Investigating Selection above a Multitouch Surface

by

Dmitry Pyryeskin

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2012

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Above-surface interaction is a new and exciting topic in the field of human-computer interaction (HCI). It focuses on the design and evaluation of systems that humans can operate by moving their hands in the space above or in front of interactive displays. While many technologies emerge that make such systems possible, much research is still needed to make this interaction as natural and effortless as possible. First this thesis presents a set of guidelines for designing above-surface interactions, a collection of widgets that were designed based on these guidelines, and a system that can approximate the height of hands above a diffused surface illumination (DSI) device without any additional sensors. Then the thesis focuses on interaction techniques for activating graphical widgets located in this above-surface space. Finally, it presents a pair of studies that were conducted to investigate item selection in the space above a multitouch surface. The first study was conducted to elicit a set of gestures for above-table widget activation from a group of users. Several gestures were proposed by the designers to be compared with the user-generated gestures. The follow-up study was conducted to evaluate and compare these gestures based on their performance. The findings of these studies showed that there was no clear agreement on what gestures should be used to select objects in mid-air, and that performance was better when using gestures that were chosen less frequently, but predicted to be better by the designers, as opposed to those most frequently suggested by participants.

Acknowledgements

First and foremost I would like to thank my supervisors Jesse Hoey and Mark Hancock, for their invaluable guidance and advice throughout the progress of my research. They helped me become a better researcher, encouraged my ideas, and helped me implement them while keeping me on track in the same time.

I would also like to thank my friends and family for support and encouragement. Thank you to Julia, my best friend, for always being there for me during the years I was away working on my research. Maran, thank you for your ideas and interest in my research.

Finally, I would like to thank my colleagues in Touchlab for their contributions and also everyone who participated in my studies.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Terms

ANOVA ..... Analysis of Variance

DOF ..... Degrees of Freedom

DSI ..... Diffused Surface Illumination

FTIR ..... Frustrated Total Internal Reflection

GUI ..... Graphical User Interface

HCI ..... Human-Computer Interaction

M ..... Mean

SD ..... Standard Deviation

SFS ..... Shape from Shading

WIMP ..... Windows, Icons, Menus, Pointers

Hoverspace ..... Space above or in front of a multitouch surface in which interactions may take place

Multitouch ..... The ability of a sensing device to recognize one or more points of contact with its surface

Chapter 1
Introduction

Surface computers interact with humans by projecting a graphical interface on a surface of an ordinary object such as a table or a wall. By using a touch detection technique most surface computers allow users to manipulate objects in a direct fashion, using their palms and fingers, and without the help of a keyboard or a mouse. This direct nature of interaction is similar to how the objects can be manipulated in the real world, and is considered to be natural and intuitive. Current display and sensing technologies allow building surface computers of large sizes and enable such computers to sense input from multiple users. Large size and ability to support multiple users make these surface computers ideal for collaboration. Groups of people can use large interactive tables to share and edit documents, organize and view pictures or do any other collaborative tasks. Lately, such devices are starting to appear in bars[1], hotels[2], corporate meeting rooms, museums and other public places.

Nowadays many new techniques are emerging that are aimed at extending human-computer interactions into three-dimensional space directly above or in front of a multitouch surface. Such techniques allow users to communicate with computers by performing hand gestures in the air and even by using their whole bodies as input devices. Above-surface interaction is a very promising area that in the near future will transform the way people communicate with computers and will enrich user experience by making it more diverse and natural. While this design space is promising and full of opportunities, one of the most compelling aspects of direct touch interaction is the clear and understandable way in which on-screen targets can be selected—by touching them with your hands or fingers.

[1] http://www.microsoft.com/en-us/news/press/2008/jun08/06-11HETSurfacePR.aspx
[2] http://www.microsoft.com/en-us/news/press/2008/aug08/08-13SheratonMSSurfacePR.aspx

However, this physicality is lost when a person interacts in the space above a surface, due to the lack of reference points when interacting in mid-air; also it is no longer clear how digital artifacts can and should be selected. Since the devices that support in-air interactions are still rare, not many people have been exposed to them. This makes it difficult to know which techniques people will expect to be able to use to interact with such devices. Will people expect to be able to grab objects in mid-air, point at objects from a distance, or will they understand the need to dwell over a 3D target to select it (for example)?

Understanding what expectations people have when they interact with these systems is extremely important for the system designers. This knowledge is necessary for the designers to develop better and more natural interaction techniques. The insight into people's mental models of the interaction techniques allows the designers to create controlled, understandable and pleasing user experiences. The research area of above-surface interaction is still very new and much research is still needed to learn these aforementioned expectations and mental models. This thesis advances the field of above-surface interaction by studying people's expectation and abilities when they select items above a multitouch surface.

This thesis explores interaction in *hoverspace* (the space above or in front of a multitouch display) by focusing specifically on item selection in the space above a multitouch surface. First, this thesis presents a set of guidelines for designing above-surface interactions and a collection of four simple widgets that were designed based on these guidelines. Then it presents the design of a system that can approximate the height of hands above a diffused surface illumination (DSI) surface computing device. Finally, the thesis presents a pair of studies that were conducted to first elicit what gestures people expect to be able to use to select on-screen targets in hoverspace and then explore the performance of several gestures chosen from the first study compared to several designer-created gestures.

The results show that not only do people disagree about how to select objects in this space, but also that the less-frequently chosen designs that the designers predicted to perform better, in most cases did, when compared to the most frequently chosen gestures from the first study.

