius, direction, opacity and depth). These are typically presented to the user as a set of sliders. Parameter adjustment requires the user to shift visual focus away from the target to acquire the thumb of the slider. Visual design often requires iterations of the following sequence of subtasks: parameter selection, adjustment, and observing the effect on the target. The duration of each iteration is further extended when the user needs to switch between different parameters. This leads to to-and-fro visual saccading between an interactive component and the target object, even if the interface is located close to the target.

These distractions of repeated saccading during tasks with high cognitive loads, such as visual design, lead to break downs in the user experience. This context is similar to that of visual analytics, wherein the visual is not a design, but a visualization of large sets of data. The problems describe above apply to interactions that require problem-solving and analysis of data. Our research attempts to develop new

forms of interaction for manipulation in such multi-dimensional abstract parameter spaces, that do not have an obvious 2D or 3D spatial representation.

The techniques for spatial manipulation use *natural* mappings to target a small subset of the parameters (location, angle, scale). Specific actions such as piles, copy-paste, and layout have the advantage of having mappings that are unambiguous. Parameters such as brightness or blur radius do not have such obvious mappings.

Damaraju and Kerne [23] use the term *abstract parameter spaces* to describe the multi-dimensional space of a set of parameters that do not have a literal spatial representation. From related work [16, 54], they show that the user context of graphical editing is focused not on navigating quickly to one particular parameter value, but the fluid and oft-repeated re-selection of a small subset of the parameters at a time. They present findings on a user study to manipulate parameters of color (hue, saturation and value) using single handed multi-finger mappings. Their technique shows an improvement over a base line of direct manipulation, that requires pointing to the desired parameter value on a displayed visual menu. The multi-touch technique uses number of fingers and simple finger disambiguation to allow quick repeated selection and adjustment of one of the three parameters.

2.7 Form factor and visual focus

Our presentation of research has considered the context of the interaction techniques developed, without much emphasis on the physical experience of the human. In research using the DiamondTouch table, we mention that the users are seated around a table. Whole handed interaction only used with devices that can support robust detection of shape of the contact. However, their techniques are unique to the hardware. Wide-spread adoption of techniques in the mainstream is limited.

Moscovich and Hughes[85] perform a detailed analysis on single and two handed

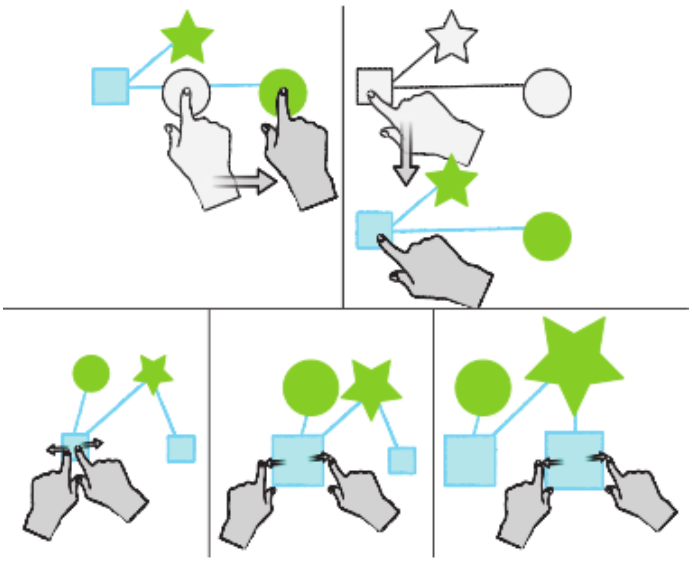

Figure 2.14: Indirectly affecting objects through a proxy, from Wigdor et al. [113].

interaction for rotation and scaling. Although research on bi-manual interaction is extensive [60, 67], this study is specifically geared to the understanding of the effect of *direct vs. indirect* manipulation. Using the metric of coordination, Moscovich finds that bimanual techniques require a strict visual correspondence of the movement of the fingers to the rotation of the object in virtual space. The findings suggest that a gain function can be applied to unimanual operations without a degrading performance significantly. Earlier work by Everitt et al. uses the term modal spaces [30], to describe how the same movement performs different gestures based on the region of the screen in which it was performed.

Wigdor et al. use indirect manipulations to a large extent with the rock and rails technique [113]. They use the concept of proxies, visual shapes connected to a single or multiple target by a line, to promote indirect interaction (see Figure 2.14). The proxy allows distant objects to be operated upon. The proxy supports all the operations that can be performed on the target directly, translation, rotation and

scaling. The proxy reduces the problem of occlusion by the human hand. A unique advantage of their use of proxies allows a one hand movement to affect multiple items at once.

Similarly, the abstract parameter manipulation described in [23], the mappings allow indirect manipulation. The advantages of this are two-fold: (1) avoiding occlusion of the target, and (2) adjustment can be made without necessitating visual focus on a visual menu.

2.8 Evaluation techniques

The applicability of interaction techniques addressed by this paper is dependent on the methods as well as results of their evaluation. The stated goal of the techniques is to *perform better*, which as per the current state of research involves an evaluation of the time to complete the task, and it's accuracy while doing so. We discuss the motivation and differentiate our approach and goals to the design of multi-touch command selection techniques.

Due to the field of multi-touch interaction being fairly recent, a number of publications fail to properly evaluate their techniques against a baseline. For example, [24] and [96] have no evaluation. Further, menu techniques such as [40] only have a usability evaluation. This scratches the surface of how we can expect the techniques to succeed in real-world applications.

Innovation in interaction techniques requires that we first need to understand the flaws and benefits of existing techniques. For an exhaustive evaluation of new techniques, we require a baseline to compare against. and a set of agreeable metrics that can quantify the performance.

2.8.1 Context and tasks

A majority of the highly specialized tasks discussed in this paper [3, 11, 39, 63, 68, 128, 129] are designed for generic use. To develop techniques for generic contexts, and then evaluate them for real-world usage is a difficult problem. The context of the research, and hence their user studies is to complete the task of selecting the menu item, or precisely manipulate a specific element. The tasks fail to adequately model the mental state of the user in a real-world setting. Several hundred trials are completed by the user in a single sitting. The tasks describe in exact detail what the user needs to accomplish, leaving very little for the user to think about. The human is reduced to a mechanical system, asked to move given a visual stimulus. The motivation of such research is that the optimisation of these fine-grained subtasks will lead to a better overall user experience.

Wobbrock et al. [120] develop a complementary research approach on evaluation. Their study shows how focusing on the user context can provide detailed and helpful guidelines for the design gesture vocabulary. Instead of focusing on the speed and accuracy, they attempt to elucidate the intuitiveness of the interaction from the user. User comfort, both physical (by describing simple hand movements) and mental (the mappings are easy to understand and remember) takes the front seat. By attempting to perform the chorded gestures described in [68], we have realized the futility of designing gestures solely based on what can be faster. Out of personal experience, the awkward hand positions that the chords require, are difficult to perform over prolonged periods of time.

Not all interaction techniques aim to make operations faster. Some of them are aimed at improving learning [33] or making operation less obtrusive to the context of the user [111]. As researchers focus on the larger picture, we require a standardisation

of tasks. Repeatability of studies will make comparison across techniques feasible.

The evaluation table shows a chronologically ordered summary of a selection of research considered in this paper. We present their objectives, tasks, the metrics used as well as the results of their evaluations. Additionally, we consider whether their evaluation used a valuable baseline beyond their own techniques. Some research does not explicitly identify whether the techniques was meant to be used with one or two hands. For example, Martinet at al., [76] describe how some users preferred the use of two hands for 3D manipulation tasks, while other used multiple fingers of a single hand. On the other hand, in the rock and rails techniques, the use of multiple fingers was ambiguous. The techniques used the posture of the one hand, but only one finger of the other hand. This classification will help compare techniques across their stated objectives.

2.8.2 Metrics

The most used metric for comparing techniques of multi-touch interaction are (1) time to complete a specified task, and the (2) number of errors made by the user. When used by themselves, the metrics fail to capture the intricacies of user experience. Some papers extend this by analysing number of operations performed by the user. Minimizing the number of operations is a secondary objective by the researchers. There are several qualitative features that make comparison amongst techniques challenging.

Qualitative evaluation has made use of questionnaires [2, 6, 15] and coded metrics from video logs [62] to capture and understand the user's context and evaluation that is not possible to elicit solely from the quantitative metrics recorded from the task.

When combining operations such as rotation and translation, it is important to understand the level of control a human can exhibit to individually perform either

operation. Coordination during integrated adjustments can be quantitatively identified through studies. Zhai and Milgram [127] introduce a new metric, *inefficiency* as a measure for coordination. The less the inefficiency, the more coordinated the technique is. The authors used this technique to measure differences in how humans exercise control over experimental hardware that provide six degrees of operational freedom (along X, Y and Z axes). This metric of coordination was used in later studies [23] to evaluate performance in abstract parameter spaces.

Metrics need not be used solely to show an improvement of an author's technique over a baseline. Discussion of results using a new metric may be used to gain an analytic understanding of use. For example, Nacenta et al. [88] use normalized time, presented as a percentage of gesture duration, to look at the temporal distribution of the operations of rotation, translation and scaling. The metric comes from outside the field of human-computer interaction from Mason and Bryden [77] in the journal of experimental brain research. For this particular research, the metric may not be of primary significance. However, the knowledge of such a metric could encourage other researchers to re-evaluate and analyse their techniques in a new light.

For the following forms of interaction, we suggest visiting the references listed. A discussion of these techniques is outside the scope of this survey.

- Tangibles [52, 97, 31, 56]
- Free air / distant interaction [81, 118, 36, 75, 12]
- 3D Volumetric displays [34]
- Under water [27], Under table[114].

2.9 Section summary

In the early days of touch based interaction, sensing technologies were in large form factors, either as a wall or horizontal table-top configurations. This prompted

the focus of initial research primarily towards co-located collaboration. The sensors granularity was coarse, hence the gestures developed involved palms, open hands and coarse movement. As sensing improved, the form factor became more compact, and finer grained. A multi-touch device could fit on a users work desk, and later into their palms. Techniques began exploring the use of fine-grained skilled finger movements. Research approaches were shaped by the resources available at the time. It is evident that a majority of the current research has a focus on developing *direct spatial manipulation*, the lowest hanging fruit of the multi-touch modality. The evaluation of such techniques prioritizes optimisation of the time to complete the small sub-tasks.

This paper considers a breadth of research that utilizes the multi-touch modality for selection of commands. Techniques of command selection beyond the literal are also considered. Direct spatial manipulation is used to define the selection of commands that adjust spatial parameters of visual elements. Research on gestural interaction is presented as a form of implicit command selection.

Through an analysis of the prior work, we gain an understanding of the difficulties in producing standardized evaluation methods that can be used as baselines for comparing command selection techniques. The demarcation of spatial and non-spatial operations is clear, the techniques for either of them cannot be evaluated in the same way. Similarly, even within non-spatial operations, the presence of a numerical parameter complicates the operation.

We presented our motivation, in Section 2.6, for designing techniques for parameter adjustment discussing the importance and uniqueness of the problem. Visual design and analytic applications have a large set of commands, and benefit from maximising their screen real-estate. The standard user interfaces for parameter adjustment (menus, buttons, sliders) use up crucial pixels, taking away space that can

otherwise be devoted to the context.

Our approach values indirect manipulation, so the user's hand can be as close to the target object as possible, allowing visual focus on the target, while performing the manipulation of parameters. We note that our goal isn't just to make the interaction faster, and more accurate, but also fluid, comfortable, and encouraging exploration of the design space.

3. INVESTIGATING MULTI-TOUCH MAPPINGS[1]

The term gestural interaction is used for any interaction that yields in a natural user experience for non-traditional inputs (other than keyboard and mouse). In the previous section, we develop a recognition technique as a foundation for developing such natural user interfaces. In our related work (Section 2.4), we see that a gesture is primarily used to perform a discrete command selection. The command may also be parameterized if it pertains to altering the spatial features of the visual such as zoom, scale, and translate.

Norman defines *mappings* are the relationship of movements of the body and the resulting operation in the virtual space [89]. The term gesture implies a distinct correlation between the movement and the operation. When the space of commands is large, this correlation becomes subjective. It is important to distinguish between the concept of a gesture and that of a mapping.

Directional swipes are often termed as gestures, but are mapped to a varied set of operations. A four finger vertical swipe on the Apple trackpad [4] can be mapped to bring up all open applications on one screen (called exposé[46]). Although the finger swipe can be called a gesture, in reality it is really mapped to an arbitrary operation. The correlation between the movement (four fingers moving vertically) and the operation (exposé) is ambiguous.

Direct spatial manipulations for touch interactions prescribe *direct* mappings, where movement of the body is performed directly on the target. Movement of the hand has a one-to-one mapping to the that of the target. With translation on a touch

[1]Part of the data reported in this chapter is reprinted with permission from "Comparing multi-touch interaction techniques for manipulation of an abstract parameter space" by Sashikanth Damaraju, and Andruid Kerne, Proceedings of the International Conference on Multimodal Interfaces, 221-224, © ACM 2011.

screen, the object is expected to stay under the finger during the operation, which can get tedious for larger screens. As we begin considering to non-spatial spaces, such direct mappings are infeasible. A parametric space is not visually represented in a two or three dimensional space.

Moscovich [85] describes the multiple advantages of *indirect* mappings for touch: occlusion of the hand, and freedom from the one-to-one mapping of the movement of the finger to that of the virtual object. Occlusion is reduced by allowing the hand to operate away from the target. The indirection no longer necessitates a direct mapping between the motor space of hand movement and the visual space of the target. With an indirect mapping, we can apply a *transfer* function to map between the two spaces, enabling traversal of larger virtual distances with the same amount of movement by the finger.

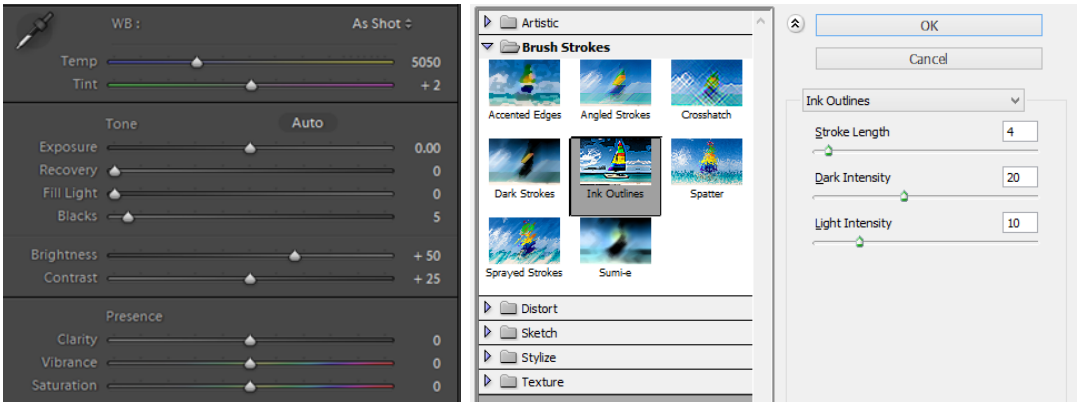

Figure 3.1: Examples of non-spatial parameter spaces in image editing. On the left are a few of the parameters in the develop menu of Adobe Lightroom. On the right are three parameters of the brush filter *ink outlines*.

3.1 Non-spatial parametric spaces

A parameter space is the set of all possible combinations of values for all the various parameters. When considering spatial parameters, the space is typically defined by position (X, Y and Z coordinates) and size. We define the term *non-spatial* parametric spaces to all parameter spaces that exclude those defined with spatial attributes. We present below a few examples from research and commercial applications.

The examples of parametric spaces shown in Figures 3.1 and 3.2. The parameters we see are integral to the operation, each of them alter the effects of the visual in some way. When performing such operations, it is more important for the result of the operation look a certain way, rather than precisely setting a parameter to a specific value.

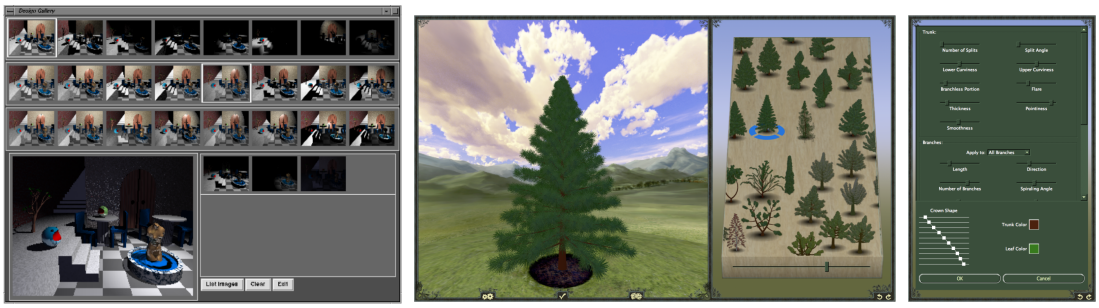

Figure 3.2: **Left**: A *design gallery* is presented to set the lighting of a scene. The three rows on top represent the amount of three different lights, shown as a series of small multiples, with the resultant stage on the bottom left. **Right**: Talton et al. [109] presents a visualization to explore the space of the types of trees. A panel of sliders alters parameters of the tree trunk, branches and leaves.

3.2 Color selection

We began our exploration of multi-touch beyond the established definition of gestural interaction. Our approach focused on understanding the impact of altering *mappings*. In this section, we present a user evaluation that studies the effect of new interaction techniques. In this case, we choose color selection. From our prior work, we differentiate our research away from spatial manipulation. The *natural* mappings in the spatial domain are defined relatively objectively. Various heuristic algorithms [88] and interaction techniques [96, 63] integrate rotation, scaling and translation to avoid disambiguating the intent of the interaction. In the non-spatial domain, the mappings are not as easily understood.

Color is a multi-dimensional non-spatial parameter space. It can be expressed with a different set of parameters:

1. **Red, Blue, Green (RGB)**: The most commonly used form of color space in the computational domain. Monitors and most televisions contain a triad of color components for each pixel. The power to each of the components of this triad determines the color of a single pixel.

2. **Cyan, Magenta, Yellow, Black (CMYK)**: Print media commonly uses four colors of ink. Varying amounts of ink produces the entire color spectrum.

3. **Hue, Saturation, Value (HSV)**: This color space splits the color spectrum into a more perceptually understandable form of color. Hue defines the color, saturation defines the amount of color, while value (sometimes called lightness) defines the

4. **CIE L*a*b***[43]: This color space that was defined to approximate human vision, with change in value in this space would correspond to change in about

the same in visual importance.

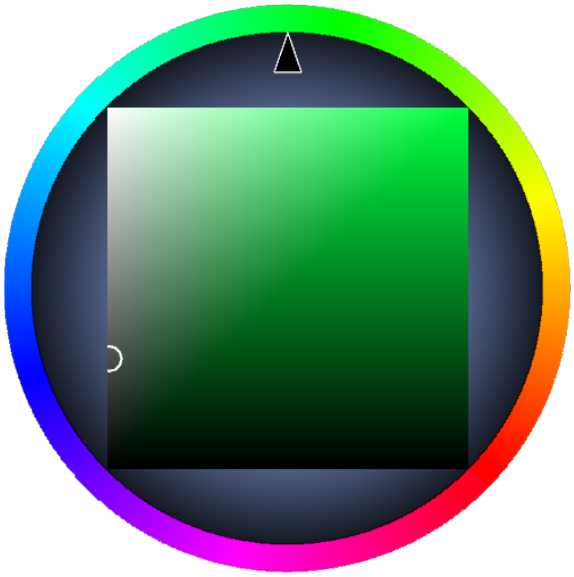

Figure 3.3: The displayed visual for color selection in the HSV color space. Hue is represented on a ring. Saturation and value are the two axes for the square gradient in the middle.

3.3 Interaction techniques

For our research we select the HSV space since it is easily to describe to participants and implement in an application. The visual feedback of color selection is shown in Figure 3.3. In a square at the center, we represent the saturation-value varying subset of the HSV color space for a given hue. A ring around this square visualizes the range of hue. The thumb is differentiated from the index finger by being closer to the user (lower on the screen). The motor space for the hue is the same as that of saturation and value, i.e., to move from one end of the range to the other, the participant must move the same distance (the side of the square) for all

three color dimensions. The multi-finger conditions described below (C1, C2) have a common mapping for hue variation the horizontal movement of the index finger is mapped to change in hue, while the thumb is used as a pivot.

The multi-touch techniques develop variations on the mappings of finger/thumb movement to color dimension. We design interaction techniques to enable the user to effectively manipulate the abstract parameter space of color. The interface widget in Figure 1 helps reinforce the mappings from the movements of the fingers to the adjustment of a specific parameter hue, saturation or value.