1.1 Background

1.1.1 Multitouch Technology

While multitouch techniques became commercially viable rather recently, some of them were being researched since as far back as 1965 [18]. Multitouch technology has gone a long way since then, slowly becoming more reliable, accurate and commonplace. Recent innovation in sensing technology [10] has led to development of affordable large-scale multitouch devices such as multitouch tables and interactive walls. Nowadays, devices equipped with multitouch screens are becoming ubiquitous on phones and tablets, and are being researched heavily on larger surfaces, such as tables and walls. This shift provides the potential for direct interaction with on-screen objects in a fashion familiar from the physical world [1,13,34].

Recent technology, such as the Microsoft Kinect, has reduced the cost of the possibility of extending this physical interaction into hoverspace. Currently, there are many methods that can be used to estimate the height of a palm above a surface: stereo cameras [20,35], depth cameras (like Kinect) [3,33], multiple layers of lasers [28], infrared emitters above a table [5] and so on. Such methods allow detection, recognition and tracking of human hands and fingers in the space above a multitouch surface. This information allows system designers to create interactions that are not bound to the surface, but rather can be performed in mid-air. By adding a third dimension to the interaction space, designers are able to create a wider

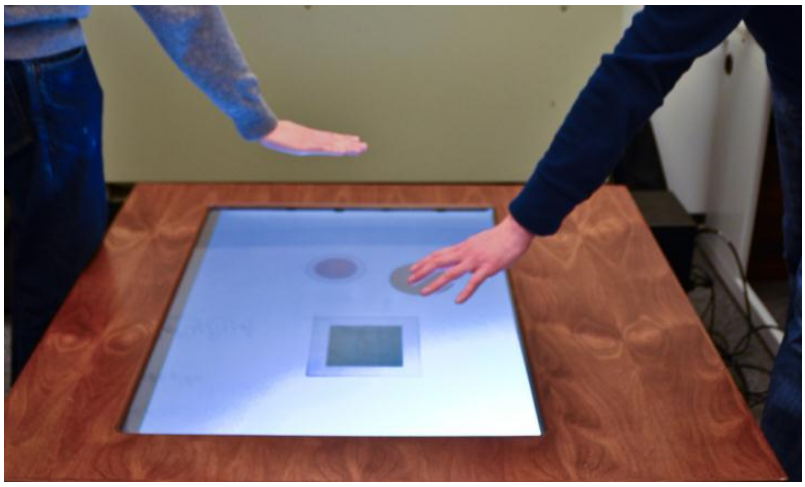


Figure 1.1. People interacting with a multitouch table in hoverspace

variety of gestures that may be more natural for users to perform than surface-based gestures. The addition of hoverspace input to touch input can provide another mode of interaction, while allowing smooth transitions from one mode to another [21]. This added dimension in the interaction space can be used for a variety of purposes, for instance to manipulate 3D artifacts [16], to provide shortcuts to applications via Hover Widgets [8], or to create occlusion-aware interfaces [30].

1.1.2 Multitouch Interaction

Nowadays, computing devices are ubiquitous, come in all shapes and sizes and most of these devices use some sort of a user interface to interact with humans. Currently, one of the most popular styles of user interface is WIMP, which is an abbreviation of the main components used in the interaction: Windows, Icons, Menus and Pointers. WIMP style of interaction was developed at Xerox PARC in 1973 and was made popular by Apple's Macintosh in 1984. Multitouch interfaces may be quite different from WIMP-style interfaces and they usually allow a user to interact with their components in richer and more natural ways: by using touches and gestures.

Item selection or activation is one of the foundations of human-computer interaction: we select menu items to indicate our choice; we activate buttons to enter commands into the system; and we select files, movies or images on a regular basis when interacting with computers. When people use computers with WIMP-style interfaces, they may use hardware buttons on their pointing devices (such as a computer mouse) to signal selection. Multitouch devices, such as cell phones and tablets, free users from the need for a pointing device. These devices use their touch-sensitive screens to let users select and manipulate interface elements directly by touching them with one or more fingers. When interactions are extended into hoverspace and people interact by moving hands above a surface, selection is no longer obvious: there are no buttons to push or surfaces to touch (see Figure 1.1). Targeting items is also a problem, because it is difficult to display information in 3D space directly, and no cost-effective solutions currently exist.

1.2 Research Questions

The problems described above form the main questions that are tackled by this thesis. They can be divided into smaller and more focused research questions:

1. Which techniques do people expect to be able to use when selecting items above a multitouch surface?
    a. Is there an agreement on which technique to use between various people?
    b. Do different people use similar techniques to select similar items?
    c. Do people use different techniques to select different items?
    d. Which factors affect the choice and preference of selection techniques?
2. Which techniques are easier to perform?
    a. Which techniques are faster than others?

b. Which techniques are more reliable (result in fewer errors and false activations)?
c. Which techniques do people find easy to perform?

1.3 Contributions

The main contributions of this thesis are: insight into people's preferences when they select items above a multitouch table, study of performance of six gestures for above-surface item selection, and a set of recommendations for system developers designing above-surface interactions. Another contribution is a method for detection of hands and height estimation in the space above a DSI surface that does not require additional sensors, such as a motion tracking system. And the final contribution to the area of above-surface interactions is a set of basic widgets designed specifically for hoverspace. These widgets are designed according a set of constraints and considerations (which are based on related work) specific for above-surface interactions (described in Chapter 3.1).

1.4 Motivation

Currently, little work has explored what gestures people expect to be able to use to select targets above a table. There is also no clear agreement on which technique should be used to select items in this hoverspace. These are the main factors that motivated the work within this thesis. The choice of study methodology was motivated by the belief that users should be included in the early stages of interaction design and that it is important for the system designers to understand what expectations people have when they interact with these systems.