In each multi-touch technique, interaction begins when the user places two fingers anywhere on the screen, without depending on touching a widget in a certain place, and ends at the last adjustment made by the user before touching the done button.

C1 **Constrained Manipulation**: (See Figure 3.4a) The mappings are as follows: horizontal movement of the index finger, with the thumb as a pivot, is mapped to change in hue. Horizontal and vertical movements of the thumb, with the index finger as a pivot, are mapped to change in saturation and value respectively, however only one of saturation or value can be manipulated at a given time. The components of the velocity vector of the thumb movement decide the parameter to be manipulated.

C2. **Index+Middle**: (See right of Figure 3.4b) Horizontal movement of the index finger pivoting from the thumb controls hue. But now, the index and middle finger together are dragged horizontally to adjust saturation, and vertically for value. We expect that by shifting the fingers in contact with the screen, selection of parameters can be made with little effort.

C3. **Mouse Interaction**: This is a control condition that allows a practical comparison with the multi-touch conditions. The circular marker in the saturation-

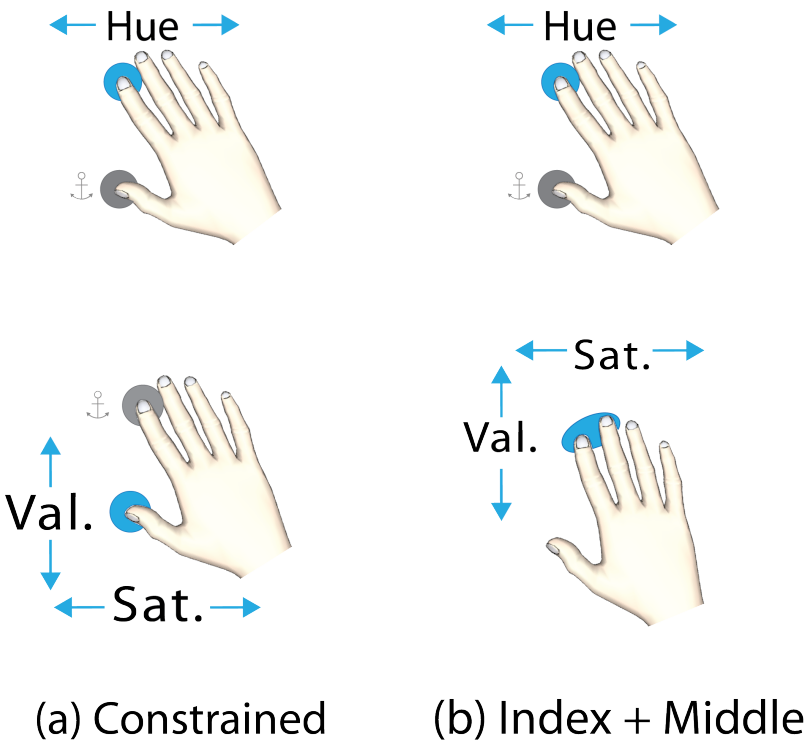

Figure 3.4: An illustration of the multi-touch techniques for selecting parameters of the color. A blue dot indicates the finger(s) moving. An anchor indicates that the finger is in contact but not moving.

value swatch moves by making clicking at a specific point on the square. Participants click directly on the desired value, either on the two dimensional saturation-value color square or on the hue ring, which is not possible with the previous conditions. We expect users will be slower in this one parameter at a time manipulation condition than in the multi-finger conditions. We use the mouse condition as a baseline to compare the effects of the multi-touch interaction techniques on task performance time.

3.4 Experimental hardware

We fabricated a 55" touch-screen that used FTIR [38] for multi-touch sensing. The hardware device was designed to be adjustable in angle and height (See Figure 3.5). For the purpose of the experiment, we selected a 30° angle, with the edge facing the user at a height of 36". This allows the participant to comfortably rest their elbows or forearms on the 2" bezel of the screen. The participant was seated before our hardware, and was allowed to adjust the height of the seat to make themselves comfortable.

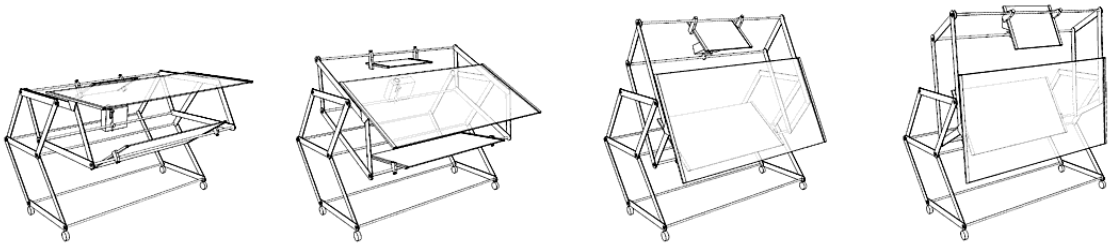

Figure 3.5: Configurations of the adjustable multi-touch sensing hardware we had fabricated. For this experiment, we chose the configuration second from the left. The angle of inclination was set at 30°.

3.5 Evaluation

We conduct a within-subjects study to compare the efficacy of techniques that map movement of fingers of a single hand to the adjustment of multiple parametric dimensions. The independent variable is the interaction technique used, and the dependent variable measured is the time taken to complete the task. We focus our attention primarily on the index finger and thumb because they are motorically characterized by highly independent degrees of freedom. The independent variable

has five conditions, described below. The participant performs multiple trials (a trial is selection of a single color) with each condition.

Participants are presented with the initial and final colors as two squares above the color picker widget. Color matching is a cognitively demanding task. Our perceptions of color are subjective, with considerable variation across users. A pilot experiment was performed with two participants. The numerical values of the target color were hidden, with the expectation that an exploration of the color space would provide a more natural real-world setting for the color selection. This resulted in several trials left incomplete, the participants were too frustrated to continue. The task did not specify which parameter needed adjustment. They often changed a parameter that was already at the correct target value, making the task much more complicated than we set it to be.

In this study, we display the numerical values of each parameter below both the initial and final color squares. For each trial, participants use one of the mappings described above to manipulate the initial color to match the target color. The current color is continuously updated to match the values adjusted by the user.

Each set consists of 12 trials. A set of 12 initial and target color combinations are randomly initialized. They are common across all conditions and participants. The order of these 12 trials is randomized for each condition. The order of conditions is randomized for each participant by selection from a *Latin square* [25]. Participants complete a set for each condition.

In total we had 18 participants $\times$ 12 trials $\times$ 3 conditions = 648 trials. Each of which required multiple parameter selection and adjustment sub-tasks.

Each new condition is marked by a tutorial screen. The experimenter accompanies this with a short demonstration of the interaction technique. Participants are allowed to practice with each new condition at the start of each trial set until s/he

is ready to begin the set. Participants are encouraged to rest between trial sets.

We log the movement of each finger for each trial of all study conditions. To avoid errors resulting from the user's perceived level of accuracy, a fixed accuracy measure is used for all trials. The participant is notified with a mark next to each parameter, when it is within range of the target value. Once all the parameters are within range, the next trial begins.

3.5.1 Hypotheses

H1. Multi-finger manipulations will be faster than the mouse condition for color selection.

H2. The efficiency of the multi-finger conditions (C1-C2) will be comparable to that of the mouse (C3).

We verified H1 by comparing completion times for trials against the mouse condition. Since we use a fixed accuracy measure for all trials, the traditional metric of error is not applicable. To test H2, we use Zhai and Milgram's measure of *inefficiency* [20], defined as $(d - s)/s$ where d is the distance traversed by the user through the parameter space, and s is the shortest path from initial position in the parameter space to the target. This measures the level of control the user exhibits during the task.

3.5.2 Results

Each trial is performed by all participants with all three conditions. Figure 3.6 shows the task completion times for the three conditions. For the following analysis, we normalize the time taken for each trial, with the equivalent time taken to perform the same trial by that participant with the mouse.

The following results are produced using a Welch two sample t-test. Participants

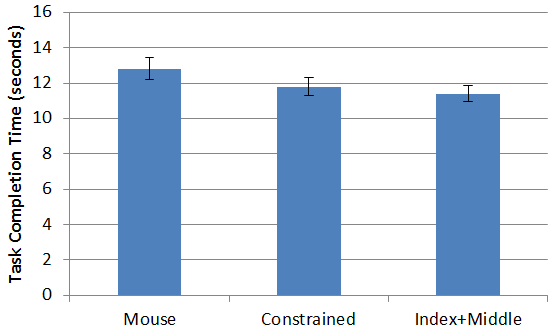

Figure 3.6: Time for multi-touch interaction technique conditions; error bars show 95% confidence.

performed 9.25% faster with the constrained multi-finger condition (C1) than with the mouse condition. The result was statistically significant ($t = 2.951, df = 17, p < 0.005$). They performed even faster (11.12%) with the index+middle finger condition (C2) than with mouse, and again this was significant ($t = 2.71, df = 17, p < 0.008$). The results validate our first hypothesis H1.

We also found from the results that the efficiencies of the multi-finger conditions (C1, C2) were not significantly different from the mouse condition ($p = 0.48, 0.43$ respectively), proving our second hypothesis (H2).

3.6 Design principles

Through this evaluation, we have noted that the participants are able to quickly learn new mappings. The movements we selected are designed to operate on the visual feedback, aiding the user to form a mental model that correlates hand movement with the operation on screen. The mappings outperformed mouse based selection. This implies that indirect touch interaction that does not require visual targeting of on-screen elements have better performance than even mouse based interaction.

4. MULTI-TAP SLIDERS[1]

Touch screens have recently become the most ubiquitous new interaction modality after the keyboard and mouse, due to adoption on small to medium size screens. When attempting to develop a touch interaction technique, there is a temptation to follow preconceived notions that the interaction must be *as intuitive* as possible. We present an argument of the limited applicability of this notion. For certain contexts it is necessary to break away from optimizing the intuition of an interaction. We present in this section an engaging new interaction experience with minimal training.

An interaction is considered intuitive when utilizes the users inherent knowledge and skills from prior experience. Such as the ability to operate real world objects: moving a piece of paper on a table is as simple as pressing on it and sliding it across. On screen spatial operations benefit the most from this sense of intuition.

Traditional interface elements have been designed to leverage this transfer of knowledge from the real-world to digital interactions. Buttons have shadows that are removed when pressed, giving a three dimensional feel like buttons do in the real world. Check boxes come from paper forms, and even the radio box has its origins in the preset buttons of a radio. Pressing on station would deactivate the button for the other station. The *skeomorphism* design approach [22] mimics real world objects for digital interfaces. It has a large benefit during the training phase, when the lay user attempts to figure out how to operate on the interface, without prior knowledge. However, we see that after decades of use, the origins may be completely unknown to the end user. A generation of users have never seen how the buttons on an older

[1]Part of the data reported in this chapter is reprinted with permission from "Multi-tap sliders: advancing touch interaction for parameter adjustment" by Sashikanth Damaraju, Jinsil Hwaryoung Seo, Tracy Hammond and Andruid Kerne, Proceedings of the International Conference on Intelligent User Interfaces, 445-452, © ACM, 2013.

radio work. They however, do know that only one item can be selected from a set of radio buttons. These interactions are the de-facto accepted standard for computing simply due to their ubiquity.

In this research, we first emphasize the interaction [9]. The visual interface is built to support this interaction. The interface serves to build the mental model in the user while providing feedback of the exact state of the interaction at all times. Due to the cognitive complexity of the creative operation being performed, the interaction should allow the user to focus more on the target of the operation, rather than the interface to do so. For example, the regular sliders are usually placed on one edge of the screen, requiring visual and motor coordination to target each parameter slider thumb to begin operation.

4.1 Designing the interaction

The human hand has a natural resting position, with the palm and fingers slightly curved inwards. This is similar to the position with which we hold the mouse. The index and thumb fingers together have the most degrees of freedom and control in daily operation.

The summarized goals for our interaction design are as follows:

- Minimize the requirement of targeting visual elements like the slider thumb.
- Palm support: The interaction shouldnt require moving the arm.
- Precise control using the index or thumb.
- Merge command selection and manipulation to make the operation integral.

We limit adjustment of the parameter to the index finger, while the middle, ring and little fingers perform the parameter selection. The index finger moves on the

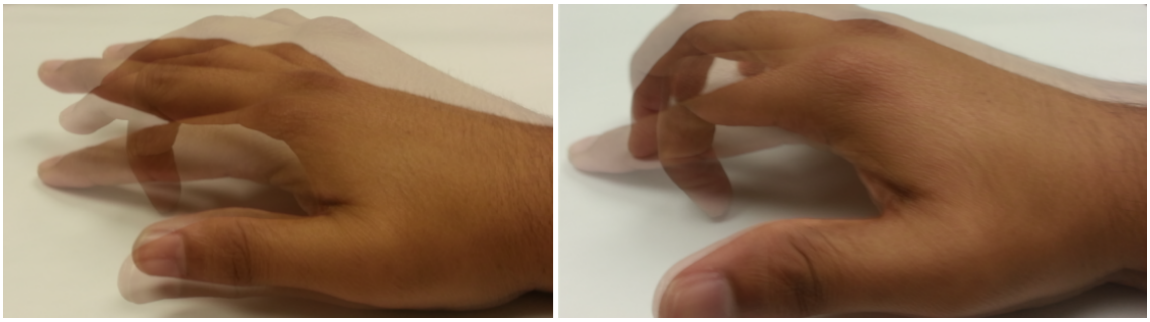

Figure 4.1: Postures of hand demonstrating the comfortable ranges of movements of the index finger on a flat surface, with the palm rested. On the left is with only the index making contact with the screen, while on the left is the same movement shown with the middle, ring, and little fingers also making contact.

screen within the comfortable limits of flexion and extension, with a total movement of about two inches. Shown in Figure 4.1 is this limited range of movement. The movement of the index finger is mapped to adjustment of a parameter. We use a simple transfer function that uses the velocity of movement for faster accelerated changes.

This adjustment is relative, meaning that the absolute position of the finger does not correlate to a specific value. Only changes are mapped. This is different from regular sliders that are mapped absolutely. The thumb of the slider denotes the current position of the slider with respect to the acceptable range.

We then use the middle, ring and little fingers to operate the parameter selection (See Figure 4.2). Unlike the related work that uses chords for maximizing the number of operations possible, we simplify the interaction by only rely on the *number of fingers* that are making contact with the screen. With no fingers making contact, the index finger adjusts the first parameter.

With one of the other three fingers down, the second parameter is selected, and subsequent movements of the index finger are mapped to adjustments in the second

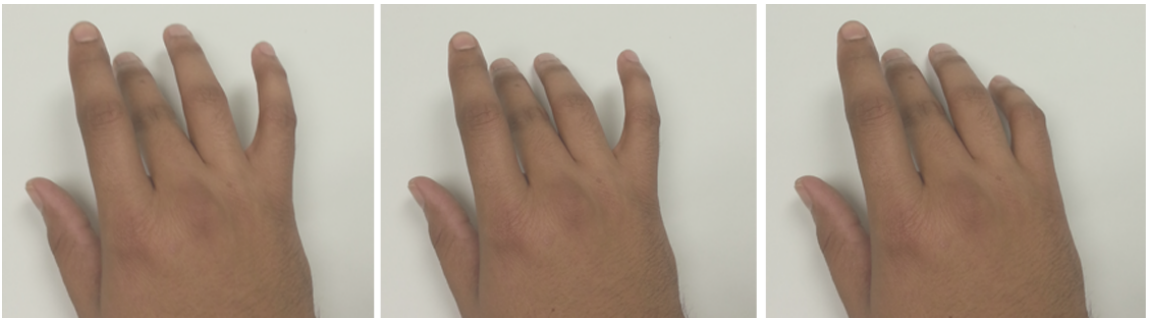

Figure 4.2: This series of images shows the movement of the middle, ring, and little fingers to consecutively make contact with the screen.

parameter and so on. With all three fingers down, the index finger will adjust the fourth and final parameter. We hypothesize that such an interaction can be learnt and quickly passed into the motor memory. Altering parameter values would become second nature for someone who has performed this technique for a couple of hours. It is seen in marking menus [65] that the accessing a menu item with a stroke becomes quicker with learning.

4.2 Designing the interface

Although technically speaking, the interaction described above could be implemented without an interface; it would have a number of disadvantages. It would take away from allowing any further operations to be performed on screen. The visual interface provides a space for interaction, keeping the rest of the screen still accessible. We prefer our technique to have an indirect interaction. The interaction should be performed on an intermediate visual, and not directly on the target of the operation, like direct spatial manipulations are performed. Henceforth the combination of the interaction and the interface will be referred to as the widget.

The interface has two main components: one receives input from the index finger

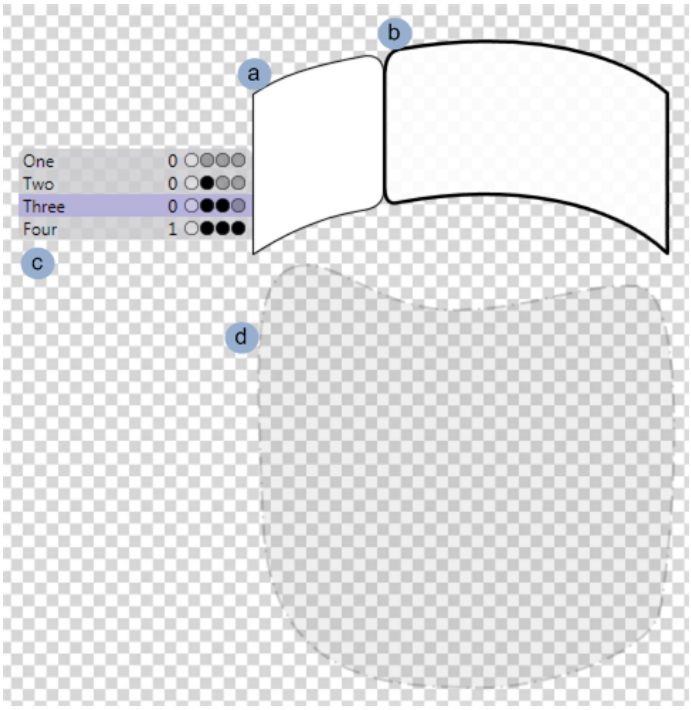

Figure 4.3: Components of the prototype: (a)*index slider* to perform adjustment of the parameter (b) *finger pad* to alter the selection of parameters (c) palm support to avoid erroneous touches, and (d) visual feedback of the currently selected parameters (parameter three shown in purple) and current values of all parameters.

(called the *index slider*), and the other from the rest of the fingers (the *finger pad*) (See Figure 4.3). The design for the prototype is straight forward, two regions side by side, with sizes to match the fingers. We also place a transparent region to allow a place for the palm to rest, so as to not confuse touch input under the palm as intentional. The resulting visual resembles a mitten, with the index finger having a different compartment. The visual has a low opacity, meaning the screen content below our interface is still visible to a certain extent.

We provide feedback on the currently selected parameter in a rectangular box placed to beside the index finger. As seen in the figure, we also place a small help

graphic beside each parameter. This graphic shows the number of fingers required to be placed on the finger pad to have that parameter selected. For the prototype, we keep the widget in a fixed location on screen.

4.3 Evaluation

4.3.1 Control conditions

We evaluate our new interaction technique with other touch interaction techniques that allow selection and adjustment of parameters. As we first introduced this design space, any existing techniques do not adequately support the task. In this section, we describe three interaction conditions: bimanual, relative sliders and traditional sliders.

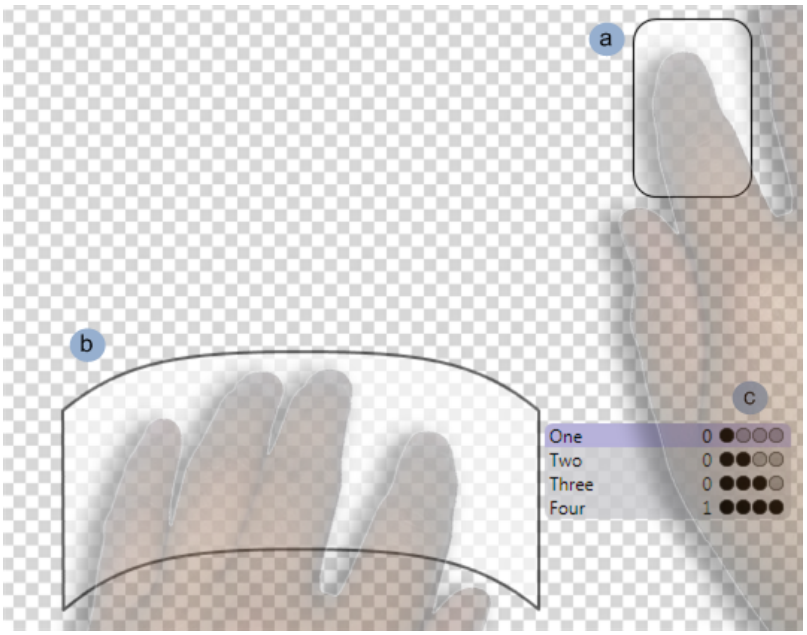

Figure 4.4: Bimanual condition containing the same parameters as seen in Figure 4.3. (a) index slider and (b) finger pad, which is now detached and placed on the other side of the screen to be used by the non-dominant hand.

4.3.1.1 Bimanual condition

We implemented an additional two handed version of the prototype to compare with. This required selection of parameters with the non-dominant hand, using the same finger-count technique as the single-handed version (See Figure 4.4). Adjustment of the parameter was performed with the index finger of the dominant hand. As all participants were right handed, this meant that the index slider was to the right of the screen, and the finger pad to the left as show here. The positions of both components was adjusted by each participant for maximum comfort before the experiment began.

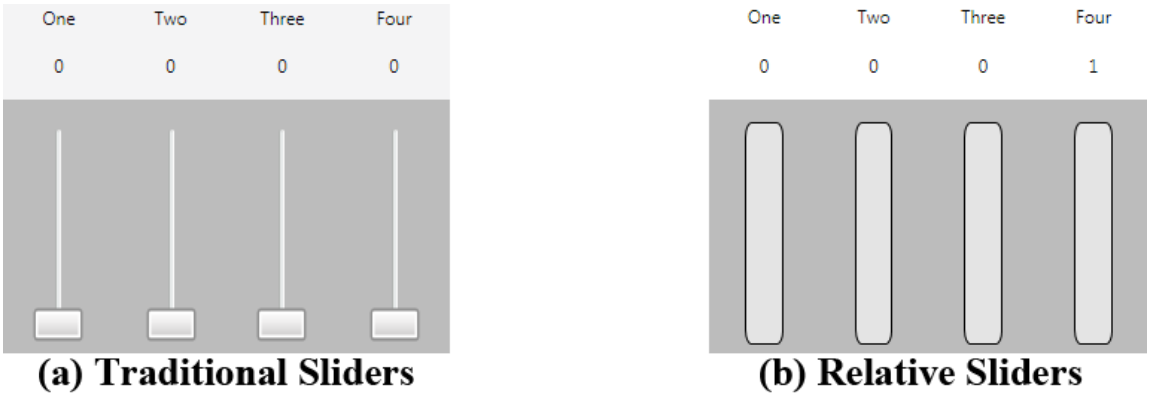

Figure 4.5: Control condition of four relative sliders. Each rectangle is similar in functionality with the index slider.

4.3.1.2 Relative sliders

The index slider is relative, meaning that the initial step of targeting the slider thumb is not required. Also, we have the added advantage of using a transfer function with acceleration that is not present in the regular slider. To make this comparison

fair, we implemented a new relative touch slider (See Figure 4.5). The relative slider functions exactly like the index slider, but does not include parameter selection. Four relative sliders were placed side-by-side vertically, one for each parameter.

4.3.1.3 Traditional sliders

As there are no special touch based slider implementations that have been developed, we placed four regular sliders arranged side-by-side vertically as the control condition. This is an unfair comparison. Operating the regular slider requires targeting the thumb of the slider first. For consistency, we disabled the feature of mouse-based sliders that causes a larger increment change when the mouse button is pressed on the bar of the slider. Users are required to first acquire the thumb of the slider before adjusting the parameter. The thumb was doubled in size to allow for easier targeting with the finger.

4.3.2 Task & procedure

We had 13 right handed participants (1 female, 12 male) perform a series of parameter selection and adjustment. For each trial, the target parameter was highlighted, and a numerical target value was presented above. The task requires participants to select the parameter, and then adjust its value to the indicated target value. Upon completion of the trial, the next target is presented.

The order of conditions was counter balanced across the participants. Each participant used all four conditions. A set of four targets (one for each parameter) was presented on screen, visualized as four rectangles with the same width, and height corresponding to the value of the parameter (See Figure 4.6).

The change in height of the rectangle as the parameter values is adjusted reinforces the correlation between direction of movement on the index slider and change in the value. Moving the index finger upward increases the value, and moving down-

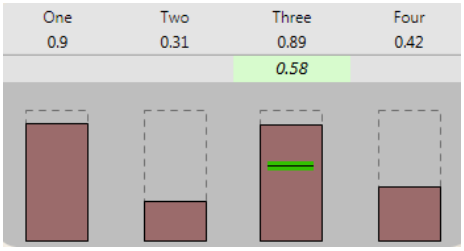

Figure 4.6: Targets for the study task. Shown here is the task to select the third parameter, and adjust its value to 0.58

ward decreases the value. The current numerical value of the parameter was displayed as a number above the rectangle.

For each trial, a target value was displayed above one of the rectangles. The user would have to then select and adjust this parameter from its current value to the target value. A short demonstration was provided to explain how the single handed and bimanual conditions work. The posture of their hands and whether or not they rested their palms on the screen was left up to the comfort of the participants. They had five minutes to familiarize themselves with the parameter selection and adjustment for each control condition.

4.3.3 Results

Although participants were concerned that the hand movements required would be inconvenient to perform, after the initial practice they felt comfortable enough to begin the trials. The relative sliders and the regular slider conditions were familiar to the participants. They had little difficulty operating them.

We did observe that during the regular and relative slider conditions, the participants did not have a consistent operation posture. Based on the last parameter adjusted and the current positioning of their fingers over the interface, they would

use the ring or middle finger to perform both the selection and adjustment of the parameters. This led to imprecise adjustment, as the precise motor control of the ring and middle finger is less than that of the index finger.

There were instances of participants switching to use the index finger after an initial adjustment was made with the other finger, because of the inconvenient posture required. This required sliding or moving the palm to a more comfortable position. Due to this, participants reported more fatigue for the regular and relative slider conditions.

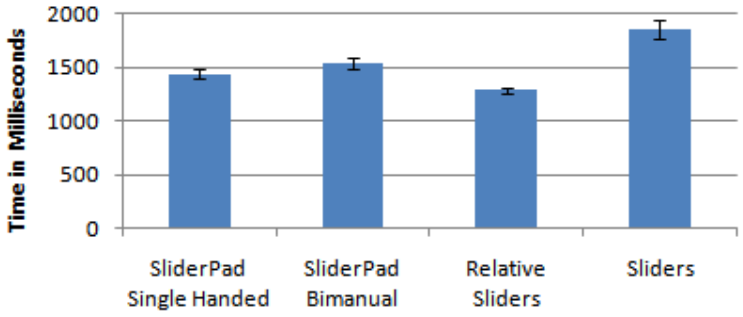

Figure 4.7: Results for selection time.

Figure 4.7 shows the mean time taken by participants between selection of parameters. A repeated-measures analysis of variance (ANOVA) showed that technique had a significant effect on parameter selection time ($F_{3,30} = 3.8, p < 0.02$). The regular slider was significantly slower at parameter selection than the other three conditions (See Figure 6. Single-handed: 1.43s, Bimanual: 1.54s, Relative: 1.28s, Regular: 1.85s). This was expected as the targeting area for the regular slider (the thumb) was smaller. There was no significant difference between the other three parameters.

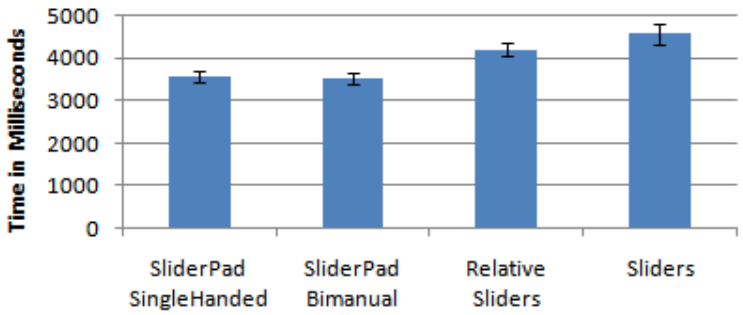

Figure 4.8: Results for completion time.

On the other hand, the time taken to complete the adjustment of the parameter for the four conditions is shown in Figure 4.8. Performing an ANOVA completion time, we found no significant difference between the four conditions ($F_{3,30} = 3.8, p < 0.02$ with Fishers Least Significant Difference at 0.72s). The completion time for the single handed and bimanual conditions is interesting. The bimanual condition can be seen as a kind of kinematic chain in which the coarse operation of parameter selection by the non-dominant hand is framing the precise value adjustment by the dominant hand. Mapping individual fingers to parameters in the multi-touch condition is a form of direct embodiment.

4.4 Design iteration

We improve on the visual and interaction design of the widget in our next iteration, using lessons learnt from the evaluation. We note that the widget occupied more screen real estate than was required for the functional operation, even though the opacity of the visual was reduced, it was still occluding content.

To reduce this occlusion by the widget, we convert the finger pad region to be a transparent *affordance* (an element that affords action [89]). We visually encompass

this invisible region with a stroke from the top of the index slider. The resultant shape resembles the top half of the yin-yang, or a brush stroke starting at the index slider and thinning away as it borders the top of the finger pad. Figure 4.9 shows the new visual.

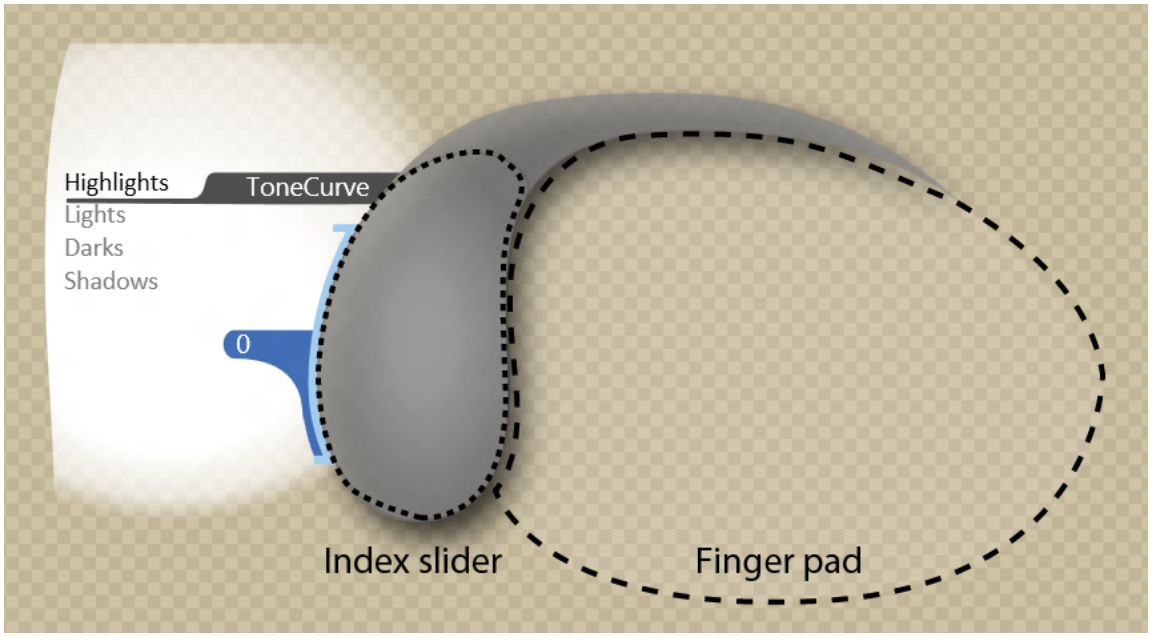

Figure 4.9: An iteration of the visual design of the multi-tap slider interface with the functional components emphasized. The *index slider* is the smaller region with dotted border on the left, while the region to the right with a dashed border is the *finger pad*.

In discussion with the participants, we noted that they felt constrained by having the widget in a single location. We introduce the functionality of translating the widget across the screen. From our design goals, we minimize the requirement to target specific visual affordances. to allow eyes-free fluid interaction. From a corollary of Fitts Law [73], we take advantage of the screen-edge concept, where the edge of the

screen is considered an easy target, since it is infinitely wide. A quick careless flick will be sufficient to target such a region.

We utilize and existing visible portion of the visual as an affordance: the region that bounds the finger pad on top. While operating the widget, using the middle finger the user can just drag up into the curved top region of the interface. This would start dragging the widget entirely, visually represented by a blue glow of the entire widget shape. During translation mode, the widget can be dragged anywhere on screen, and placed simply by raising the finger. Regular operation of the widget is halted during translation. We extend this functionality to allow scaling the widget. A second finger placed anywhere on the widget begins the familiar pinch-to-zoom operation.

4.4.1 Visual feedback

Our design approach prioritizes the interaction of the technique over the specific visual interface. However, we design the visual to be rich with state information when required. The feedback needs to be subtle so as to not distract from the target of the parameter adjustment operation. At the same time when the human looks towards this widget, we expect that they receive immediate feedback of the following:

1. Currently selected parameter
2. List of parameters available
3. Feedback of interaction: whether adjustment is under way or not
4. Current value of the selected parameter
5. Position of the current value with respect to the range of acceptable values of the parameter

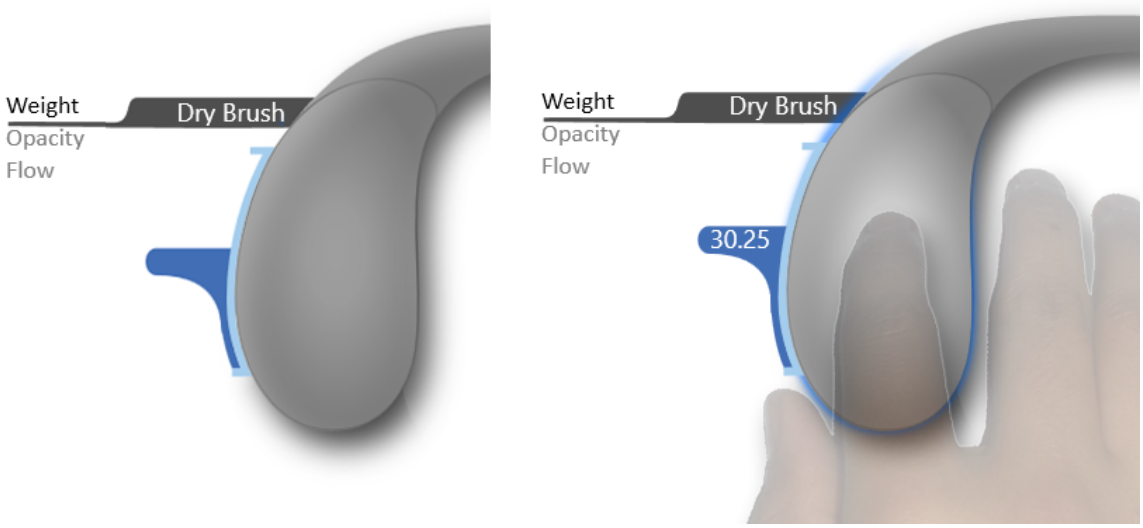

Figure 4.10: The blue outline on the right shows the feedback of parameter adjustment mode when the finger makes contact with the index slider.

Figure 4.10 shows the feedback of interaction, with a blue outline around the region of the *index pad*. The currently selected parameter (in this case *weight*) is shown in black, while the other available parameter (*opacity* and *flow*) are shown in grey.

4.4.2 Translation and scaling

We introduce the functionality of translation and scaling the widget in this iteration of design. The main criteria in designing this interaction is to maintain the ability to perform these operations as fluidly, with as little motor and cognitive effort as possible. Figures 4.11 and 4.12 show details of the operation, we well as the visual feedback performed. These new operations require a mode switch, so as to not interfere with the parameter adjustment and selection operations.

As shown in Figure 4.11, the top of the widget serves as an affordance for the

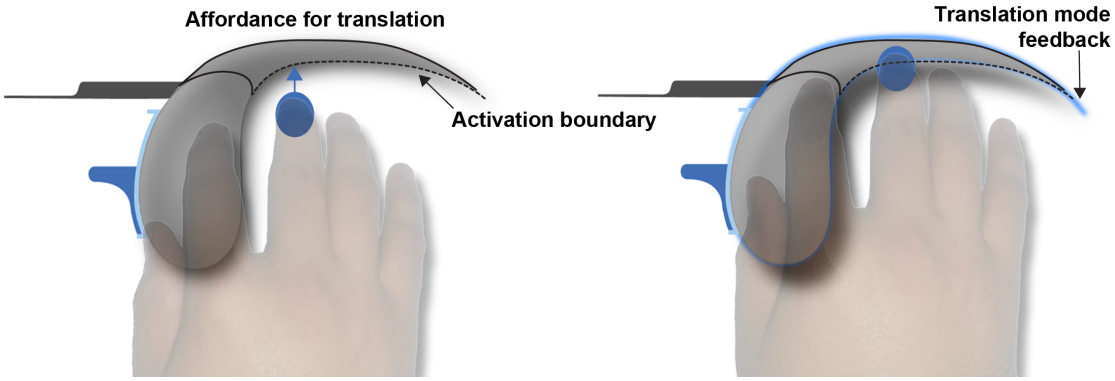

Figure 4.11: The activation boundary serves as an infinitely large target for the middle, ring, or little fingers. Dragging from the finger pad into the boundary changes the widget into translation mode, made visible by the blue outline.

translation. This interaction does not require the user to specifically target this thin portion of the widget. Instead, we *capture* the motion of the finger as it moves up from the finger pad. The moment that the touch enters the affordance, subsequent movement of the finger in any direction translate the widget to the new position. Since the arch of the widget encloses the finger pad on the top boundary, the *target* becomes easy to enter. Entering translation mode can be performed without much effort.

The widget is designed to fit comfortably under the relaxed position of the human hand. Since each individual hand varies in size, we require that the customization of the widget be as simple as possible. The translation mode requires a movement of the hand with the palm raised. When in this mode, moving a second finger performs the scaling. Figure 4.12 shows two different styles of scaling. On the left, the thumb moves in and out to make the widget smaller or larger respectively. On the right, the combined movements of the index and middle fingers scales the widget.

Raising all fingers off the surface of the screen completes the operation of trans-

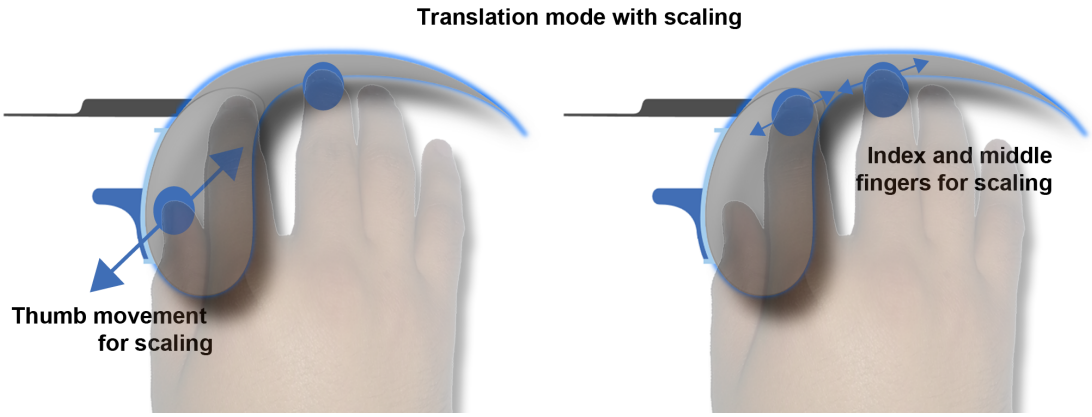

Figure 4.12: Scaling the widget. Once in translation mode, scaling is performed by movement of a second finger anywhere over the widget using the pinch gesture.

lation and scaling.

4.5 Working with existing applications

We develop a framework that enables the operation of a standard user interface through the multi-tap sliders. This is performed through the use of the mouse to operate existing interfaces, while touch operates on the multi-tap slidesr. We use the open source library Sikuli [105], a product of the research from Yeh et al. [126]. The sikuli library uses computer vision to find specific labels and items on screen. We integrate the operation of the multi-tap slider with the existing interface in an application. Figure 4.13 shows a screenshot of the multi-tap sliders being used along with the Adobe Lightroom [45] application. Selection of parameters on the multi-tap slider moves the mouse cursor to the corresponding slider on the traditional interface. Similarly, adjustment on the multi-tap slider moves the slider by the appropriate amount. XML messages are used to communicate between the Sikuli library and our multi-tap sliders.