1.5 Methodology

The methodology used in this thesis is similar to the approach used by Wobbrock et al. [38] to elicit surface-based gestures from participants. In the gesture elicitation study, the participants were asked to come up with any gesture they thought was suitable for above-surface selection, without any regard to the potential limitations of software and hardware used for recognition of these gestures. To minimize the influence of these limitations on peoples' expectations, I recruited participants with no prior exposure to surface computers and no knowledge of the sensing technology used in these devices. The follow-up experiment measured and compared the performance of some of the user-suggested gestures and several gestures designed by researchers. The same methodology was successfully used by other researchers to develop gesture sets for other domains [6,15].

1.6 Thesis Outline

The next chapter of this thesis presents a review of literature related to the four areas of the field of Human-Computer Interaction that are relevant to the topic of the thesis: detection of movement above a surface, interactions above a surface, studying gestures, and the area of in-air target selection. The main purpose of this chapter is to briefly introduce the reader to the disciplines surrounding this research, to describe the current state of affairs in these disciplines, to ground the assumptions and decisions made in the studies, and to define the scope and position of this thesis with regard to the related literature.

The third chapter builds on the ideas extracted from the related work and describes the main pillars that form the foundation of the experimental part of this research: design constraints and considerations for above-surface interactions and a set of above-surface widgets.

The fourth chapter presents the system that I developed for above-surface hand detection and describes the height-estimation algorithm used in that system.

The fifth chapter describes the main part of this research: two experiments that were performed to first elicit gestures for above-surface selection and then to evaluate and compare some of these user-proposed gestures and several designer-suggested gestures. A detailed description of the experimental design is presented as well as an in-depth statistical analysis of the results.

In the last chapter I summarize the research presented in this thesis, outline the findings of the studies and present the answers to the research objectives. I also present ideas for future research.

Chapter 2
Related Work

In this chapter, I present a review of previous work relevant to the work presented in this thesis, which is focused on the specific field of *target selection in space above a multitouch surface*. This field is a very specific part of the much wider field of *human-computer interaction*. More precisely, it lies on the intersection of three general subfields of HCI: *gesture-based interaction*, *above-surface interaction* and *target selection*. Above-surface interaction is related to *Interactive Tabletop and Surface Environments*; gesture-based interaction is also related to these environments, but is not exclusive to them; target selection is a part of a wider field of *targeting in GUIs*, which also includes *target acquisition*. Figure 2.1 shows a graphical representation of these fields. The various methods of detecting and tracking of human hands and fingers above or in front of an interactive surface are also important to this work and are reviewed in this chapter as well.

The related work therefore is divided into four primary categories: *above-surface*

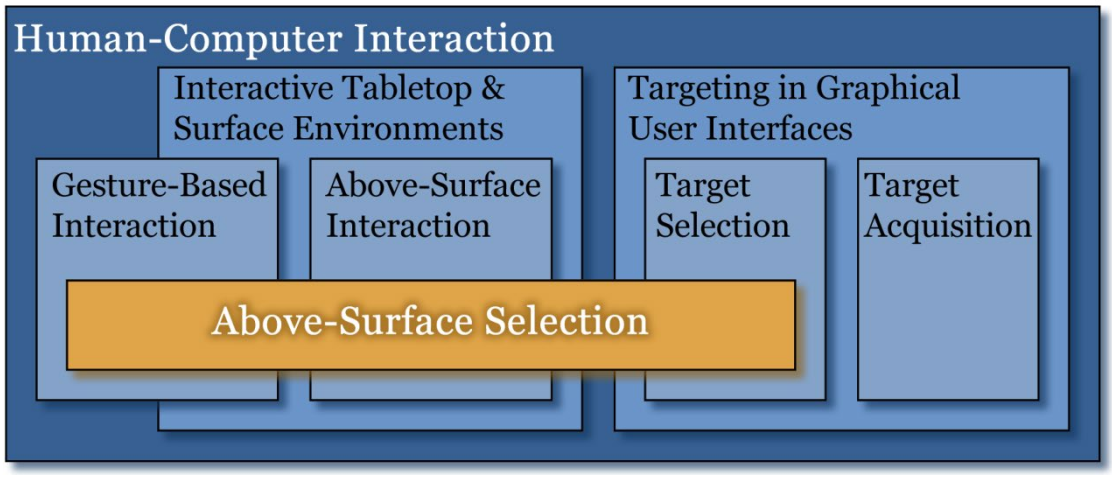


Figure 2.1. The context of this research

movement detection, *interaction above a surface*, *the methods for studying gestures*, and the area of *target selection*. First a summary of hardware and software techniques developed to detect a moving hand above or in front of an interactive surface is presented. This is followed by a review of other literature related to research of interaction above or in front of an interactive surface. Then I discuss methods used by other researchers to study gesture-based interactions. Finally, an overview of literature relevant to target selection is presented.

2.1 Detection of Movement above a Surface

The subject of in-air hand detection has already been researched for some time and several types of hand tracking systems have been proposed. These systems use a variety of techniques to detect the position of hands in 3D space; some of them are also able to recognize the hand posture and positions of individual fingers.

Traditional marker-based systems use expensive arrays of cameras with overlapping fields of view to track the positions of retro-reflective markers or LEDs attached to various parts of a hand (or even body). A number of studies [9,29] used a Vicon[3] motion tracking system to detect the position of participants' hands, fingers and even individual phalanges. Vicon systems are able to uniquely recognize each marker and track them in 3D space with remarkable precision (sub-millimetre 3D coordinates at frequencies up to 120 Hz). However, this type of system can be obtrusive, fairly expensive and it can take time to properly place or wear the reflective markers. While my system cannot compete with a high-end motion-capture system in terms of accuracy, it is simpler, less expensive, and requires only a single camera.

[3] http://www.vicon.com/

Instrumented glove systems such as CyberGlove[4] are also able to precisely capture the 3D position of hands and fingers in real time. However such systems are expensive and can be awkward, because they rely on exoskeletons or embed multiple sensors into a glove, which can be restrictive to movement of the hand.

An alternative approach proposed by Wang and Popović [31] uses a single camera to track a hand wearing an ordinary cloth glove that is imprinted with a custom pattern. It is simpler and cheaper than marker-based systems and the instrumented gloves described above, but it provides less accurate tracking compared to those systems. Another drawback of this system is high processing power requirements, since the system searches a database of pre-recorded hand postures to recognize the posture in each frame; the interactive demos described in their paper ran at 10 Hz on a quad-core CPU.

Recent advances in technology have enabled the development of robust and precise *device-free* tracking systems that do not require users to put anything on their hands or hold any wired or wireless devices to enable hand-tracking capabilities. Some of these systems rely on multiple stereo cameras to calculate the 3D location of a person's hands and fingers [20,35]. Such systems require much more computational power than other vision-based systems and also require very precise camera calibration. Other approaches rely on depth cameras for tracking [3,33], such as that found in the Microsoft Kinect. Z-Touch uses multiple layers of lasers to estimate the position of a person's hands and fingers [28], Shadow Tracking uses infrared emitters located above the surface to create and track shadows of people's hands together with touch regions [5], SecondLight relies on a switchable diffusing material to detect objects above a surface [17], and another approach proposed by Wilson uses a Kinect camera to detect touches on non-instrumented surfaces [37].

[4] http://www.cyberglovesystems.com/

Device-free methods are much more suitable for spontaneous, natural and unimpeded interactions with interactive surfaces and tables, unlike methods that use gloves or handheld devices with buttons for a number of reasons:

- Devices, such as gloves, wands or reflective markers, require some time to put on and/or learn how to operate
- The number of available hand-held devices places a limit on the number of users that can interact with the surface simultaneously
- Glove-based hand tracking methods carry some assumptions about the position of the user's hand and fingers with respect to the tracker and often require recalibration for every new user
- Simple handheld devices (such as a wand with buttons) are actually preferred by users to gloves [25]
- Any device held in the hand can become awkward while gesturing and may significantly limit the variety of gestures that a user can perform

Due to these and other reasons I have developed a device-free technique for my studies. An extensive selection of device-free methods for detecting and tracking hands and fingers above a surface is discussed below in greater detail.

2.1.1 Multiple cameras

Some of the above-surface hand tracking methods use multiple stereo cameras to calculate the 3D location of a person's hands and fingers [20,35]. These systems require much more computational power than other vision-based systems and also require very precise camera calibration.

Perceptive Workbench [20], for example, uses 9 ceiling-mounted infrared illuminators, 2 under-table near-infrared light sources, an above-table camera, a side camera and an under-

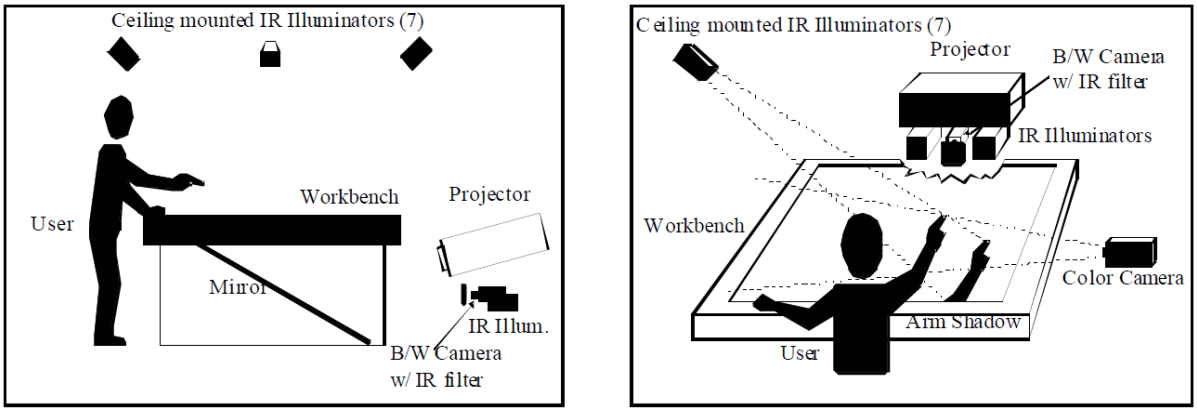


Figure 2.2. Perceptive Workbench, light and camera positions [5]

table camera to recognize and track objects (see Figure 2.2). It relies on a complex algorithm to reconstruct 3D shapes of objects based on shadows produced by multiple light sources. The reconstructed objects are then placed into the virtual world, where they may be manipulated. The system also provides some hand recognition: it detects the user's hand position, and the direction in which the user is pointing. The complexity of this multi-camera set up, however, presents a number of challenges: difficulty of finding a good shadow to reliably track an arm, occlusion of arms in the field of view of the side camera (which is a serious problem for multi-user applications), and the need for extremely precise calibration of camera and light source locations for accurate tracking.

TouchLight [35] is another example of a multi-camera set up. It uses image processing techniques to track users' hands by combining the output of two video cameras placed behind a semi-transparent plane set in front of a user, which also serves as a rear-projected display. Depth information is computed by calculating *binocular disparity* [32], the difference in two images of the same object perceived by two cameras set a certain distance apart (which is similar to the way a human brain calculates depth information based on the

[5] © Bastian Leibe, used with permission

differences of images perceived by left and right eyes). To make this "depth from stereo" algorithm used in the system less computationally intensive, the author simplified it by restricting it to detecting objects located on a particular plane in three dimensions (the display surface) rather than the depth of everything in the scene. Therefore, while this system can theoretically be used to track hands in the space in front of the display as well, it would require much more processing power and much more precise camera calibration. On an unrelated note, such hand tracking systems with transparent displays were successfully used for dramatic effect in such movies as *The Matrix Reloaded* and *Minority Report*.