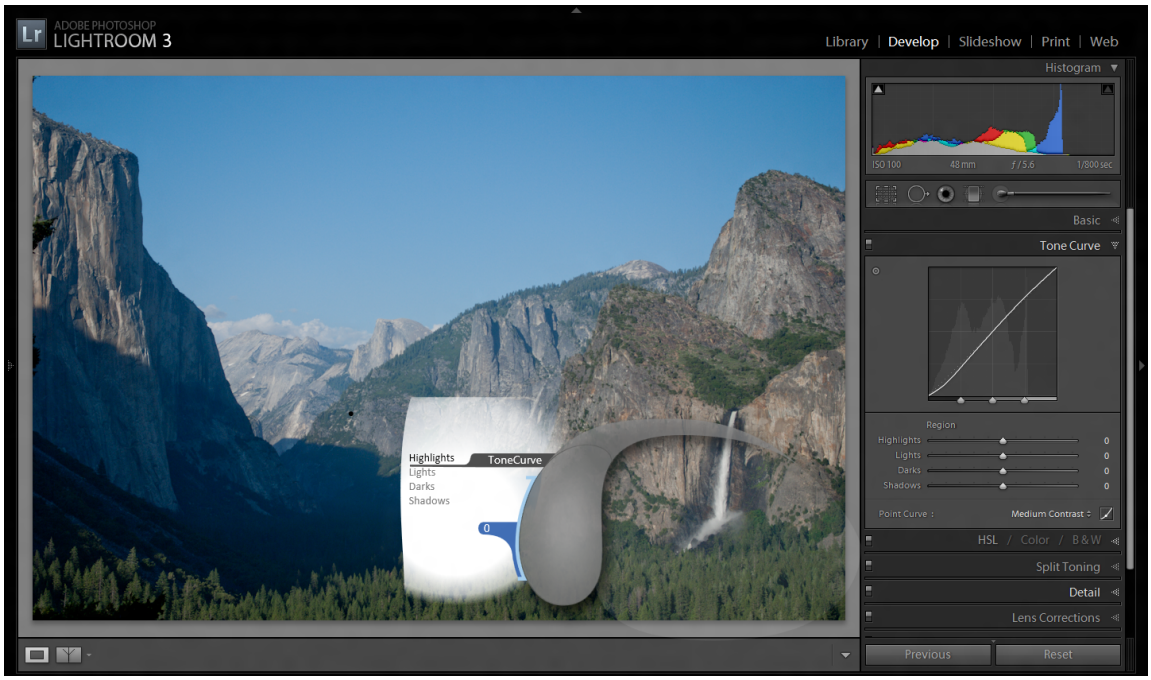

Figure 4.13: Integrated view of the widget with Adobe Lightroom [45]. The widget shows the Tone Curve operation being performed with four parameters, the corresponding parameters on the interface are seen on the right panel.

4.6 Design principles

We introduce a new space of design: non-spatial parameter manipulation. The main contribution of this work is the fluid transition between the selection and adjustment of parameters. We note that traditional menus and control panels fail to properly support the users mental model, the perceived integrality of the parameters is under-utilized in the interaction.

In this section, we also emphasize the success of the design process. We start not with the visual interface, but by first thinking about the movement of the hand. The interface plays a support role in this experience. Such a design process broadens the scope of future interaction techniques. New forms of interaction require new

metaphors. The constraints of previous input modalities need not apply to new ones. Rethinking interactions in a human centered process allows for innovative and engaging experiences.

4.7 Section summary

Multi-tap sliders represent a novel approach to the multi-touch interaction design space. Most prior work in touch can loosely be categorized into the broader categories of spatial manipulation and gestural interaction. Both these categories have a limited applicability to complex operations. Our research is focused on making creative tasks fluid. Specifically, we design techniques for operations with multiple parameters that require constant tweaking.

We present a process that begins with developing an interaction, followed by designing an interface that supports this interaction. This distinction allows us as interaction designers to discover movements that are first and foremost comfortable to perform, over multiple repetitions, with the least movement throughout the operation. We have shown that such a process can lead to a widget with fluid interaction and a clean aesthetic.

This research explores the design space of creating new paradigms of interaction. We show how a new paradigm can combine precision with comfort. Freeing ourselves from the literal graphical elements, we think in the abstract about mappings between movement of the body and digital manipulation. During our evaluation, we find that the distinguishing factor affecting parameter adjustment time is whether the sliders are relative or absolute.

Relative sliders have an advantage of having a larger target area, not the small thumb of the slider improves selection time. This means the users avoid one targeting operation altogether. We show our design for parameter selection could improve over

this relative slider. The index finger is consistently used for adjustment regardless of which parameter is selected, making the hand movement more consistent over the entire operation.

5. MULTI-TAP SLIDERS FOR DATA ANALYSIS

The previous section describes the approach to the design of the multi-tap slider with the example domain of photo post-processing. In this section, we consider an application of the multi-tap sliders to navigate through another non-spatial multi-dimensional parametric space. As in the case of photgraphic post-processing, there is a significant portion of data analysis that is performed by humans through visualizations of the data. Since the data itself could come in any form, a spreadsheet, a database or a sequence of text, spatial manipulation techniques are rarely applicable for such a context.

5.1 Myspace data set

We consider a scenario in which a researcher is given the task of analyzing data from the Myspace social network [87]. This example used $300,000$ posts that were scraped from the web between a period of two years between 2006 and 2008. The researcher is interested in investigating the possibility of determining cyber bullies and predators through this data set. The data set consists of the following parameters that are associated with each post:

- AuthorID*
- Name*
- Age
- Gender
- City
- Region
- Country
- RelationshipStatus

- HereFor
- SexualOrientation
- Hometown
- Height
- BodyType
- Ethnicity
- Religion
- Smoke
- Drink
- Occupation
- Children
- Education
- Income

*AuthorID and Name are the only two mandatory fields. The rest of the fields are optional and self reported. We filtered the posts to only consider those whose authors have reported their age and gender, resulting in a total of 105,000 posts.

We further associate each post with the following series of calculated parameters

- all caps words: number of words in a post with all upper case
- characters: total number of characters in the post
- commas
- common words: number of words from a pre-selected list of common words
- uncommon words
- egocentric pronouns: I, my, mine, me
- male words
- female words

- ellipses: Usage of ...
- emoticons
- nosed emoticons
- noseless emoticons
- exclamation points
- hedge words
- hyperlinks
- misspelled words
- netspeak words: abbreviations commonly used on the internet chat (eg: lol for laugh out loud)
- non alphabetic words
- repeated letter words: (eg: hiii)
- sentence case words
- slang words
- swear words
- symbol characters

These new calculated parameters provide a multi-dimensional quantitative space that is difficult to visualize completely. However, the researcher is primarily interested in looking at correlations between her calculated parameters and the self-reported age and gender of the author. The researcher has a hunch that perpetrators of cyber bullying and online predation may be identifiable by certain textual patterns in their posts.

5.2 Visualization

The size of the data set makes it infeasible for a single person to identify patterns by looking at a spreadsheet of this data. Taking it one step further, it is also unlikely

to understand what the patterns are without first navigating through the multi-dimensional space to understand correlations between the various dimensions. We use the Tableau Public [51] software to generate the figures in this section, while the multi-tap slider interacts using the Sikuli API as in Section 4.5. Fig 5.1 shows the distinct number of authors on the Y-axis and age of authors on the X-axis, split vertically by gender. From this alone, there isn't much that can be surmised apart from the age distribution. All we can see is that a majority of the authors are in their early twenties, with females only slightly outnumbering male authors.

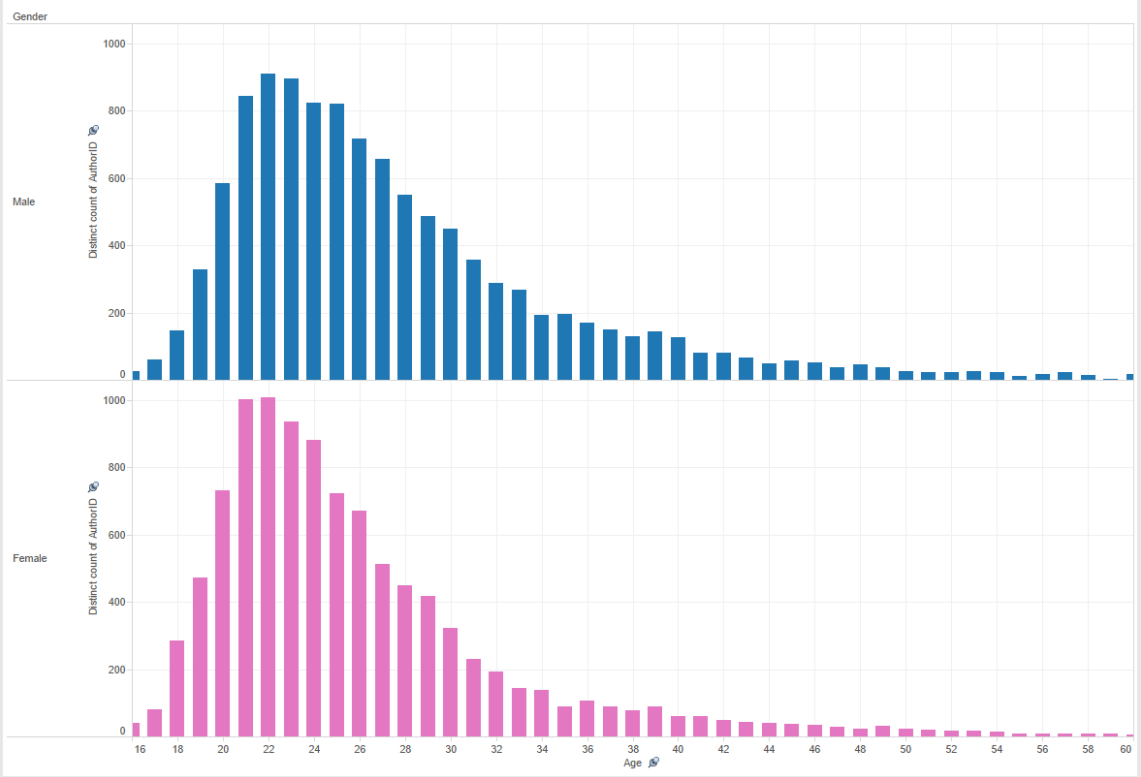

Figure 5.1: Visualization of the number of authors of Myspace posts in our data set, classified by age and gender.

5.2.1 *Interaction - data filters*

In Fig 5.2, we consider a small subset of our calculated features. The researcher is curious about the correlation of these parameters, primarily considering the impact on the distribution of age and gender seen in Fig 5.1. What she cares about is not the specific values of the individual parameters in Fig 5.2, but what interesting patterns are discovered in the Age-Gender distribution.

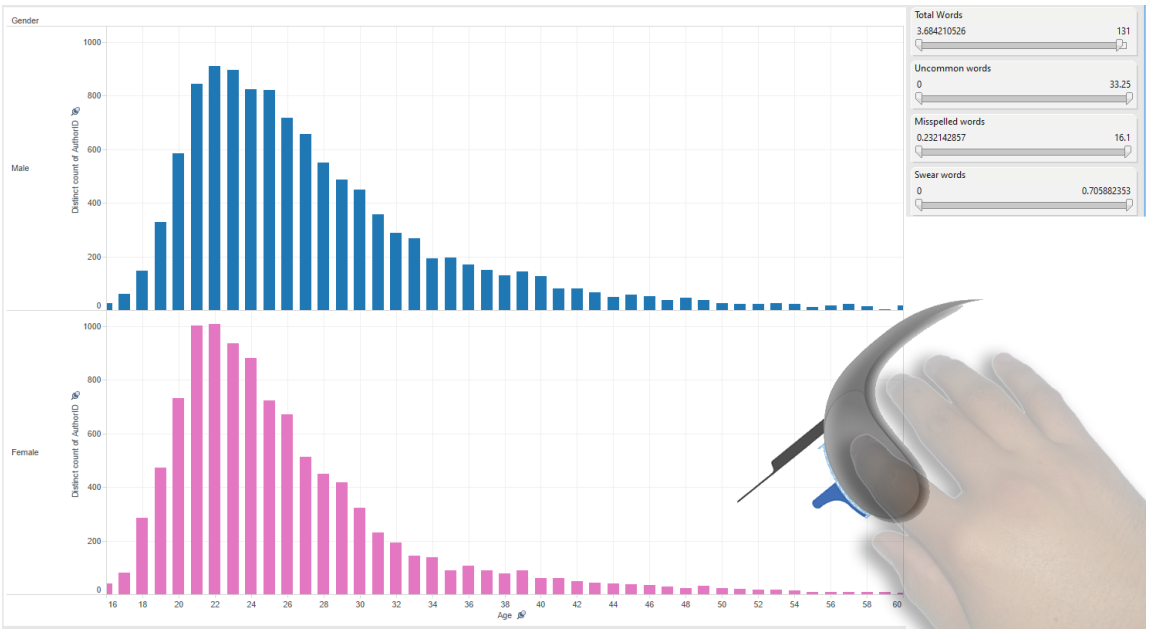

Figure 5.2: On the top right is a subset of calculated parameters derived from a pre-selected dictionary list. We integrate this subset of parameters with our multi-tap slider. For legibility, subsequent figures will not show the multi-tap slider.

The sliders allow a quick filtration of the large data set based on specific parameters. The subset of four parameters seen in the figure are considered the input parameter space, while the age-gender distribution on the left is the visualized output. As a single slider is moved, the visualization on the right is updated to show

only those authors that meet the criteria of the parameter sliders. In Fig 5.3 we see only the authors that meet the criteria of having at a minimum of three misspellings in their posts.

This configuration is similar to our context of photo-processing, wherein the user is more interested in a specific visual, and not the literal target of the interaction. The similarities between Figures 5.2 and 4.13 are clearly visible. We can clearly see that the visual focus of the interaction is on the graph rather than the specific values of the slider.

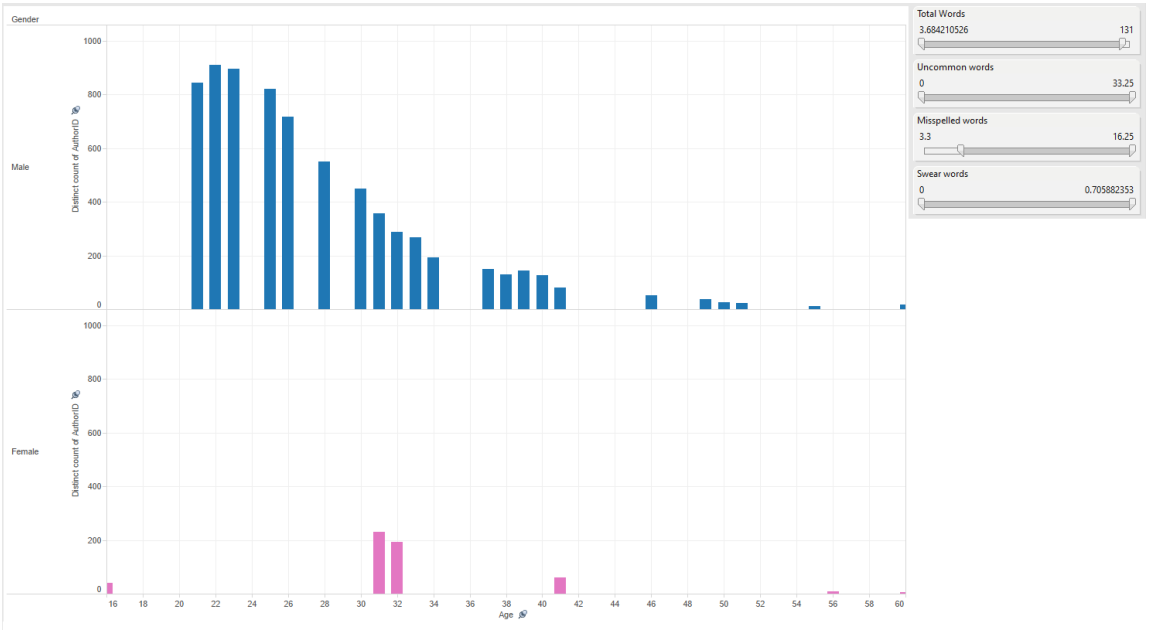

Figure 5.3: The multi-tap slider is used to manipulate the third parameter: Misspelled words, while the visualization on the left changes to reflect the authors with a minimum of 3 misspelled words in each post. We find that more male authors are likely to misspell words than their female counterparts.

The researcher finds an interesting correlation in Fig 5.3 by playing with the slid-

ers. She finds that most misspellings are made by male rather than female authors. Findings such as these can be attributed to the ease of pursuing hunches. If the software requires a series of steps to investigate such intuitions about the data set, the probability of a user taking such steps diminishes exponentially.

With the multi-tap sliders, the physical and cognitive effort of navigating the four dimensional input space seen in Fig 5.3 is qualitatively lesser than that of the mouse. The visual and motor targetting of the slider on the right is completely replaced by the simple act of placing fingers down on the screen. This makes the interaction less streneous, and hence the user is more likely to experiment with various approaches or hunches she may have about the data set.

5.3 Pattern finding scenario

In the following three figures, we show correlations between two parameters in yet another subset. Here we consider the following subset of parameters:

- Egocentric pronouns
- Female words
- Male words
- Netspeak words

The figures reveal an interesting narrative. In Fig 5.4 we see that more women use egocentric pronouns than men. And as expected, Fig 5.5 shows that female words are used more frequently by the female gender. However, when it comes to using technology savvy net-speak words, we see in Fig 5.6 that women are more likely to adopt the new language than men in their late twenties. Such an exploration of patterns was made possible by the fluid ability to navigate through a complex

multi-dimensional space with a pre-defined, context-agnostic interaction technique: the multi-tap sliders.

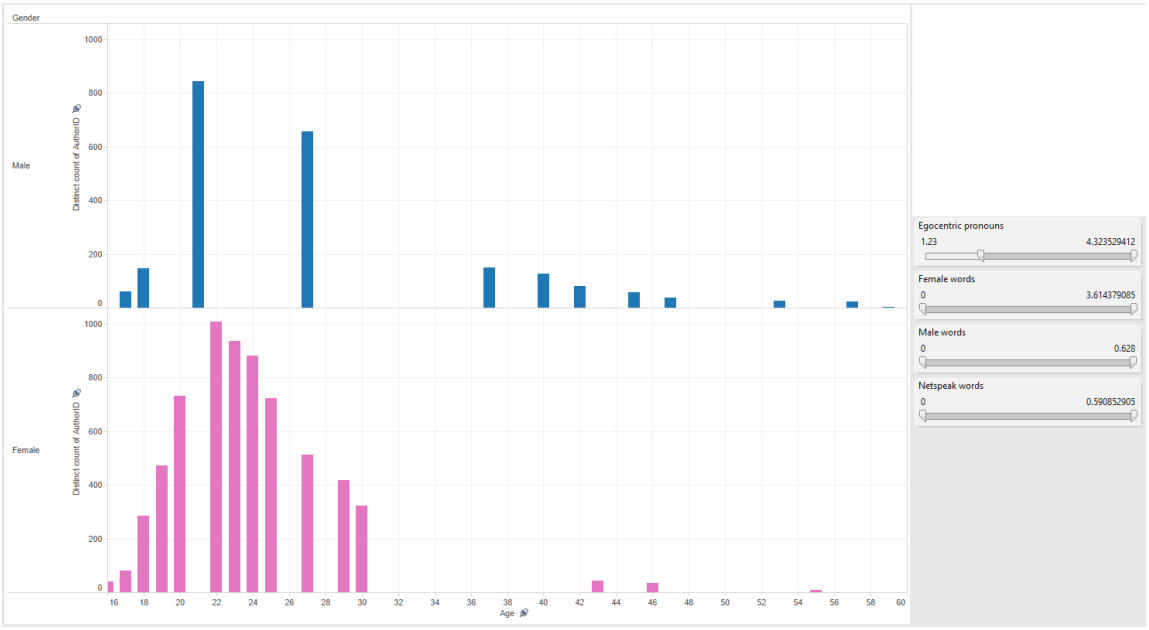

Figure 5.4: Adjusting the first slider on the right shows the distribution of authors that use at least a few ego centric words. As we move the slider to the right, we find that more woman use such words than men do. This is a dynamic process, the revelation isn't evident at a single value. During the process of adjustment we see the pattern emerge.