My approach uses only a single camera, which makes it much simpler, cheaper and less computationally intensive than multi-camera setups.

2.1.2 Depth cameras

DepthTouch [3] and LightSpace [33] detect and track hands using one or more depth cameras, such as that found in the Microsoft Kinect. Such cameras are able to detect both color and depth information for each pixel. They used to be fairly expensive prior to the release of Microsoft Kinect.

The DepthTouch [3] system shares many configuration similarities with the TouchLight prototype described above. It also consists of a semi-transparent rear-projected screen, but instead of two stereo cameras, a single depth camera (ZSense) is mounted behind the screen to track the position of users' hands. The camera's position behind the screen minimizes situations in which one hand occludes the other and allows for tracking of the user's hands by segmenting the depth information computed by the camera. The tracking algorithm segments the depth data, first discarding areas that are too close or too far from the screen and then separating the depth image of the user's torso from the image of the hands. The system suffers from several limitations, such as noisiness of the depth image, which requires

smoothing that produces some lag; this noise also makes it difficult to track small extruded points, such as fingers. Also, this system is unable to distinguish between hands that are close together, close to the user's body, or not in front of the body (e.g., to the side).

LightSpace [33] is a small room installation equipped with 3 depth cameras (PrimeSense) and 3 projectors calibrated to real world 3D coordinates. The system is able to detect objects and users and correctly project graphics onto any surface. After a calibration, the cameras capture a 3D mesh of the sensed portion of the space in real-time. Projectors are calibrated as well, and when an object is placed into the virtual world, it can be correctly projected onto any part of the calculated 3D mesh. Instead of using sophisticated resource-intensive algorithms, like skeletal tracking, this system relies on simplified methods of tracking users' hands in spaces above surfaces. In the particular set up described in the paper there was a volume with the height of 10 cm defined above the table located in the installation and another volume with the depth of 10 cm located in front of a portion of a wall. Users were able to interact with virtual objects, by moving their hands in one or both of these volumes. Virtual objects, appearing on one of those interactive surfaces, could be "picked up" or "dropped" by the users and transferred to another location or even another surface.

The drawbacks of the LightSpace system include high demands for processing power (only up to 2-3 users could be supported without slowing down the system below 30 Hz), occlusion of users' hands by their heads (since the cameras are located above the table), and a noticeable lag of the system during fast hand movements. Another limitation is that the depth cameras used in the system are not precise enough to track fingers, which limits the way users can interact with the system.

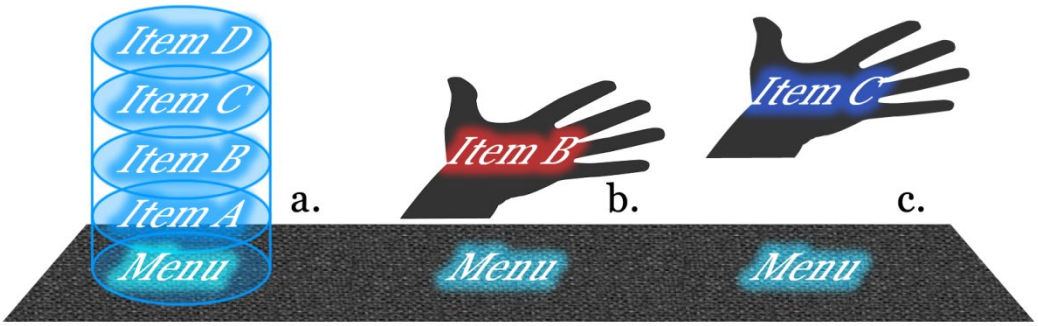


Figure 2.3. Spatial menu used in LightSpace installation: a. virtual menu items are placed at different heights, b. and c. when a hand is held at certain heights the corresponding items are projected onto it

A technique called "Spatial Menu" deserves a special mention: it is an interesting interaction technique proposed by the authors of the LightSpace paper. The menu can be visualized as a long and narrow virtual column located at a certain spot (marked by a special marker on the floor). Menu items are placed at different heights above the menu and when a user moves their hand through the menu, the items are projected onto the hand (Figure 2.3). The selection is triggered by holding the hand in place for 2 seconds. This menu was the inspiration for the Menu widget used in my studies (see sub-section 3.2.2.2).

Unlike both approaches described above, my system is able to precisely detect locations of fingers touching the surface. It is also considerably simpler and cheaper than the LightSpace prototype. My algorithm is also less computationally intensive.

2.1.3 Other hardware

A few other innovative methods were also proposed to detect hands and fingers in the space above an interactive surface: Z-Touch uses multiple layers of lasers to estimate the position of a person's hands and fingers [28], Shadow Tracking positions infrared emitters above the table to create and track shadows of people's hands together with touch regions

[5], SecondLight relies on a switchable diffusing material to detect objects above a surface [17], and another approach uses a Kinect camera to detect touches on any surface [37].

Z-Touch [28] is a multitouch table infrastructure that enables 3D gesture interaction above tabletop surfaces. The hardware system is composed of 3 layers of infrared laser planes placed above a rear-projected glass surface and a high-speed camera (Point Grey Grasshopper, which can capture 8-bit gray-scale VGA images at 200 fps) placed under the surface. The laser layers are switched on and off alternatingly creating planes of laser light parallel to the screen's surface. When an object, such a finger, intersects a laser light plane, the light is scattered and detected by the under-table camera, which is synchronized with the laser planes. The resulting bright blobs are used to calculate the position and angle of fingers with great accuracy. The limitations of such system are its need for extremely precise calibration of the lasers, cumbersome apparatus, fairly small height of the hoverspace (approximately 4 cm), problems with occlusion, and difficulties with matching blobs produced by multiple fingers. Since the vertical position of an object is estimated at only 3 levels, there is not enough information for accurate detection of vertical speed and acceleration.

The Shadow Tracking [5] system enhances a multitouch table based on FTIR technology [10], which uses a regular camera to detect bright blobs of infrared light created by fingers touching the surface, which is flooded with infrared light based on the total internal reflection effect. To enable the table to sense hands hovering above the surface, an array of infrared light sources is placed above the table so that the objects hovering above the table would cast shadows onto the surface. The table's built-in camera is able to detect these shadows. The light sources inside and above the table switch on and off alternatingly to create separate hover and touch images to be captured by the camera. A computer vision algorithm used in this system segments these hover and touch images and assigns touch

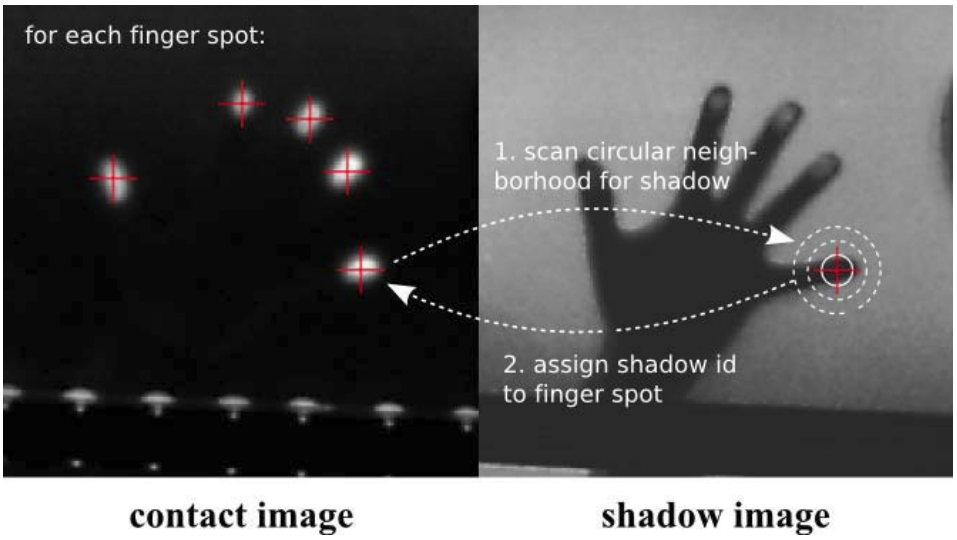


Figure 2.4. A contact image and a shadow image used in Shadow Tracking system [6]

regions to appropriate shadows (see Figure 2.4). This method is able to detect objects hovering above the table's surface, but it is unable to estimate the height of these objects. The complexity of the original hardware was also increased with the additional ceiling-mounted light sources and a control circuit.

The SecondLight [17] system also enhances an FTIR-based multitouch table. It inherits the benefits of a rear-projected multitouch system and enables interactions to be extended into the space far above its surface. The main difference from all other systems is the use of a special rear-projection screen material that can be rapidly switched between transparent and diffuse states using electrical signals. While the surface is in its diffuse state, the SecondLight system can display digital content on the surface and sense hands and fingers touching the surface (using FTIR technology [10]), and when in its transparent state, the system can project through the surface and detect objects above it. The state of the surface is switched at high frequency so that the human eye cannot detect flickering. Two projectors are placed

[6] © Florian Echtler, used with permission

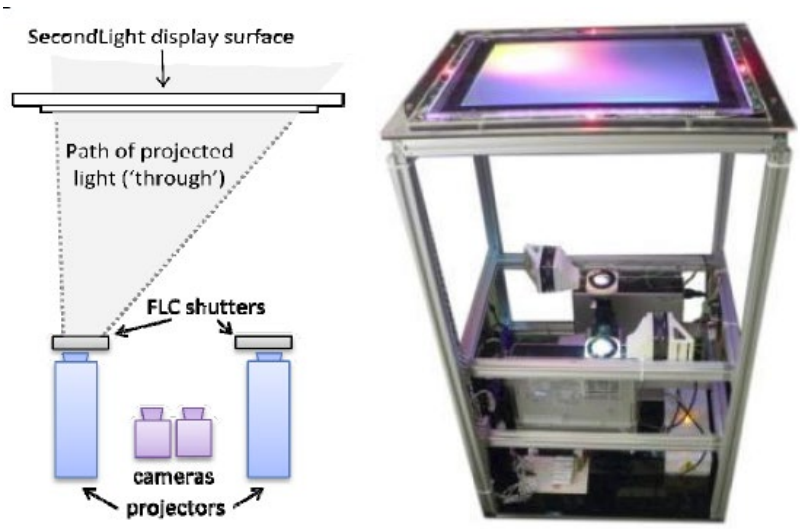


Figure 2.5. The layout of the main components of SecondLight [7]

below the surface and equipped with digital shutters. The system uses two cameras to sense on-screen and above-screen interaction. A dedicated circuit is used to synchronize the state of the surface, cameras and projectors (see Figure 2.5). SecondLight's main benefits are: accurate recognition and tracking of touches, the ability to see objects clearly through the surface and the ability to project through the surface. The system is however unable to estimate the height of the objects it detects above the surface, but the authors suggest that this task can be done with the use of depth cameras.

Another interesting method of hand tracking uses a depth camera as a touch sensor to turn any surface (even non-flat) into a multitouch device [37]. A Microsoft Kinect camera mounted above a tabletop surface detects hands and fingers and uses the depth data to calculate regions where fingers were touching the surface. While the system can potentially transform any non-instrumented surface into a multitouch device, it suffers from occlusion (a hand may occlude a finger, or a user's head may occlude a hand) and relies on a lot of assumptions to detect the touch regions (thickness of a finger, shape of the surface).