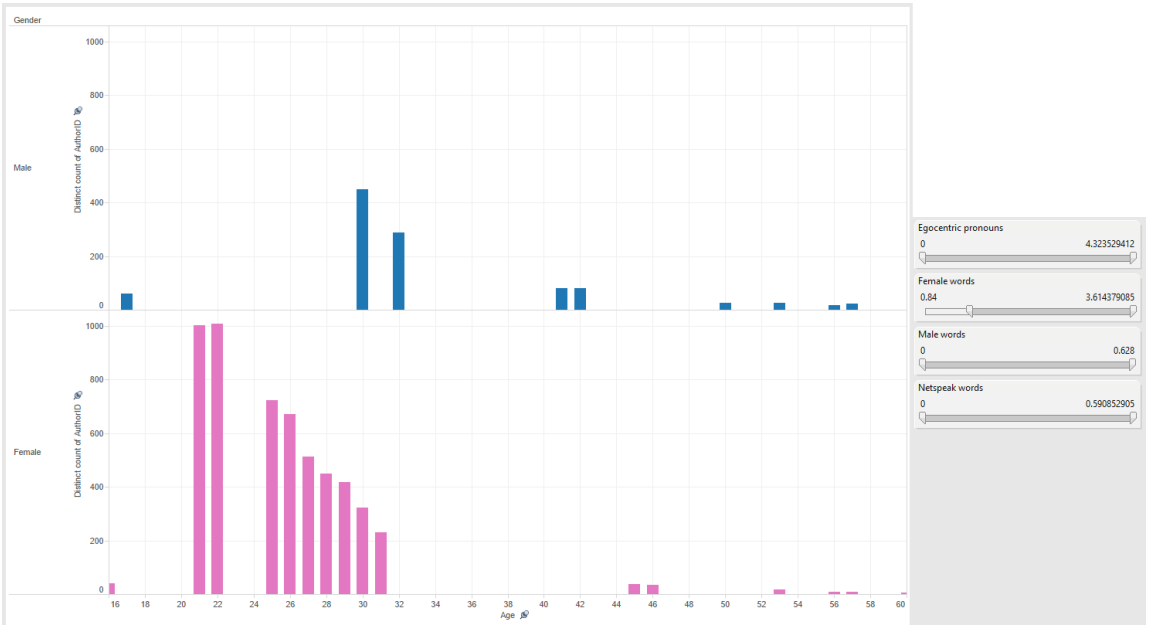

Figure 5.5: The third slider reveals the expected distribution that shows that female authors are more likely to use female words.

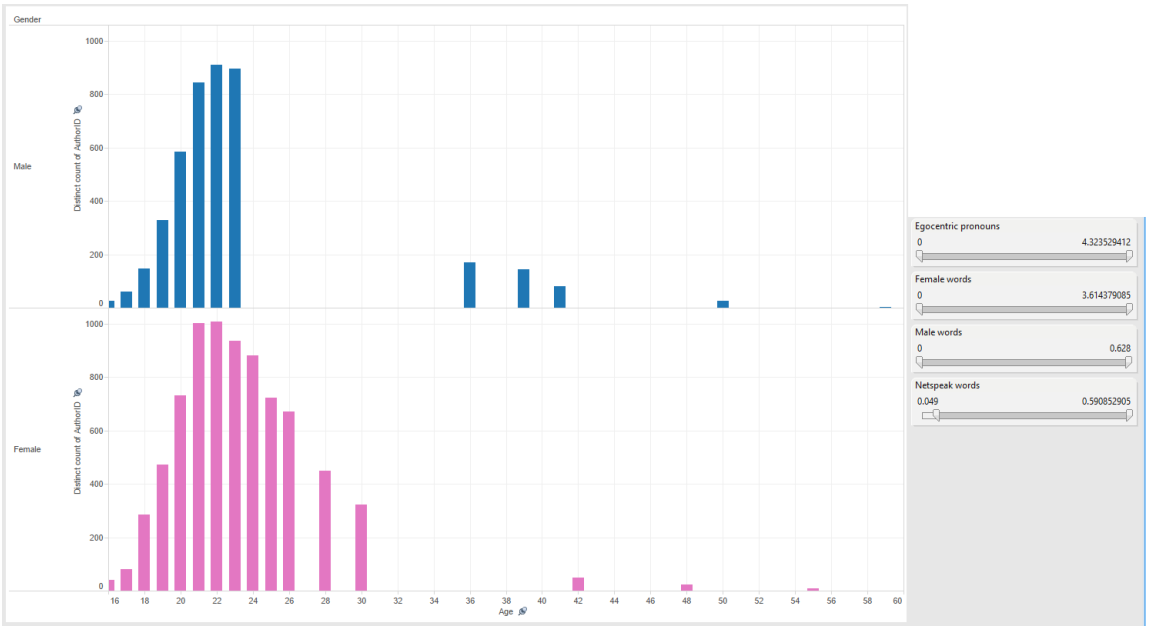

Figure 5.6: On increasing the fourth slider, we find an interesting result. Women in their twenties are more likely than their male counterparts to use technology savvy jargon in their posts.

From this scenario of data analysis, we see the benefit of using the multi-tap sliders beyond the domain of photographic post-processing. Non-spatial parameter spaces are prevalent in a large number of domains, it may take a non-trivial amount of effort to transform the space into something that can easily integrate with the multi-tap slider.

The choice of what parameters to select is up to the domain expert, and is crucial to the successful analysis and discovery of patterns within the data set. In the scenario above, we simplified the output visualization to only three dimensions: number of authors, age and gender. We could have instead opted to visualize more details. Fig 5.7 shows an example that reveals a visualization showing 10 different parameters. This figure increases the complexity and thus the cognitive load on the

user. There may be instances that such visualizations be unavoidable, however, care should be taken on over-burdening the user with information on screen.

Figure 5.7: A complex visualization showing the gender and age distribution of authors on several calculated parameters. Navigating a subset of parameters with such a visualization is more time consuming, as every adjustment of the input parameter would require a conscious awareness of what has changed in this visualization.

6. EXPLORING THE DESIGN SPACE OF MULTI-TAP SLIDERS

In this section, we present an exploration of the design space of multi-tap sliders. As a novel metaphor, the multi-tap sliders assign specific operations to four fingers of the hand. The index finger performs adjustment. The ring, middle and little fingers together are responsible for selection of the parameter. Here we describe different parameter types. The fundamental operation and experience of the slider remains the same, while the spaces that are being manipulated can vary.

6.1 Multi-scale navigation

The three fingers (middle, ring and little) function as mode switches. For use in multi-scale navigation, we also overload the metaphor with increasing levels of precision. As more fingers make contact with the screen, the more precise the adjustment. The semantics of parameter selection is transformed to adjusting the granularity of a single adjustment. In the context of video scrolling, the timeline is used to quickly select a time in the video. However, the precision of the selection is limited by the width of the scroll bar. The smallest movement possible is a pixel. Consider a full-length feature film of 2 hours (120 minutes), a single pixel of movement on a 600 pixel wide timeline would map to 12 seconds. For a touch screen, moving the finger by a single pixel is below the threshold of what is practically possible.

With our widget, we would map the first parameter to a 5% move in the timeline (6 minutes from our example), while the second parameter would map to a minute, third to 1 second move in the timeline, and finally the fourth parameter would map to a single frame movement. This would allow an ability to pick a specific frame in a full length feature film with ease.

While navigating a dense tree of information like the phylogenic tree that repre-

sents the ancestry of all known species, we could toggle which level of the tree we are scrolling through. This would make the act of scrolling more functional, separating it from targeting extraneous elements for switching levels. Our technique can be easily applied to dates and times.

6.1.1 Range manipulation

A range consists of a minimum and a maximum value. Interacting with a range can have three operations: altering the minimum, the maximum or for convenience move the entire range at a time. We map parameters to each of the operations. The widget can then perform increments or reductions in either of the parameters simply by moving the index finger, and placing the other fingers on the screen.

The thumb, with the index finger, have the most degrees of freedom. Either finger can be easily moved independently without causing any involuntary movements in other fingers. The comfortable range of movement of the thumb is in two directions: moving into and away from the palm, and an up-down motion. When using a touch screen, this becomes a flick towards the palm, away from it and raising the thumb off the screen. Mallik [74] demonstrates the use of the thumb, moving within the comfortable range of motion to perform the selection of menu items.

6.2 Thumb interaction

The thumb's natural range of movement is fairly limited while the hand is over a touch screen. A tiny ($\approx 20^\circ$) vertical movement determines if the thumb makes contact with the screen or not. This movement is well within the comfort zone of a human hand. In addition, a short range of horizontal movement ($\approx 15^\circ$ on either side of the natural position) is also within the comfort zone. We show how these movements can be integrated into the use of the multi-tap slider.

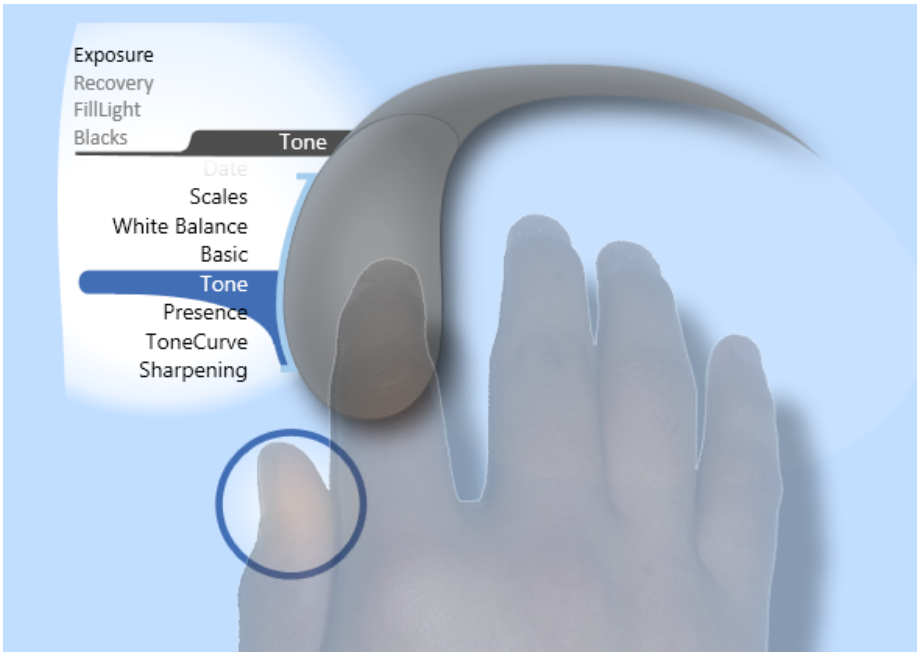

Figure 6.1: Placing the thumb onto the screen brings up the parameter menu. Scrolling with the index finger selects a set, lifting the thumb enters the parameterization menu.

6.2.1 Parameter set menu

In our design we use the thumb for two distinct interactions. For the first, touching and holding down onto the screen raising the parameter menu. This is a distinct mode, that the user maintains while the thumb makes contact with the screen. Figure 6.1 show the parameter menu. When this menu is displayed, the index finger performs the selection. Raising the finger selects the set of parameters.

6.2.2 Undo/redo

The goal of this research is to encourage exploration of multi-dimensional parameter spaces. Fine grained adjustment in the dimension of each parameter is carefully performed, while the visual focus is maintained on the target. Figure 6.2 shows how

small movements of the thumb are mapped to undo and redo operations. The precise position of the thumb region can be customized for each user. The operation can still be performed without looking at the interface. The flicks are an easy movement to perform. The inward undo movement takes less effort than the outward redo movement, as found by a study mapping the operation to back and forth in web Browsers [86].

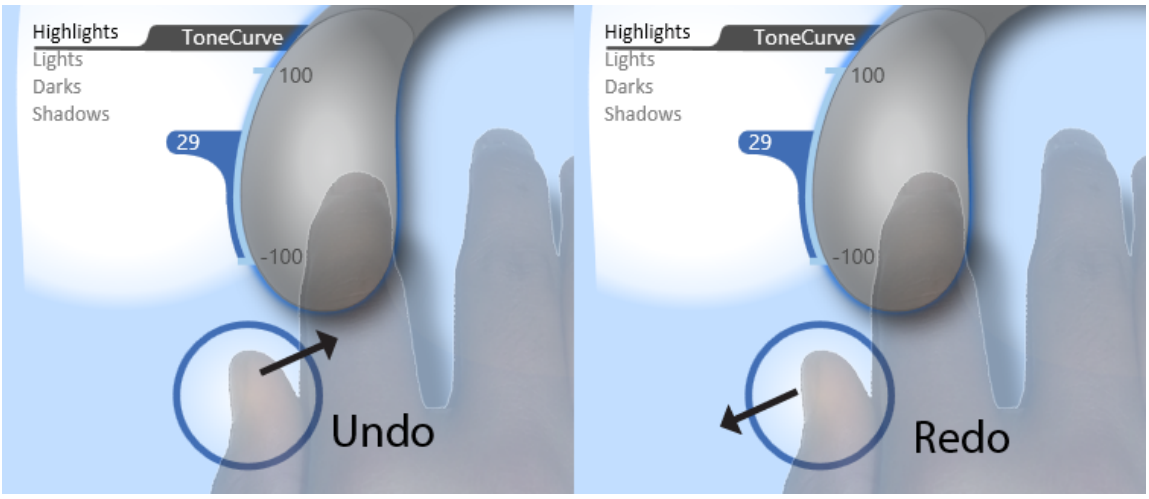

Figure 6.2: Using the thumb for undo and redo operations

6.2.3 Integrated tool parameterization

Tools in design applications are loaded with a variety of customizable parameters. The most basic brush tool in Adobe Photoshop has the following parameters: brush type, size, hardness, opacity and flow. In the toolbox, the other tools (blur, sharpen, smudge, burn, dodge and sponge to name a few) have a similar set of parameters. During regular use of the tool, designers experiment with different parameter values. This requires frequent back and forth trips from the canvas to the parameter toolbar,

placed on the top of the screen by default. Even when the toolbar is placed closer to the canvas, it requires targeting of small items. The *Springboard* technique [42] addresses the problem of selection of the tool, we presents a solution for customizing parameters of each tool.

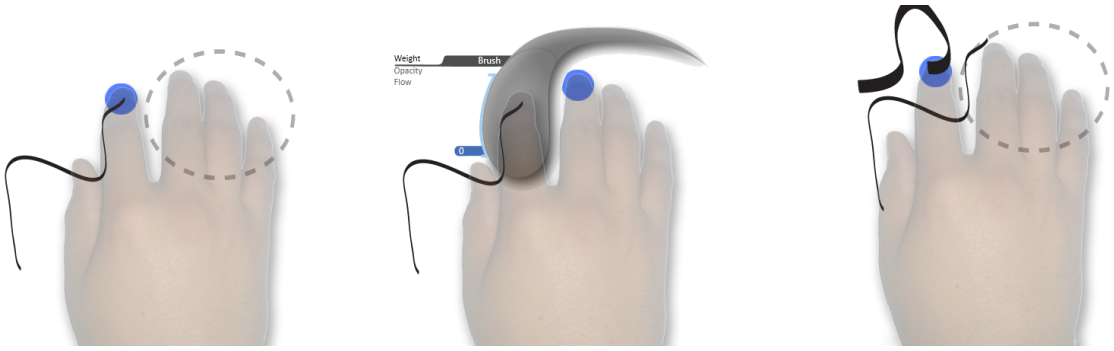

Figure 6.3: The finger pad follows the first touch point, which is shown here operating the brush tool. The dashed circle is the transparent finger pad region highlighted for visibility. Placing the middle finger on the screen raises the widget, allowing selection and adjustment of the stroke weight. Raising fingers from the finger pad makes the widget invisible, while the index finger continues operating the brush.

We modify our interaction technique to make the index finger operate as a tool brush, instead of the adjustment (see Figure 6.3). However, we make the *finger pad* follow the index finger as a transparent tracking toolbar (like the ToolGlass interface by Bier et al. [13]). The widget itself is not displayed. When required, placing a finger onto the finger pad will raise our widget. Now, the parameters of the tool can be selected and adjusted. Raising all fingers from the finger pad will make the widget disappear, reverting the index finger to operate as a tool again. This provides an in-context access to the tool parameters.

6.2.4 Bimanual control

We recognize the limitations of the slider to allow for only four parameters. There are operations that consist of more than four *integral* parameters, such as the split toning operation, which presents upwards of 8 different hues that could be altered. All eight of these hues are integral to the operation, grouping them in two groups of four breaks the mental model of the user. Switching between any of these eight parameters should be as simple as switching between the four parameters that our technique currently supports.

In cases that require more than four parameters, we use a secondary finger pad to be used with the non-dominant hand. The number of fingers placed on this secondary pad selects another four parameters within the operation, supporting a total of 16 parameters for a single integral operation. To select and adjust the sixth parameter, the user performs the following:

1. Using the non-dominant hand, place a finger on the secondary finger pad.
2. Using the dominant hand with the regular widget, place one finger on the primary finger pad.
3. The index finger of the primary hand performs the adjustment.

However, it should be noted that juggling all sixteen parameters for a single operation would be overwhelming for the user. Care must be taken when deciding how many parameters must be grouped into an operation, and when possible group other parameters into another operation.

6.3 Alternate input techniques

The input technique we developed this technique for were large ($>$ 15 inches) touch screens. The industry has been slower to adopt this form factor, due to the

lack of a strong use case that shows the strength of using touch for more complex scenarios. The most common form factors with the capability of sensing multiple touch contacts are, in order of prevalence: smartphones ($3.5 - 5$ inches), tablets / Apple iPads[47] ($7 - 11$ inches) and a few laptops ($10 - 15$ inches).

The multi-tap sliders can function as they are on smart phones. their size is sufficient to contain four fingers, including the range of movement for the index finger. By having interaction on the phones, with the visual on a monitor, users can navigate parameter spaces as they would on a large touch screen. Since the multi-tap sliders are designed to be an indirect interaction, the visual disconnect between the movement of the hand on the phone, and the changing visual on the screen does not take away much from the experience. It is possible to have additional feedback on the monitor to bridge the gap, so as to not require looking down at the phone for the feedback. For the next range of touch input devices, tablets/laptops ($7 - 15$ inches) the problem of occlusion becomes more important. With limited screen real estate, devoting a large percentage to interaction would reduce the available space for the rest of the application.

Another prevalent multi-touch input device are trackpads, most notably the ones on the Apple MacBooks [48]. These are only input devices, with no visual display. Traditionally, the trackpads are used for single directional swipes for scrolling or switching between applications. More investigation is necessary to determine if detailed movements of the fingers can be tracked so as to adapt the multi-tap sliders. It is possible for the interaction to be completely removed from the visual. If proper feedback is provided, the user could use only the trackpad to perform the selection and adjustment of parameters.

6.4 Section summary

The motivation of the design for multi-tap sliders was to provide fluid access to non-spatial parameter spaces. The focus of the interaction was on continuous numerical parameters. Based on the success of multi-tap slider over traditional sliders, we began the exploration of the space of parameter types that would benefit from the merged interaction of selection and adjustment.

The emphasis here is on making complex spaces accessible. By creating a new standard for interacting with multi-dimensional non-spatial parameter spaces, we increase the scope of the user to re-use the skill of using the multi-tap slider. Using this technique allows researchers and software developers of various domains to expose more functionality to the user. Functionality that was previously withheld to reduce the complexity of the application.

7. MULTITOUCH GESTURE TRAINING AND RECOGNITION[1]

The typical gestures used in spatial manipulation are designed during the development of the application. The recognition of such gestures are programmed into the code of the application at compile time. The operation is tightly integrated into the visual feedback. The operation begins when fingers make contact with a visual target. A quick feedback loop to makes the experience of direct spatial manipulation more natural. For this research, we aim to expand the scope of gestures beyond spatial manipulation.

Single touch *gestures* are similar to pen strokes. The received input for the system in both cases is a sequence of point coordinates. There is a rich history of stroke recognition algorithms for stylus based gestural interaction [41, 17, 110]. Simultaneously tracking multiple points of contact present a different set of challenges. We bring up issues with recognition of multiple simultaneous finger interaction.

We begin by first attempting to expand the scope of the gestures used for multitouch interaction. To do so, our approach first develops a recognition technique for arbitrarily designed sets of gestures. The goal of this stage of research is to develop a library that can reliably learn new gestures by example [99], and subsequently be trivially used by an application.

At the onset of this approach, we were fabricating a multi-touch input device that would be capable of sensing pressure, width, height and even shape of each finger input. A prototype were constructed that could successfully sense these features from the human hand. Figure 7.1 shows the operation of the sensor, based on the

[1]Part of the data reported in this chapter is reprinted with permission from "Comparing multitouch interaction techniques for manipulation of an abstract parameter space" by Sashikanth Damaraju, and Andruid Kerne, Extended Abstracts of the IEEE Workshop on Tabletops and Interactive Serfaces, 1-3,

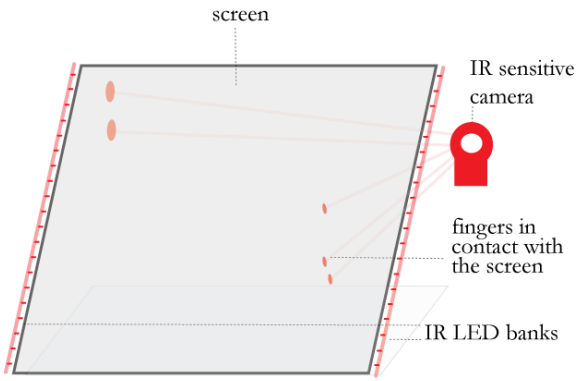

Figure 7.1: The components of a infra-red vision based multi-touch sensing system.

concept of frustrated total internal reflection [38]. Infra-red light projected into the frame of an acrylic screen is dispersed by a finger only when contact is made. This dispersed light is captured by a camera, and then processed into a set of features: X, Y coordinates of each point of contact, shape, width, height and pressure of contact.

A recognition technique we select should be able to scale to more than the basic X,Y coordinates of each finger. With these touch features, we aim to capture aesthetics of the touch. Schiphorst [101] defines a vocabulary of touch-efforts (tap, pat, hold, stroke, glide, jab, knock, slap, press, rub and knead) that describe the qualitative experience in a computationally definable form.

A software pipeline (Figure 7.2) is constructed, consisting of a) the image processing library which is responsible for the feature extraction and b) the recognition system that uses the extracted features as the basis for gesture recognition. An image processing stage, using the open source toolkit CCV [90], is first used to process the raw camera input with a series of filters that clean up the live video signal. Next, points of contact are identified and tracked, resulting in extraction of a feature set for each video frame.

Feature data is passed to the recognition system using UDP via the Tuio tangible interface I/O protocol [58], which is layered on the Open Sound Control protocol [122]. To enable our application to be cross-platform compatible we utilize the Jahmm [55] toolkit to implement hidden Markov models.

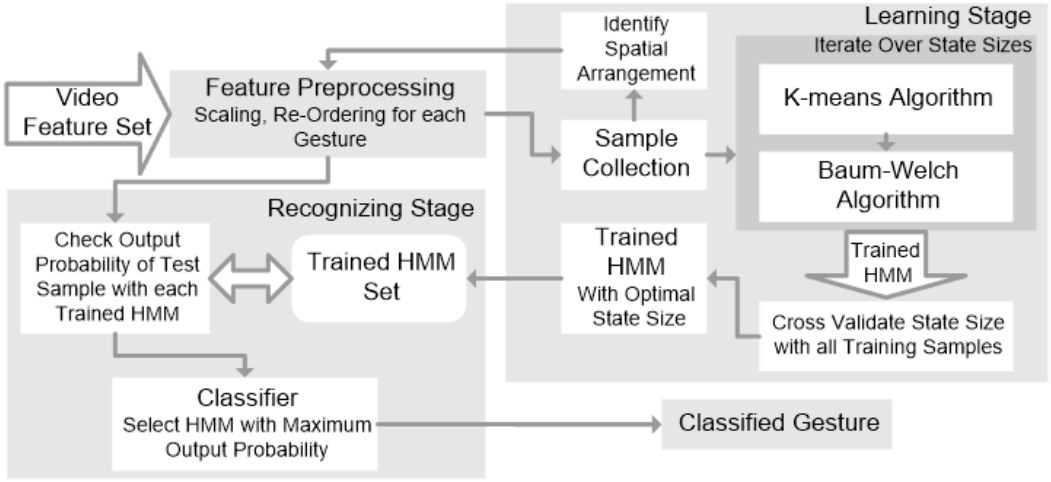

Figure 7.2: An architectural overview of the mGestr system, showing the stages of learning and recognizing a gesture.

7.1 mGestr

We describe our *m*ulti-touch *ges*ture *t*raining and *r*ecognition (mGestr) system that is designed to learn an arbitrary set of new gestures by example. Computationally, we define a gesture as a human action that begins with placing one or more fingers on the interactive surface, and ends when no fingers remain on the surface. A sequence of frames from a camera is captured to record each gesture. For each frame of the sequence, a feature set is derived by an open source video processing

and feature acquisition toolkit, and used to recognize the gesture through a pipeline of processing stages. Figure 7.2 shows the stages involved in the learning and recognition pipeline.

7.1.1 Feature preprocessing

The feature preprocessing stage normalizes the data in the X, Y feature space to allow the gesture to be recognized irrespective of where it was performed on the screen. We center the data by first calculating the mean point of contacts on the first frame of a gesture. The origin of the coordinate system for each point of contact is translated from the top left of the screen to this new mean. The samples are also scaled to a constant size for improved recognition.

7.1.1.1 Ordering fingers

The ordering of the fingers in the feature set provided by video processing is determined by the temporal order of placement of the fingers on the surface in the first frame of a gesture. Placing the index finger first followed by the thumb would produce a feature stream with each frame consisting of an array of contacts. The first element in the array represents the index finger's features and the second that of the thumb. In case the thumb is place first on the screen and then the index finger, the positions of the features in the array are reversed.

This ordering is possibly inconsistent across different samples of the same gesture. For portability and wide spread use, we constrain ourselves to hardware that does not allow for a disambiguation of fingers. Since we have no information of the posture of fingers or their positioning before hand, an inconsistent ordering is a problem for the learning and recognition system.

Unlike pen and mouse interaction, multiple points of contact are simultaneously received by the system. Since this features of this are not recognized using visual

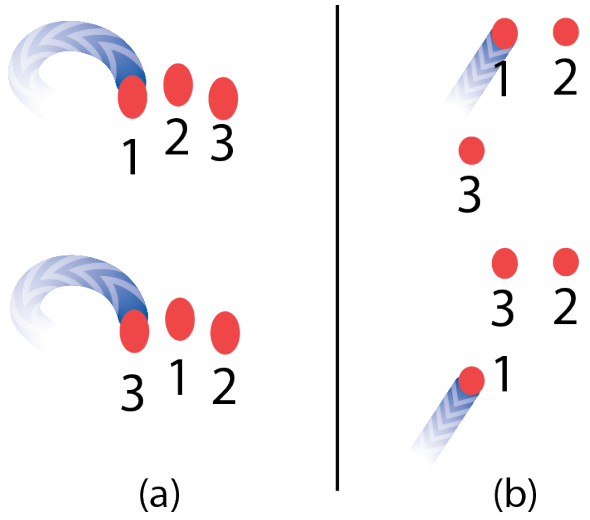

Figure 7.3: The order of placement of fingers creates noise in the recognition system. (a) The same gesture performed twice, each time a different order in which the fingers were placed on screen. (b) Two different three fingered gestures, one which requires movement of the index finger (top) and one in which the thumb is moved (bottom).

features, the raw data must be consistent across performances of the gesture. For an arbitrary gesture, it is not trivial to process the multi-finger stream of input data without first ensuring that they are in the same order that is expected of each gesture. Figure 7.3 shows how the order of placement of fingers could lead to erroneous recognition. We do not have any prior knowledge to the positioning of fingers for an arbitrary new gesture.

The spatial arrangement is identified for the first frame of each gesture sample during the learning stage. Given a set of samples for a gesture, we first calculate the distances between the fingers for the three heuristics. We pick the heuristic that yields the highest mean distance. Figure 7.4 shows the operation of three different heuristics used to order fingers. This is then stored along with the gesture model. Each frame of each sample of this gesture are then re-ordered accordingly. This results in a sample

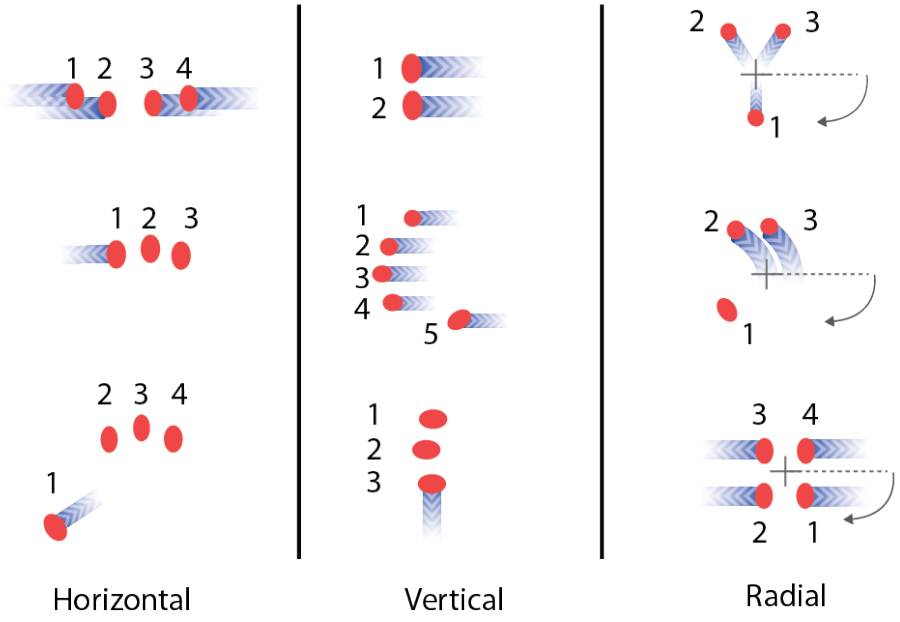

Figure 7.4: Three simple heuristics to consistently order a range of different gestures.

set of a gesture in which all samples have consistently labeled fingers, regardless of the order of finger placement on the screen. The selected heuristic is a property of the gesture, and is used later on in the recognition stages.

7.1.2 Hidden Markov models

Hidden Markov models (HMMs) are state machines, that receive input, and produce a state change along with an output. They can be used to efficiently recognize multidimensional data streams. They have been shown to be very useful for time-series data, which is what we have in our domain of multi-touch gestures. In theory, these characteristics of HMMs allow scaling from simple X, Y coordinates to more advanced features that we will extract from our new multi-touch sensing hardware. We present a summarized description of the structure and algorithms to operate a hidden Markov model below. For a more complete description refer to Rabiner [95].

An HMM is formally defined by the following:

- The number of states in the model N.
- A set of distinct observation symbols $V = \{v_1, v_2 \dots v_M\}$, denoted as M.
- The state transition probability distribution $A = \{a_{ij}\}$ where $a_{ij} = P[q_{t+1} = S_j | q_t = S_i], 1 \leq i, j \leq N$. This is a matrix of size $N \times N$, which contains the probability P of transition from one state S_i at time t to another state S_j at the next time step $t + 1$, where the subscripts i and j lie between 1 and N.
- The observation symbol probability distribution in state j, $B = b_j(k)$, where $b_j(k) = P[v_k \text{at} t | q_t = S_j], 1 \leq i \leq N$ and $1 \leq j \leq M$.
- The initial state distribution $\pi_i = P[q_1 = S_i], 1 \leq i \leq N$.

For compact notation, an HMM is denoted as $\lambda = (A, B, \pi)$. To use an HMM in a real-world application, there are three main problems that need to be solved. We state the problem as a question, and present the solution below:

1. **Training a model**: Given a sequence of observations O, each representing a gesture sample performed by a user, how can the parameters of the model (A, B and) be obtained such that the probability $P[O|\lambda]$ is maximized? Although there is no known closed-form solution for finding the parameters given a finite observation sequence, we use the Baum-Welch algorithm [8] to iteratively find the local maximum by using an initial estimate of λ. To improve the efficiency of finding this local maximum, we use the K-means algorithm to provide a better estimate of the model parameters A, B and π. These two algorithms together are used to train a model to respond with the maximum probability for a particular gesture.

2. **Evaluation**: Given an observation sequence $O = O_1, O_2, \ldots, O_T$ of multitouch features, where O_i contains the feature values extracted from a frame of raw camera input and a model λ, how do we compute the probability $P(O|\lambda)$ of the sequence from the model efficiently? This problem is required to estimate the probability of an observation sequence for a particular model. In our system, this probability decides the recognition of the gesture. The forward-backward search [95] performs an efficient computation for this problem.

3. **Optimal State Sequence**: Given an observation O and known HMM λ, how do we choose a sequence of states $Q = Q_1, Q_2 \ldots Q_T$ within the model that best fits the observation sequence? The Viterbi algorithm, based on dynamic programming [32], obtains the single best sequence Q by maximizing the criteria $P[Q|O, \lambda]$. This helps us not only to find the optimal state sequence, but also to understand more about the structure of the model. We can also obtain statistics on each individual state, such as how often it has been visited, or which states are its successors and predecessors.

We parameterize the HMMs for multitouch gesture recognition. The number of points of contact from human hand(s) on the screen varies, so does the dimensionality of the feature set extracted through image processing. For the first iteration of this research we use only the coordinates of the points of contact, the dimensionality of the data is twice the number of fingers used to perform the gesture. To model this multi-dimensional data, we use a continuous observation probability distribution for B. At each state of the model S_i, $b_i(k)$ is a multivariate Gaussian distribution that represents a multidimensional vector of values corresponding to the feature set of multitouch inputs. Each gesture is a time-based sequence of multi-dimensional points.

We first start with the training stage to create a set of HMMs for each gesture that we intend to recognize, using the method outlined in (1) above. The initial estimates for the models are obtained using the K-Means algorithm [57] as described below. Once an HMM is trained for a particular gesture, we present each input gesture as sequence of observations, and using (2), we form a probability estimate for each. For an input gesture, we then have a numerical value that represents how well it matches each of the trained gestures models. The gesture is recognized as the one whose trained HMM returned the maximum probability.

7.1.3 Learning stage

During the learning phase, after the observations have been collected and preprocessed, they are passed to the K-means clustering algorithm. The result of K-means clustering is an initial estimate of the Hidden Markov Model parameters. The estimate is, in turn, passed to the Baum-Welch algorithm, which tunes these parameters to return a high probability for assigning the given training sample sequences to a particular multitouch gesture from the vocabulary. Only gestures for which the model is trained for will return a high probability. Samples of other gestures will return lower probabilities, allowing us to use the HMMs as recognizers.

7.1.3.1 K-Means clustering algorithm

We apply the K-means clustering algorithm [57] to provide an initial estimate of the model parameters. This initial estimate is crucial in increasing the efficiency of the entire learning process. Initially, K-means begins with a random estimate of the HMM model parameters (A, B, π). For continuous observation densities, a segmental K-means procedure is used to cluster the observation vectors within each state into a set of clusters using a Euclidean measure, in which each cluster represents one of the M mixtures of the observation probability distribution. From the clustering, an

updated set of model parameters (A, B, π) is derived.

The updated model obtained from these parameters is compared to the previous model. If the model distance between the current and the previous state exceeds a threshold, then the old model is replaced by the new one, and the loop is repeated. If the distance score falls below the threshold, then model convergence is assumed and the final model parameters are saved. This algorithm is used to obtain a better than random initial estimate of the observation probability density parameter B. This initial estimate is required for the Baum-Welch Algorithm to efficiently train the HMM for the gesture segment.

7.1.3.2 Baum-Welch Algorithm

Given the good initial estimate generated by K-means clustering algorithm, we pass its output to the Baum-Welch Algorithm [8]. The Baum-Welch Algorithm uses the training set to tune the parameters (A, B, π) to obtain an optimal HMM in an iterative fashion by converging towards the local maximum, to return the highest probability gesture result for the feature observation sequence. This algorithm is an implementation of the Expectation-Maximization algorithm, where the underlying state sequence is the missing data.

The output of this algorithm is a hidden Markov model whose parameters have been optimized to return a high probability given another sample of the same gesture as input. This model is the basis for our gesture recognition system. HMMs that are trained on other gestures will return a far lower probability. By comparing these probabilities for a given input sample, we can recognize it as the gesture corresponding to the model that returns the highest probability.

7.1.3.3 State Size

The optimal HMM depends on the number of states in the model, which varies for each gesture; a simple stroke may require only two states to efficiently model it, while a complex, multi-finger gesture may require 10 or more states. To enable hands-free online learning, we train HMMs for all state sizes starting from 2 until 15. By using the training samples, we evaluate the probability of all the samples for the HMMs of each size. Each sample will select an optimal state size. We select the best state size as that which has been selected by most of the samples. This optimal HMM is then stored in the trained HMM set for sample testing.

7.1.4 Recognition stage

Each gesture has a corresponding trained HMM saved from the previous stage. When the recognition system is live, the incoming samples are sent as input to each of the trained HMMs. The probability of matching with each HMM is calculated using the forward-backward algorithm described in Section 7.1.37.1.3.2. The output probabilities from each HMM are compared. The HMM that yields the maximum probability is identified, and the input sample is recognized as the corresponding gesture associated with that HMM.

7.2 Evaluation

To evaluate this system, we first develop a set of new multi-touch gestures as seen in Figure 7.5. To build a training set, we recruited participants to perform these gestures. Ten users were recruited in total. The gesture samples chart (Figure 7.5) was presented to the participant. For each gesture, the experimenter performed a demonstration of the gesture. The participant then performed 20 samples of each gesture. In total, we collected 20 samples, for 20 gestures with 10 users giving us 4000

gesture samples across users. This set is then used to perform an off-line evaluation of the system. We present results from this evaluation.

7.2.1 Training set

Our goal is to build a practical system that uses natural movements of the hand to perform complex actions within an application. To start, we developed a vocabulary of 20 different gestures for testing the current learning and recognition system (Figure 7.5). The gestures were formed to exemplify subtle differences in initial positions of the fingers and directions of movement. They are grouped by the number of fingers used to perform each one. Two-finger gestures are only compared against other two finger gestures, the same for all other gestures.

Twenty samples of each gesture were collected from each of 10 users, 9 male and 1 female, all of whom are right-handed. We expect that the ability to recognize a vocabulary of gestures with subtle variations, such as the direction of movement of the thumb in gestures with three fingers, will be critical for developing expressive applications.

7.2.2 Results

To validate the robustness of the recognition system, 20 runs of training and testing were performed. Each run starts with a random selection of 20 training samples for each gesture from the entire set from all 10 users. These training samples are used to train HMMs for each gesture. The entire collection of samples is then tested against these trained models. The recognition labels are then verified to yield the gesture recognition accuracy as a percentage. Figure 6 shows the recognition accuracy for sets of gestures. The recognition rates for the gestures using two, three, four and five fingers are 97.3%, 95.7%, 96.1% and 91.2%, for an average gesture recognition rate of 95.06%, and the result is statistically significant ($p < .0001$). These

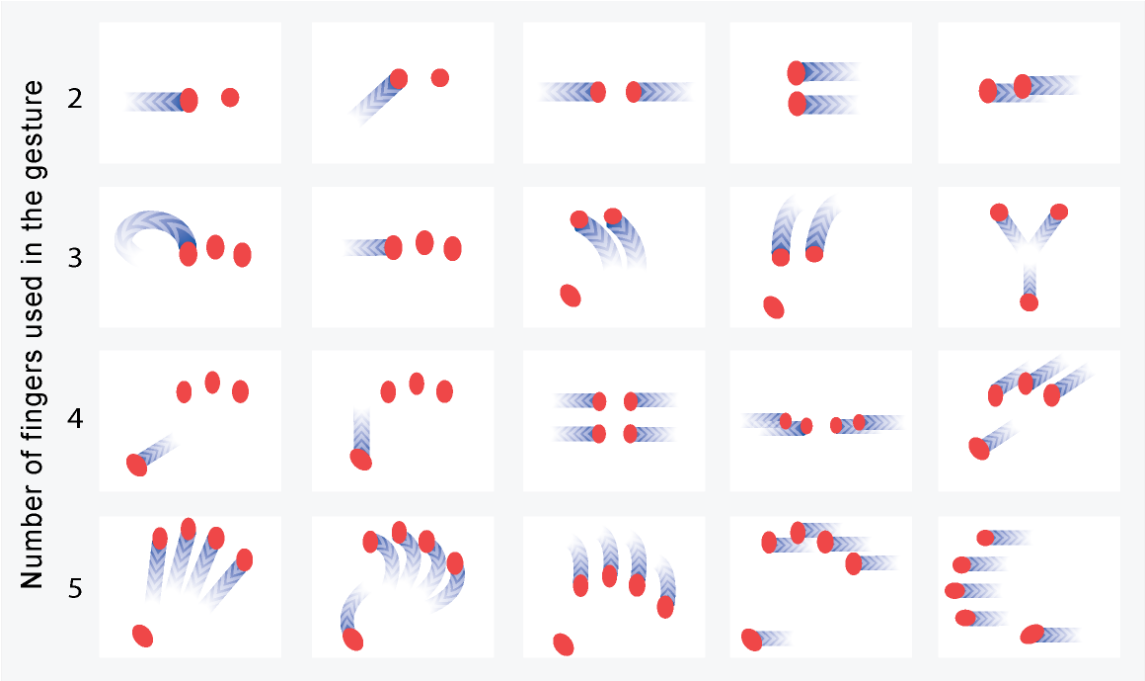

Figure 7.5: A sample gesture set, selected to test the performance of the mGestr system.

results show that our technique recognizes gestures within a reasonable accuracy for practical use. As it can be trained using samples from more than one user, the trained HMM can also be stored within an application or shared across users to avoid re-training.