[7] © Shahram Izadi, used with permission

However, even though the touch-recognition capacity of such a system is less than that of more conventional systems, it offers interesting opportunities, such as using the depth information to track hands above the surface.

My approach is simpler and cheaper compared to all systems described above (with the possible exception of the last approach). It is able to estimate the height of a hand with far greater resolution and at a greater distance than Z-touch; the Shadow Tracking and SecondLight systems are unable to estimate the height at all; and the Kinect-based method is not as precise at detecting touches as my system.

2.1.4 Contributions

Most of the methods described above require custom-built hardware or modifications of existing multitouch tables. I add to this work by demonstrating how to use an existing DSI setup to track hands above a surface. My approach is purely software-based; it requires no additional hardware, sensors or modifications to the standard DSI surface. This approach can be considered a special case of the more general shape from shading (SFS) problem. SFS techniques compute the shape of a surface using a single greyscale image as input [23]. The computed surface may be used to estimate the height of a hand above the surface and for gesture recognition. The algorithm used in my software is not as resource-intensive as algorithms required by methods based on multiple cameras or depth-cameras. Due to the under-table position of the camera used in the DSI setup, the hand occlusion problem is not as severe as in approaches that place cameras on the side or above the surface. The DSI set up is also inexpensive[8] and requires no costly equipment, such as high-speed cameras or multiple depth cameras. My approach is robust enough to support multiple users and up to

[8] The approximate cost of my experimental setup was US$200 (for acrylic and delivery) + US$200 (for high-end rear-projection film and delivery) + US$100 (for camera). The setup also required a projector and a PC.

30 simultaneous finger touches. It is able to estimate the height of human palms above the surface using computer vision methods and linear regression (see Chapter 4 for more details).

2.2 Interaction above a Surface

Emergence of a variety of in-air tracking techniques led to the development of a great number of interesting techniques designed for interaction above a multitouch surface. In this section I present a review of relevant literature.

Hover Widgets [8] introduces the possibility of using the space above a surface to create a new command layer that is clearly distinct from the input layer on the surface. The authors of the paper used a pen-based device (a Table PC) that supports a *tracking state*: it senses the location of the pen when it is slightly above the surface. Users are able to use the pen device as usual when it is in contact with the Tablet PC's screen (for writing or drawing), but when the pen is held slightly above the screen, it can be used to invoke certain commands by following 'L'-shaped paths. Studies show that users were able to use Hover Widgets successfully to replace some interface elements, like toolbars and menus. Furthermore, Hover Widgets reduced movement time and improved accuracy, when compared to a standard toolbar icon. I have adapted the idea of creating a distinct command layer to my research. In the demo painting application described in Appendix 3 the surface of the table acts as a canvas and new strokes are added to it every time it is touched by a finger. The hoverspace is used as a command layer and users may navigate its graphical elements freely without adding any unwanted strokes to the canvas.

Hoverspace can also be used to create techniques for more natural manipulation of 3D artifacts on a multitouch surface [16]. Interactions with regular multitouch surfaces are restricted to the 2-dimentional surface of the display, which works well when directly

controlling 2D objects with 3 degrees of freedom (DOF): rotation and translation in two dimensions. However, this directness breaks down when manipulating 3D objects. Hancock et al. [12] proposed using more fingers (up to 3) to make manipulation of 3D objects with 6 DOF of movement and rotation as natural as possible. Using hoverspace data, however, I may allow users to simply lift their hands above the surface to lift an object, as described by Hilliges et al. [16]. While I did not use 3D objects in my studies, I designed a Menu widget—a 3-dimentional stack of 2D items. The stack may be browsed by moving a hand up and down above it similar to the Spatial Menu used in the LightSpace system [33].

Han and Park [11] developed a zooming interaction that uses above-surface tracking capabilities. The amount of zoom in the proposed interaction is controlled by moving one's hand up and down above the interactive surface. The study showed that hover-based zooming significantly outperformed the conventional two-finger zooming in speed, and was also preferred by the participants of the study. In the scope of my research, this finding suggests that widgets that can be controlled using the height of a hand may be faster and easier to use than widgets that rely on multi-finger interaction on the surface.

My work expands this space and explores more specifically the target selection aspect of above-surface interaction techniques, once a widget has been acquired.

2.3 Studying Gestures

Gestures can be a rich, versatile, and natural way for people to interact with each other and with technology. Both on-screen and above-surface gesture-based interaction have been studied in depth, and a review of relevant literature is presented in this section.

2.3.1 Designer-driven Approach

A common approach to evaluating these gestures is to first design them, based on experience or related work, and then evaluate their performance when compared to other alternatives. Grossman and Balakrishnan designed and evaluated 3D selection techniques for volumetric displays [7]. They proposed 4 different enhancements to a pre-existing 3D selection technique called "ray cursor" [7], then these alternatives were quantitatively evaluated in an experiment and the most successful (fastest and most accurate) one was identified. Marquardt et al. proposed the idea of unifying the touch interactions and hoverspace interactions into a single continuous interaction space [21]. They proposed a wide range of gestures and techniques that may merge on-surface and above-surface modalities of interaction into a seamless interaction space. Spindler et al. [26] explored interaction techniques that make use of a layered space above a multitouch table. They used a tangible magic lens system, which is able to track a small handheld display in the air above an interactive surface and project images onto it. Users are able to explore the virtual world by holding and moving the lens in hoverspace. The paper describes a study performed to determine the accuracy at which basic tasks (such as holding and searching) can be performed using a magic lens and provides design recommendations on the use of layers in above-surface interactions.