We investigated the systems suitability for user-independent gesture recognition. A leave-one-user-out cross validation was performed with the collected samples by excluding one users samples from the random selection of 20 training samples during the learning stage. The testing unit then performed recognition of only this users samples. The recognition rates averaged over 20 runs for two, three, four and five fingers are 88.12%, 86.63%, 92.83%, 75.83% with an average recognition rate of 85.74% and the result was statistically significant ($p < .0001$). This shows that the

recognition system is robust in supporting the recognition of gestures performed by a user not involved in the learning process.

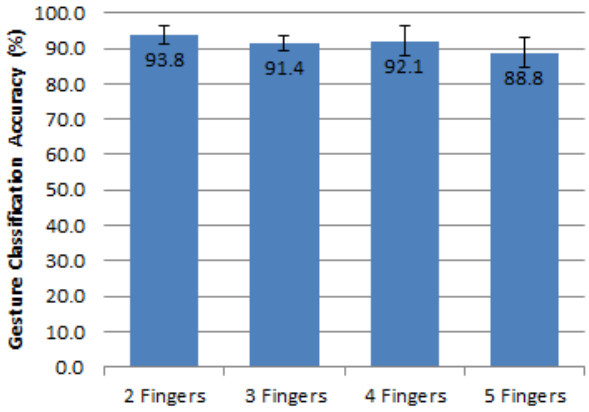

Figure 7.6: The results from the evaluation. Results are grouped by number of fingers, each group contained five gestures.

7.3 GRaaS: gesture recognition as a service

We describe components of a recognition system operating as a application-independent service. Communication to this service is achieved via OSC messages [122]. By encapsulating the recognition as a stand-alone service, it can be used by an interactive application on any platform. A graphical interface is designed with the following functionality:

1. Saving new gesture samples
2. Browsing through a repository of previous samples
3. Selecting a set of samples to train a new gesture model

7.3.1 Graphical interface

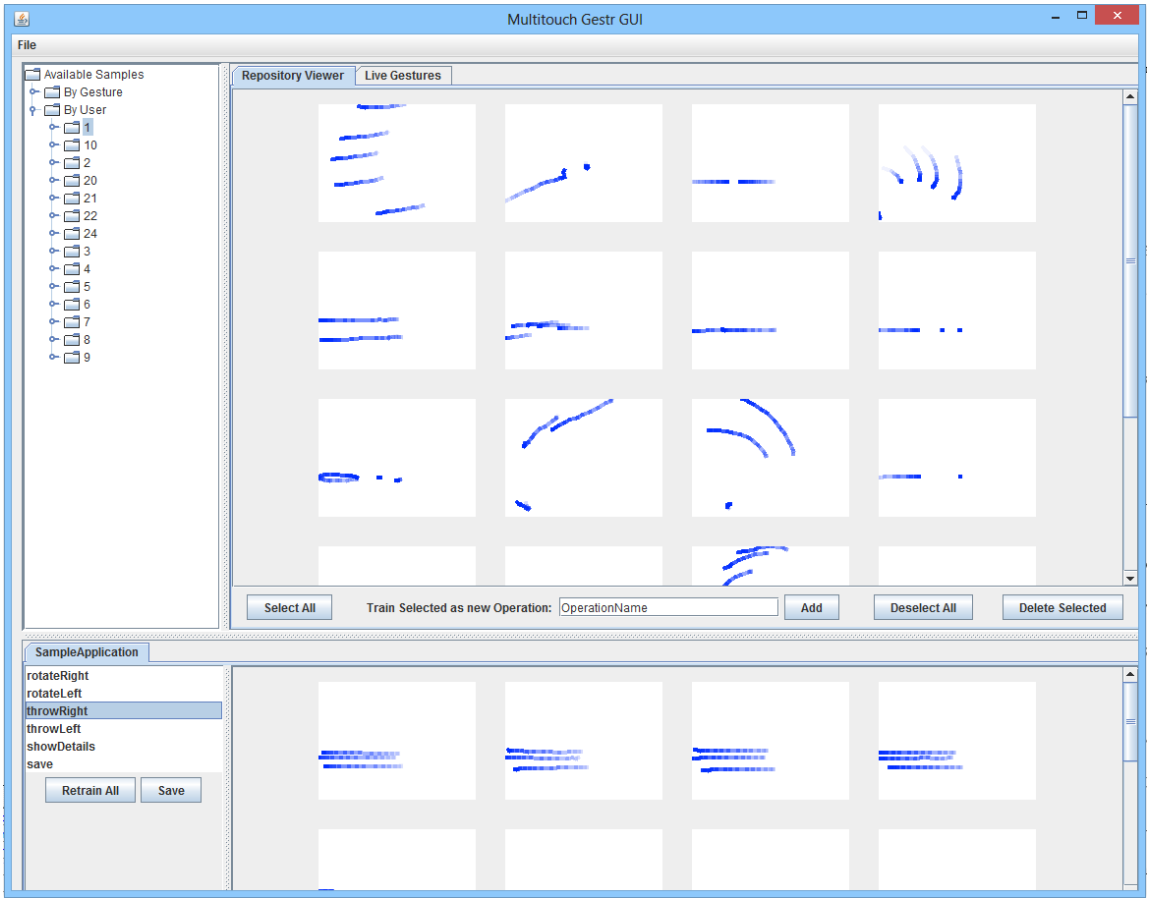

Figure 7.7: The graphical interface for the gesture repository.

A graphical interface (see Figure 7.7) shows repository of saved samples on top, and an example gesture set on the bottom. The samples are organized in two top-level hierarchies: by user and by gesture. This dual categorization offers a gesture designer two options of navigating the repository. The interface also provides the ability to save new gestures. To enter new gestures, a gesture designer first states the number of fingers used in the gesture, and an id for the user performing them.

As the gesture is performed, the interface shows the points tracked. In case of the gesture receiving more or less fingers than required, the interface reports an error.

7.3.2 Communication interface

The gesture training and recognition system can be fully operated by sending OSC messages. We choose this messaging protocol since it has a very low requirement to use in an application, and is designed for a low-latency scenario such as ours. The following list of messages details the supported functionality. Message use `/gestr/action` as their OSC address pattern [122]. The first string argument is the action. The responses for each message is also detailed here, with their arguments.

- **`save`** The save message uses the second argument to serialize the active gesture set. Saved binary files have a .gestr extension.
  Response: **`saved`** `filepath_saved_to`

- **`load`** The second argument is used the load a gesture set from a .gestr file.
  Response: **`loaded`** `filepath_loaded_from`

- **`clear`** Clears any active gesture set. No gestures will be recognized after a clear message.
  Response: **`cleared`**

- **`parameterize`** The parameterize message (see Parameterize) allows addition of parameters to any previously trained gesture. The arguments for this message are (in order) i) the name of the gesture to parameterize ii) the name of the new parameter and iii) the parameter specification string.
  Response: `parameterized "description_of_parameterization"`

- **`train`** The train message will train a new gesture with the collected gesture samples.

  Response: `trained gesture name`

- **`recognize`** The recognize message uses the active gesture set to recognize the last sample collected.

  Response: `recognized gesture_name X Y`

For `save` and `load`, the second argument, the file path, can be either relative to the executable, or absolute.

In addition to the `/gestr/action` messages, MGestr uses the `/gestr/sample` address pattern to mark the start and end of a gesture sample. The first argument for this message is one of the two strings: 'start' and 'end'. This control flow is used by the graphical user interface to pass saved gesture samples to MGestr.

Parameter updates are sent via a `/gestr/action` message with first argument as `parameter_update` followed by the list of arguments associated with the parameter:

`/gestr/action` **`parameter_update`** `parameter_name parameter_value_1 parameter_value_2`

7.3.3 Segmentation and recognition

The MGestr system reads an multitouch input stream to identify gesture segments and subsequently recognize them using the currently active gesture set. A gesture segment is a sequence of frames of a fixed number of fingers moving across the multi-touch surface. The start of a gesture is made by a texttt*finger_down* event, by placing a finger on screen. If more fingers are placed on screen, the previous frames are discarded. Once the finger(s) begin moving, the frames are stored as a gesture segment. To identify the end of gestures, two delimiters are used.

1. When a finger is lifted off the surface, the gesture segment is marked as ended.

2. Check whether all the fingers have stopped moving (within a small threshold). If all fingers have stopped moving, the gesture segment is ended. Any subsequent movement is considered a new gesture segment.

To filter noisy input, a minimum number of frames are required to be collected before a gesture segment is recognized. The gesture segment is formed, it is tested against the active gesture set. The result of the recognition is sent via OSC messages on the output port.

7.3.4 Parameterization

We add functionality to a recognized gesture by associating certain parameters to the gesture. Parameters values are derived directly or calculated from the stream of features provided by the input device. On recognizing a gesture, if the gesture is associated with parameters, a `parameter_update` messages is sent on the output port for every frame. The parameterization stage ends only when all fingers are raised off the surface. Multiple parameters can be associated with a gesture. Gestures can be parameterized by providing a parameterize message (see Communication). A parameter can be defined by using one of the types provided. Each parameter type can be declared by a string. The following list shows the list of currently supported parameter types, along with their required information.

- **`fing_x`** *`fing-index`* : the x coordinate of the finger identified by the index is used as the parameter value.

- **`fing_y`** *`fing-index`* : the y coordinate of the finger identified by the index is used as the parameter value.

- **fing_dist** *fing-index-1 fing-index-2* : A parameter is derived by calculating the distance between two fingers, identified by their respective indeces.

- **all_mean** : The mean of all fingers on screen.

During parameterization, if a finger is raised, parameterization is paused. Updates of the parameters are not sent while paused. Once the finger is placed back on the surface, the parameterization resumes. This allows the user to comfortably raise and lower a finger during parameterization without having to repeat the gesture.

7.3.4.1 Example

Consider the parameterization of a three finger gesture (let's call it GestureA), that uses the thumb, index and middle fingers. The following messages add 3 parameters to this gesture. In this case the indices of the thumb, index and middle fingers are 0, 1 and 2 respectively.

/gestr/action `"parameterize" "GestureA" "Param1" "fing_dist 0 2"` This adds a parameter whose value is the distance between the thumb and middle finger. Moving the thumb into the palm would yield smaller values, and extending the thumb away from the palm would yield higher parameter values.

/gestr/action `"parameterize" "GestureA" "Param2" "fing_y 1"` The y coordinate of the index finger is used as the second parameter.

7.4 Design principles

We need to develop further methods for delimiting gestures, to bring these methods into real world applications. We do not want users to have to regularly remove their hands from the surface, as is required for gesture delimiting in the present implementation.

One lesson learned during the design of our gesture vocabulary is that it is easy to inadvertently design gestures that turn out to be similar, confusing the recognizer. The need for clear recognition may pose constraints on the gesture design (choreography) process. Thus, an iterative design process is called for, in which gestures are defined in relationship to actions, and then prototyped in the recognition system. Both the gesture set and the recognition system will need to be tuned to optimize performance, and the participant experience. We can aid designers with a similarity

The system we developed always returns a recognized gesture, based on the set of gestures that are currently considered. We need to enforce a rejection scheme that will discard samples that do not achieve an acceptable match to the gestures considered. This can be done by placing a lower threshold on the value returned by the HMMs for recognized gestures.

Another imperative part of the iterative design process is to take human participants into account. We have found that people vary in their manual dexterity, as manifested in their affinity for performing particular gestures. Personality can be a factor in which gestures will feel natural to an individual. The design implication here is that gestural advanced visual interfaces must support more than one set of mappings. Participants must be free to choose the one that best suits them, and, further, to customize to create their own personal gesture set and associated.

7.5 Section summary

We have developed a robust user-independent method for learning and recognizing multi-touch gestures using Hidden Markov Models. The image processing library extracts the screen co-ordinates for each finger in contact with the screen. Gesture segments are temporally delimited by periods of contact by one or more fingers. Feature data for a gesture segment is processed by the recognition pipeline, terminating

with the HMM. The data validates the method.

We achieved a recognition rate of 95.06% overall across gestures using samples from all users, while a leave one out cross validation on users produced a recognition rate of 85.74%. Thus, personalized training data is not required. Further, we have identified a crucial problem for robust, user-independent recognition: consistent ordering of the feature values for the fingers of the human hand during the course of a single gesture, and presented an initial solution to this problem.

Future work will develop intuitive mappings between gestures and actions. We hypothesize that natural mappings can help users overcome the experience of command overload, in which they feel overwhelmed by the vast set of possible operations, and the menu and icon systems that deliver them. The more varied and complex the set of operations in a system, the more the user must remember and negotiate.

We need to develop human-centered systems to enable motivated users to create their own gesture/mappings, and social systems for sharing the customizations.

8. CONCLUSION

We have presented two main products as a contribution to multi-touch computing. The first is a technique to learn by example and subsequently recognize arbitrary gestures in real-time. The second is a new interaction technique that uses comfortable, relaxed movements of the human hand to both select and adjust multiple numerical parameters.

We shift our focus to the domain of non-spatial parameter spaces, as a problem in navigation. We introduce this space as an important new domain for interaction research. An operation that requires changing multiple parameters would benefit from a fluid technique that connects adjustment with selection. Our interaction studies mappings of movements of the hand with changes in the parameter. Our initial study on techniques for color selection in Section 3 finds that participants were quick to learn new mappings to pick from hue, saturation and value and subsequently specify a value for each dimension.

In Section 4, we leverage this skill of learning new mappings to develop a new technique. The goal was to develop a generic technique that can be applied to a number of different parameter spaces. We approach this by first analyzing the physiology of the human hand, specifically the range of comfortable movements that can be detected by a multi-touch screen. The index finger can be moved precisely within a short range of $\approx$ 2 inches, with little effort. Our technique enables exploration of multi-dimensional parametric spaces with the available dexterity of the human hand. An evaluation of our new interaction technique shows an improvement over traditional sliders.

Section 5 describes the application of the MultiTap sliders in another non-spatial

context. The data set is a series of textual posts on the social networking site Myspace [87], tagged with metadata about the author. We describe a scenario in which a researcher is tasked to discover patterns in the data. She begins by calculating quantitative parameters (or features) of each post. These are then visualized. The MultiTap slider help her to navigate through the data set by filtering it through smaller subset of parameter values.

The research on multi-tap sliders demonstrates a design approach that satisfies a number of constraints: (1) interaction should not require the participant to target specific visual elements that distract her from the focus of the operation (2) the interaction must consist only of movements within the comfortable range of motion of the hand, and (3) the movement required to transition from selecting a parameter to adjusting a parameter must be as fluid as possible. The design approach emphasizes the interaction before the visual. The visual is present for feedback when necessary.

We perform an exploration of the design space of multi-tap sliders. We add to the functionality of the sliders using the thumb, and the non-dominant hand. The wide applicability of the sliders make this research a strong contribution to the field of human computer interaction.

mGestr is presented in Section 7 as a platform for building new multi-touch gestural interfaces. Issues of pre-processing multi-touch input for consistency are raised and solutions presented. To evaluate the recognition system, we develop a sample set that spans a range of movements. Evaluation shows a success rate of 95.06%. We develop a service with a network based communication system to allow easy integration with interactive software.

REFERENCES

[1] Gregory D. Abowd and Elizabeth D. Mynatt. Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction*, 7(1):29–58, March 2000.

[2] Anand Agarawala and Ravin Balakrishnan. Keepin' it real: pushing the desktop metaphor with physics, piles and the pen. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 1283–1292, New York, NY, USA, 2006. ACM.

[3] Pär-Anders Albinsson and Shumin Zhai. High precision touch screen interaction. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI '03*, page 105, New York, New York, USA, April 2003. ACM Press.

[4] Apple. Magic Trackpad. `http://www.apple.com/magictrackpad/`. Accessed: 2013-03-22.

[5] Gilles Bailly, Eric Lecolinet, and Yves Guiard. Finger-count & radial-stroke shortcuts: 2 techniques for augmenting linear menus on multi-touch surfaces. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems*, CHI '10, pages 591–594, New York, NY, USA, 2010. ACM.

[6] Nikola Banovic, Frank Chun Yat Li, David Dearman, Koji Yatani, and Khai N. Truong. Design of unimanual multi-finger pie menu interaction. In *Proceedings of the ACM International Conference on Interactive Tabletops*

and Surfaces - ITS '11, page 120, New York, New York, USA, November 2011. ACM Press.

[7] Olivier Bau and Wendy E. Mackay. OctoPocus. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology - UIST '08*, UIST '08, page 37, New York, New York, USA, October 2008. ACM Press.

[8] Leonard Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164 – 171, 1970.

[9] Michel Beaudouin-Lafon. Designing interaction, not interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces - AVI '04*, page 15, New York, New York, USA, May 2004. ACM Press.

[10] Benjamin B. Bederson and James D. Hollan. Pad++. In *Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology - UIST '94*, pages 17–26, New York, New York, USA, November 1994. ACM Press.

[11] Hrvoje Benko, Andrew D Wilson, and Patrick Baudisch. Precise selection techniques for multi-touch screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 1263–1272, New York, NY, USA, 2006. ACM.

[12] Anastasia Bezerianos and Ravin Balakrishnan. The vacuum: facilitating the manipulation of distant objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 361–370, New York, NY, USA, 2005. ACM.

[13] Eric A Bier, Maureen C Stone, Ken Pier, William Buxton, and Tony D DeRose. Toolglass and magic lenses: the see-through interface. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 73–80, New York, NY, USA, 1993. ACM.

[14] Andrew Bragdon, Arman Uguray, Daniel Wigdor, Stylianos Anagnostopoulos, Robert Zeleznik, and Rutledge Feman. Gesture play: motivating online gesture learning with fun, positive reinforcement and physical metaphors. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 39–48, New York, NY, USA, 2010. ACM.

[15] Peter Brandl, Jakob Leitner, Thomas Seifried, Michael Haller, Bernard Doray, and Paul To. Occlusion-aware menu design for digital tabletops. In *Proceedings of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems*, CHI '09, pages 3223–3228, New York, NY, USA, 2009. ACM.

[16] William Buxton. Chunking and phrasing and the design of human-computer dialogues. In *Proceedings of the IFIP World Computer Congress*, pages 475–480. North Holland Publishers, 1986.

[17] Xiang Cao, Nicolas Villar, and Shahram Izadi. Comparing user performance with single-finger, whole-hand, and hybrid pointing devices. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems - CHI '10*, page 1643, New York, New York, USA, April 2010. ACM Press.

[18] Xiang Cao, Andrew D Wilson, Ravin Balakrishnan, Ken Hinckley, and Scott E Hudson. ShapeTouch: leveraging contact shape on interactive surfaces. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. IEEE, 2008.

[19] Stuart K Card. User perceptual mechanisms in the search of computer command menus. In *Proceedings of the 1982 Conference on Human Factors in Computing Systems*, CHI '82, pages 190–196, New York, NY, USA, 1982. ACM.

[20] João Carreira and Paulo Peixoto. Nuisance free recognition of hand postures over a tabletop display. In *Proceedings of the HCSNet Workshop on Use of Vision in Human-Computer Interaction - Volume 56*, VisHCI '06, pages 73–78, Darlinghurst, Australia, 2006. Australian Computer Society, Inc.

[21] Leslie E Chipman, Benjamin B Bederson, and Jennifer A Golbeck. SlideBar: Analysis of a linear input device. *Behaviour & Information Technology*, 23(1):1–9, January 2004.

[22] Richard Coyne. *Designing Information Technology in the Postmodern Age: from Method to Metaphor*. MIT Press, Cambridge, MA, USA, 1995.

[23] Sashikanth Damaraju and Andruid Kerne. Comparing multi-touch interaction techniques for manipulation of an abstract parameter space. In *Proceedings of the 13th International Conference on Multimodal Interfaces - ICMI '11*, page 221, New York, New York, USA, November 2011. ACM Press.

[24] Philip L Davidson and Jefferson Y Han. Extending 2D object arrangement with pressure-sensitive layering cues. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*, UIST '08, pages 87–90, New York, NY, USA, 2008. ACM.

[25] Jozesf Denes and A. Donald Keedwell. *Latin Squares and Their Applications*. Academic Press Inc, 1974.

[26] Paul Dietz and Darren Leigh. DiamondTouch: a multi-user touch technology. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, UIST '01, pages 219–226, New York, NY, USA, 2001. ACM.

[27] Paul H Dietz, Jonathan Westhues, John Barnwell, Jefferson Y Han, and William Yerazunis. Submerging technologies. In *ACM SIGGRAPH 2006 Emerging Technologies*, SIGGRAPH '06, New York, NY, USA, 2006. ACM.

[28] Florian Echtler, Andreas Dippon, Marcus Tönnis, and Gudrun Klinker. Inverted FTIR. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces - ITS '09*, page 29, New York, New York, USA, November 2009. ACM Press.

[29] Alan Esenther and Kathy Ryall. Fluid DTMouse: better mouse support for touch-based interactions. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '06, pages 112–115, New York, NY, USA, May 2006. ACM.

[30] Katherine Everitt, Chia Shen, Kathy Ryall, and Clifton Forlines. Modal spaces: spatial multiplexing to mediate direct-touch input on large displays. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI '05, pages 1359–1362, New York, NY, USA, 2005. ACM.

[31] Kenneth P Fishkin. A taxonomy for and analysis of tangible interfaces. *Personal Ubiquitous Computing*, 8(5):347–358, September 2004.

[32] David Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[33] Dustin Freeman, Hrvoje Benko, Meredith Ringel Morris, and Daniel Wigdor. ShadowGuides: visualizations for in-situ learning of multi-touch and whole-hand gestures. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces - ITS '09*, ITS '09, pages 165–172, New York, New York, USA, November 2009. ACM Press.

[34] Tovi Grossman, Daniel Wigdor, and Ravin Balakrishnan. Multi-finger gestural interaction with 3D volumetric displays. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, UIST '04, pages 61–70, New York, NY, USA, 2004. ACM.

[35] François Guimbretière, Andrew Martin, and Terry Winograd. Benefits of Merging Command Selection and Direct Manipulation. *ACM Transactions on Computer-Human Interaction*, 12(3):460–476, September 2005.

[36] François Guimbretière, Maureen Stone, and Terry Winograd. Fluid interaction with high-resolution wall-size displays. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, UIST '01, pages 21–30, New York, NY, USA, 2001. ACM.

[37] François Guimbretiére and Terry Winograd. FlowMenu: combining command, text, and data entry. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, UIST '00, pages 213–216, New York, NY, USA, November 2000. ACM.

[38] Jefferson Y Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*, UIST '05, pages 115–118, New York, NY, USA, 2005. ACM.

[39] Mark Hancock, Sheelagh Carpendale, and Andy Cockburn. Shallow-depth 3D interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '07*, page 1147, New York, New York, USA, April 2007. ACM Press.

[40] Tobias Hesselmann, Stefan Flöring, and Marwin Schmitt. Stacked Half-Pie menus: navigating nested menus on interactive tabletops. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, pages 173–180, New York, NY, USA, 2009. ACM.

[41] Ken Hinckley, Francois Guimbretiere, Maneesh Agrawala, Georg Apitz, and Nicholas Chen. Phrasing techniques for multi-stroke selection gestures. In *Proceedings of Graphics Interface 2006*, GI '06, pages 147–154, Toronto, Ont., Canada, Canada, 2006. Canadian Information Processing Society.

[42] Ken Hinckley, Francois Guimbretiere, Patrick Baudisch, Raman Sarin, Maneesh Agrawala, and Ed Cutrell. The springboard: multiple modes in one spring-loaded control. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 181–190, New York, NY, USA, 2006. ACM.

[43] Richard Sewall Hunter. Accuracy, Precision, and Stability of New Photo-electric Color-Difference Meter. *Journal of the Optical Society of America*, 38(12):1092, December 1948.

[44] Samuel A. Iacolina, Alessandro Soro, and Riccardo Scateni. Improving FTIR based multi-touch sensors with IR shadow tracking. In *Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems - EICS '11*, page 241, New York, New York, USA, June 2011. ACM Press.

[45] Adobe Inc. Photoshop Lightroom. http://www.adobe.com/products/photoshoplightroom.edu.html. Accessed: 2013-03-22.

[46] Apple Inc. Expose. http://support.apple.com/kb/ht2503. Accessed: 2013-03-22.

[47] Apple Inc. iPad. http://www.apple.com/ipad. Accessed: 2013-03-22.

[48] Apple Inc. Mac. http://www.apple.com/mac. Accessed: 2013-03-22.

[49] Microsoft Inc. PixelSense Hardware. http://www.microsoft.com/en-us/pixelsense/default.aspx. Accessed: 2013-03-22.

[50] Microsoft Inc. XBox Kinect. http://www.xbox.com/kinect. Accessed: 2013-03-22.

[51] Tableau Inc. Tableau Public. http://www.tableausoftware.com/. Accessed: 2013-03-22.

[52] Hiroshi Ishii and Brygg Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '97, pages 234–241, New York, NY, USA, 1997. ACM.

[53] Robert J K Jacob, Audrey Girouard, Leanne M Hirshfield, Michael S Horn, Orit Shaer, Erin Treacy Solovey, and Jamie Zigelbaum. Reality-based interaction: a framework for post-WIMP interfaces. In *Proceeding of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 201–210, New York, NY, USA, 2008. ACM.

[54] Robert J K Jacob, Linda E Sibert, Daniel C McFarlane, and M Preston Mullen Jr. Integrality and separability of input devices. *ACM Transactions on Computer Human Interaction*, 1(1):3–26, March 1994.

[55] Jahmm. `https://code.google.com/p/jahmm/`. Accessed: 2013-03-22.

[56] Sergi Jordà, Günter Geiger, Marcos Alonso, and Martin Kaltenbrunner. The reacTable: exploring the synergy between live music performance and tabletop tangible interfaces. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, TEI '07, pages 139–146, New York, NY, USA, 2007. ACM.

[57] Biing-Hwang Juang and Lawrence R. Rabiner. The segmental K-means algorithm for estimating parameters of hidden Markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(9):1639–1641, 1990.

[58] Martin Kaltenbrunner and Ross Bencina. reacTIVision: a computer-vision framework for table-based tangible interaction. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, TEI '07, pages 69–74, New York, NY, USA, 2007. ACM.

[59] Dietrich Kammer, Frank Lamack, Rainer Groh, Boris de Ruyter, Reiner Wichert, David Keyson, Panos Markopoulos, Norbert Streitz, Monica Divitini, Nikolaos Georgantas, and Antonio Mana Gomez. Enhancing the expressiveness of fingers: multi-touch ring menus for everyday applications. In Boris Ruyter, Reiner Wichert, David V. Keyson, Panos Markopoulos, Norbert Streitz, Monica Divitini, Nikolaos Georgantas, and Antonio Mana Gomez, editors, *Ambient Intelligence*, volume 6439 of *Lecture Notes in Computer Science*, pages 259–264. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[60] Kenrick Kin, Maneesh Agrawala, and Tony DeRose. Determining the benefits of direct-touch, bimanual, and multifinger input on a multitouch workstation. In *Proceedings of Graphics Interface 2009*, GI '09, pages 119–124, Toronto, Ont., Canada, Canada, 2009. Canadian Information Processing Society.

[61] Kenrick Kin, Björn Hartmann, Tony DeRose, and Maneesh Agrawala. Proton++: A Customizable Declarative Multitouch Framework. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology - UIST '12*, page 477, New York, New York, USA, October 2012. ACM Press.

[62] Russell Kruger, Sheelagh Carpendale, Stacey D. Scott, and Saul Greenberg. Roles of orientation in tabletop collaboration: comprehension, coordination and communication. *Computer Supported Cooperative Work (CSCW)*, 13(5-6):501–537, December 2004.

[63] Russell Kruger, Sheelagh Carpendale, Stacey D. Scott, and Anthony Tang. Fluid integration of rotation and translation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '05*, page 601, New York, New York, USA, April 2005. ACM Press.

[64] Gordon Kurtenbach and William Buxton. Issues in combining marking and direct manipulation techniques. In *Proceedings of the 4th Annual ACM Symposium on User Interface Software and Technology*, UIST '91, pages 137–144, New York, NY, USA, October 1991. ACM.

[65] Gordon Kurtenbach and William Buxton. 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, pages 482–487, New York, NY, USA, 1993. ACM.

[66] Celine Latulipe, Ian Bell, Charles L A Clarke, and Craig S Kaplan. symTone: two-handed manipulation of tone reproduction curves. In *Proceedings of Graphics Interface 2006*, GI '06, pages 9–16, Toronto, Ont., Canada, Canada, 2006. Canadian Information Processing Society.

[67] Andrea Leganchuk, Shumin Zhai, and William Buxton. Manual and cognitive benefits of two-handed input: an experimental study. *ACM Transactions on Computer Human Interaction*, 5(4):326–359, December 1998.

[68] G Julian Lepinski, Tovi Grossman, and George Fitzmaurice. The design and evaluation of multitouch marking menus. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems*, CHI '10, pages 2233–2242, New York, NY, USA, 2010. ACM.

[69] A Chris Long Jr., James A Landay, Lawrence A Rowe, and Joseph Michiels. Visual similarity of pen gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '00, pages 360–367, New York, NY, USA, 2000. ACM.

[70] Allan Christian Long Jr., James A Landay, and Lawrence A Rowe. Implications for a gesture design tool. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, pages 40–47, New York, NY, USA, 1999. ACM.

[71] Hao Lü and Yang Li. Gesture coder: a tool for programming multi-touch gestures by demonstration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 2875–2884, New York, NY, USA, 2012. ACM.

[72] George Lucchese, Martin Field, Jimmy Ho, Ricardo Gutierrez-Osuna, and Tracy Hammond. Gesturecommander: continuous touch-based gesture

prediction. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '12, pages 1925–1930, New York, NY, USA, 2012. ACM.

[73] I. Scott MacKenzie. Fitts' Law as a Research and Design Tool in Human-Computer Interaction. *Human-Computer Interaction*, 7(1):91–139, March 1992.

[74] Shahzad Malik. *An exploration of multi-finger interaction on multi-touch surfaces*. PhD thesis, Toronto, Ont., Canada, Canada, 2007.

[75] Shahzad Malik, Abhishek Ranjan, and Ravin Balakrishnan. Interacting with large displays from a distance with vision-tracked multi-finger gestural input. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*, UIST '05, pages 43–52, New York, NY, USA, 2005. ACM.

[76] Anthony Martinet, Géry Casiez, and Laurent Grisoni. The effect of DOF separation in 3D manipulation tasks with multi-touch displays. In *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology - VRST '10*, page 111, New York, New York, USA, November 2010. ACM Press.

[77] Andrea H. Mason and Pamela J. Bryden. Coordination and concurrency in bimanual rotation tasks when moving away from and toward the body. *Experimental brain research. Experimentelle Hirnforschung. Expérimentation cérébrale*, 183(4):541–56, December 2007.

[78] Justin Matejka, Tovi Grossman, Jessica Lo, and George Fitzmaurice. The design and evaluation of multi-finger mouse emulation techniques. In *Proceedings of the 27th International Conference on Human Factors in*

Computing Systems, CHI '09, pages 1073–1082, New York, NY, USA, 2009. ACM.

[79] Michael Mcguffin, Nicolas Burtnyk, Gordon Kurtenbach, and King Street. FaST Sliders: integrating marking menus and the adjustment of continuous values. In *Proceedings of Graphics Interface*, GI '02, pages 35–42, 2002.

[80] Jon Moeller and Andruid Kerne. Scanning FTIR: unobtrusive optoelectronic multi-touch sensing through waveguide transmissivity imaging. In *Proceedings of the Fourth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '10, pages 73–76, New York, NY, USA, 2010. ACM.

[81] Jon Moeller, Andruid Kerne, and Sashikanth Damaraju. ZeroTouch. In *Proceedings of the 2011 Annual Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA '11*, page 1165, New York, New York, USA, May 2011. ACM Press.

[82] Carlos Morimoto, Yaser Yacoob, and Larry Davis. Recognition of head gestures using hidden Markov models. In *Proceedings of 13th International Conference on Pattern Recognition*, pages 461–465 vol.3. IEEE, 1996.

[83] Meredith Ringel Morris, Jacob O Wobbrock, and Andrew D Wilson. Understanding users' preferences for surface gestures. In *Proceedings of Graphics Interface 2010*, GI '10, pages 261–268, Toronto, Ont., Canada, Canada, 2010. Canadian Information Processing Society.

[84] Tomer Moscovich. Contact area interaction with sliding widgets. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology*, UIST '09, pages 13–22, New York, NY, USA, October 2009. ACM.

[85] Tomer Moscovich and John F. Hughes. Indirect mappings of multi-touch input using one and two hands. In *Proceeding of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1275–1284, New York, NY, USA, April 2008. ACM.

[86] Michael Moyle and Andy Cockburn. The design and evaluation of a flick gesture for 'back' and 'forward' in web browsers. In *Proceedings of the Fourth Australasian user interface Conference on User interfaces 2003 - Volume 18*, AUIC '03, pages 39–46, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[87] Myspace. `http://www.myspace.com`. Accessed: 2013-03-22.

[88] Miguel A. Nacenta, Patrick Baudisch, Hrvoje Benko, and Andy Wilson. Separability of spatial manipulations in multi-touch interfaces. pages 175–182, May 2009.

[89] Donald A. Norman. *The Design of Everyday Things*. Basic Books, 2002.

[90] NUIGroup. `http://ccv.nuigroup.com/`. Accessed: 2013-03-22.

[91] Daniel L Odell, Richard C Davis, Andrew Smith, and Paul K Wright. Toolglasses, marking menus, and hotkeys: a comparison of one and two-handed command selection techniques. In *Proceedings of Graphics Interface 2004*, GI '04, pages 17–24, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

[92] Alex Olwal, Steven Feiner, and Susanna Heyman. Rubbing and tapping for precise and rapid selection on touch-screen displays. In *Proceeding of the Twenty-Sixth Annual CHI Conference on Human Factors in Computing*

Systems - CHI '08, page 295, New York, New York, USA, April 2008. ACM Press.

[93] Stuart Pook, Eric Lecolinet, Guy Vaysseix, and Emmanuel Barillot. Control menus. In *CHI '00 Extended Abstracts on Human Factors in Computing Systems*, page 263, New York, New York, USA, April 2000. ACM Press.

[94] Richard Lee Potter, Linda J Weldon, and Ben Shneiderman. Improving the accuracy of touch screens: an experimental evaluation of three strategies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '88, pages 27–32, New York, NY, USA, 1988. ACM.

[95] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[96] Jason L Reisman, Philip L Davidson, and Jefferson Y Han. A screen-space formulation for 2D and 3D direct manipulation. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology*, UIST '09, pages 69–78, New York, NY, USA, 2009. ACM.

[97] Jun Rekimoto. Pick-and-drop: a direct manipulation technique for multiple computer environments. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology*, UIST '97, pages 31–39, New York, NY, USA, 1997. ACM.

[98] Jun Rekimoto. SmartSkin: an infrastructure for freehand manipulation on interactive surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '02, pages 113–120, New York, NY, USA, 2002. ACM.

[99] Dean Rubine. Specifying gestures by example. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '91, pages 329–337, New York, NY, USA, 1991. ACM.

[100] Dean Rubine. Combining gestures and direct manipulation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, pages 659–660, New York, NY, USA, 1992. ACM.

[101] Thecla Schiphorst. soft(n). In *Proceedings of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA '09*, page 2427, New York, New York, USA, April 2009. ACM Press.

[102] Michael Schmidt and Gerhard Weber. Enhancing single touch gesture classifiers to multitouch support. In *Proceedings of the 12th International Conference on Computers Helping People with Special Needs*, ICCHP'10, pages 490–497, Berlin, Heidelberg, July 2010. Springer-Verlag.

[103] Andrew Sears and Ben Shneiderman. High precision touchscreens: design strategies and comparisons with a mouse. *International Journal of Man-Machine Studies*, 34(4):593–613, April 1991.

[104] Ben Shneiderman. Direct Manipulation: A Step Beyond Programming Languages. *Computer*, 16(8):57–69, August 1983.

[105] Sikuli. GUI Automation framework. `http://www.sikuli.org/`. Accessed: 2013-03-22.

[106] Jason Stewart, Benjamin B. Bederson, and Allison Druin. Single display groupware. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 286–293, New York, New York, USA, May 1999. ACM Press.

[107] Ahmed N Sulaiman and Patrick Olivier. Attribute gates. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*, UIST '08, pages 57–66, New York, NY, USA, 2008. ACM.

[108] Yoshiki Takeoka, Takashi Miyaki, and Jun Rekimoto. Z-touch: an infrastructure for 3D gesture interaction in the proximity of tabletop surfaces. In *ACM International Conference on Interactive Tabletops and Surfaces - ITS '10*, page 91, New York, New York, USA, November 2010. ACM Press.

[109] Jerry O. Talton, Daniel Gibson, Lingfeng Yang, Pat Hanrahan, and Vladlen Koltun. Exploratory modeling with collaborative design spaces. *ACM Transactions on Graphics*, 28(5):1, December 2009.

[110] Huawei Tu, Xiangshi Ren, and Shumin Zhai. A comparative evaluation of finger and pen stroke gestures. In *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems - CHI '12*, page 1287, New York, New York, USA, May 2012. ACM Press.

[111] Andrew Webb and Andruid Kerne. The in-context slider: a fluid interface component for visualization and adjustment of values while authoring. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '08, pages 91–99, New York, NY, USA, 2008. ACM.

[112] Wayne Westerman. *Hand Tracking, Finger Identification, and Chordic Manipulation on a Multi-Touch Surface*. PhD thesis, University of Delaware, 1999.

[113] Daniel Wigdor, Hrvoje Benko, John Pella, Jarrod Lombardo, and Sarah Williams. Rock & rails : extending multi-touch interactions with shape gestures to enable precise spatial manipulations. In *Proceedings of the 2011*

Annual Conference on Human Factors in Computing Systems - CHI '11, page 1581, New York, New York, USA, May 2011. ACM Press.

[114] Daniel Wigdor, Darren Leigh, Clifton Forlines, Samuel Shipman, John Barnwell, Ravin Balakrishnan, and Chia Shen. Under the table interaction. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*, UIST '06, pages 259–268, New York, NY, USA, 2006. ACM.

[115] Daniel Wigdor, Sarah Williams, Michael Cronin, Robert Levy, Katie White, Maxim Mazeev, and Hrvoje Benko. Ripples: utilizing per-contact visualizations to improve user interaction with touch displays. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology - UIST '09*, UIST '09, page 3, New York, New York, USA, October 2009. ACM Press.

[116] Daniel Wigdor and Dennis Wixon. *Brave NUI World: Designing Natural User Interfaces for Touch and Gesture*. Morgan Kaufmann, 2011.

[117] A.D. Wilson and A.F. Bobick. Parametric hidden Markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):884–900, 1999.

[118] Andrew D. Wilson. TouchLight: an imaging touch screen and display for gesture-based interaction. In *Proceedings of the 6th International Conference on Multimodal Interfaces*, ICMI '04, pages 69–76, New York, NY, USA, October 2004. ACM.

[119] Andrew D Wilson, Shahram Izadi, Otmar Hilliges, Armando Garcia-Mendoza, and David Kirk. Bringing physics to the surface. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*, UIST '08, pages 67–76, New York, NY, USA, 2008. ACM.

[120] Jacob O Wobbrock, Meredith Ringel Morris, and Andrew D Wilson. User-defined gestures for surface computing. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems*, CHI '09, pages 1083–1092, New York, NY, USA, 2009. ACM.

[121] Jacob O Wobbrock, Andrew D Wilson, and Yang Li. 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, pages 159–168, New York, NY, USA, 2007. ACM.

[122] Matt Wright and Adrian Freed. Open sound control: a new protocol for communicating with sound synthesizers. In *Proceedings of the 1997 International Computer Music Conference*, pages 101–104, Thessaloniki, Hellas, Greece, 1997.

[123] Mike Wu and Ravin Balakrishnan. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*, UIST '03, pages 193–202, New York, NY, USA, 2003. ACM.

[124] Mike Wu, Chia Shen, Kathy Ryall, Clifton Forlines, and Ravin Balakrishnan. Gesture Registration, Relaxation, and Reuse for Multi-Point Direct-Touch Surfaces. *Proceedings of IEEE Tabletop - The International Workshop on Horinztal Interactive Human Computer Systems*, 2006:183 – 190, 2005.

[125] Junji Yamato, Jun Ohya, and Ken-Ichiro Ishii. Recognizing human action in time-sequential images using hidden Markov model. In *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 379–385. IEEE Computing Society Press.

[126] Tom Yeh, Tsung-Hsiang Chang, and Robert C. Miller. Sikuli. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology - UIST '09*, page 183, New York, New York, USA, October 2009. ACM Press.

[127] Shumin Zhai and Paul Milgram. Quantifying coordination in multiple DOF movement and its application to evaluating 6 DOF input devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '98*, pages 320–327, New York, New York, USA, January 1998. ACM Press.

[128] Shengdong Zhao, Maneesh Agrawala, and Ken Hinckley. Zone and polygon menus: using relative position to increase the breadth of multi-stroke marking menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 1077–1086, New York, NY, USA, 2006. ACM.

[129] Shengdong Zhao and Ravin Balakrishnan. Simple vs. compound mark hierarchical marking menus. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, UIST '04, pages 33–42, New York, NY, USA, 2004. ACM.