2.3.2 User-elicited Approach

An alternative approach is to first elicit what gestures people expect to correspond to a given action, rather than have the system designer determine what gesture is best-suited to an action. Nielsen et al. [22] argue that a human-centered approach in the development of gestures yields better and more natural results than a traditional technology-based approach. Wobbrock et al. [38] support this statement and successfully employ user-

centered gesture development methodology to develop a versatile gesture set for tabletop interactions. Both works point out that the surface gestures designed by system designers may be influenced by the concern for reliable recognition and may result in arbitrary gesture sets that are awkward to use. The same methodology has been used by other researchers as well. Most notably, Frisch et al. [6] investigated user-centered gestures for diagram editing on interactive surfaces. They asked participants of the study to perform spontaneous gestures for a set of tasks. As a result, they developed a user-defined gesture set and were able to get some insight into users' mental models related to performing certain tasks on tabletop computers. Henze et al. [15] have successfully applied the same methodology to a different domain: controlling music playback.

My work uses this same approach by Wobbrock et al. [38] to elicit gestures from participants, who are not familiar with the way my system is able to recognize the gestures and therefore are not constrained in their interaction.

2.4 In-air Target Selection

Device-free interaction makes signaling selection non-trivial: there are no buttons to press when interacting with a system in mid-air. The most common solution to this problem is the use of gestures, which are a rich, natural and versatile method of interaction between humans. A number of system designers have used various gestures in their hoverspace-enabled systems. In this section I review literature describing some of these gestures.

One of the most basic solutions that has been adopted by many eye-tracking systems (such as one proposed by Hansen et al. [14]) is to use a dwell time threshold to indicate selection. This approach is simple to implement, but it suffers from a problem called "The Midas Touch Effect", which means that this selection method may result in a large number of unwanted selections. An unwanted selection may happen, for example, when a user gets

distracted and stops moving for a period longer than the dwell time. In an eye tracking system, this means the user must constantly shift their gaze, otherwise if their gaze is held on a certain interface element for a fixed amount of time the system will interpret it as a selection. Using dwell time for selection also introduces some latency into the system, which can be noticeable when a large number of selection events are required.

A variety of other gestures were proposed for hoverspace target selection, some of them are listed here. Wilson [36] designed a simple computer vision technique that enables reliable recognition of the hole formed when an index finger touches the thumb. Hilliges et al. [16] used this technique to detect *pinch gestures* in their system. Banerjee et al. [2] designed the *SideTrigger* gesture: to acquire targets a user points with the dominant hand's index finger while the other fingers are curled towards the palm and to signal selection the user brings the thumb close to the curled middle finger. Grossman et al. [9] used the *Thumb Trigger* gesture for item selection in their 3D volumetric display system, which is similar to the SideTrigger gesture described above, but the user brings the thumb to the extended index finger instead of middle finger to signal selection. Vogel and Balakrishnan designed the *AirTap* selection gestures for their remote pointing system [29]: the technique is similar to the downward motion of the index finger when clicking a mouse button or tapping a touch screen.

Currently there is a distinct lack of user-designed gestures for target selection in the literature. As Wobbrock et al. [38] put forward, while gestures defined by the system designers certainly produce good results, they may not reflect the expectations of users, and their development may be influenced by concerns for reliable recognition [22]. My work remedies this state of affairs by giving users an opportunity to generate their own gestures.

2.5 Discussion

This literature review shows that there exists a variety of methods that can be used for detection of movement above or in front of a multitouch surface. These methods have various requirements in terms of price, complexity and processing power, they provide designers with various benefits and they also have different drawbacks. I described the strengths and weaknesses of these methods, and also discussed how they compare to the proposed system.

I also presented a review of literature related to designing interaction in front of a surface and discussed several techniques that make use of the extra control dimension provided by hoverspace. From this related work I extracted a set of design constraints and considerations that were used in the design of my own interaction techniques.

I contrasted two general approaches of studying gestures: designer-based and user-centered. I motivated the decision to employ the user-centered methodology proposed by Wobbrock et al. [38] by referring to other work that has successfully applied this same approach.

Finally, I presented a review of literature related to the area of in-air target selection, which shows the lack of attention to the user-designed gestures for this task. This lack of attention is one of the main factors that motivated this research.

The literature review provides a foundation in which the rest of this research is grounded. The ideas described in the following chapter are built upon this foundation and in turn provide theoretical support for the practical part of the research. In particular, the next chapter describes design constraints and considerations specific for above-surface interaction and presents a set of widgets designed specifically to study objects selection in hoverspace.

Chapter 3
Designing Above-surface Techniques

While many different designs for above-surface selection gestures have been suggested, little work has explored people's expectations of what these gestured should be. Over the past decade, there has been a surge in interface development for gesture-based interaction directly on the surface of a screen, but much is still not understood about interaction above a surface. Extending interaction into the 3D space above the surface of a multitouch surface may interfere with some of the familiar paradigms and mental models and introduces a number of interesting challenges in interaction design, visualization design, and ergonomics.

In this chapter, I describe the design considerations and constraints that are unique to hoverspace interaction. Then I demonstrate how to apply these design constraints and considerations by describing the design of two abstract and two practical examples of above-surface widgets.

3.1 Design Constraints and Considerations

When designing interaction for hoverspace, I have considered a number of factors. These considerations are based largely on prior work in above-surface research, but made specific to my primary goals of addressing: non-physical interaction, the use of layers, the transition between above-surface and touch interaction, and issues of fatigue.

3.1.1 Non-Physical Interaction

One of the most important challenges is the lack of reference points in the air. When navigating on a 2D surface, people can interact directly with the interface by touching artifacts such as buttons, menus, and images. In contrast, when a target is in hoverspace, it

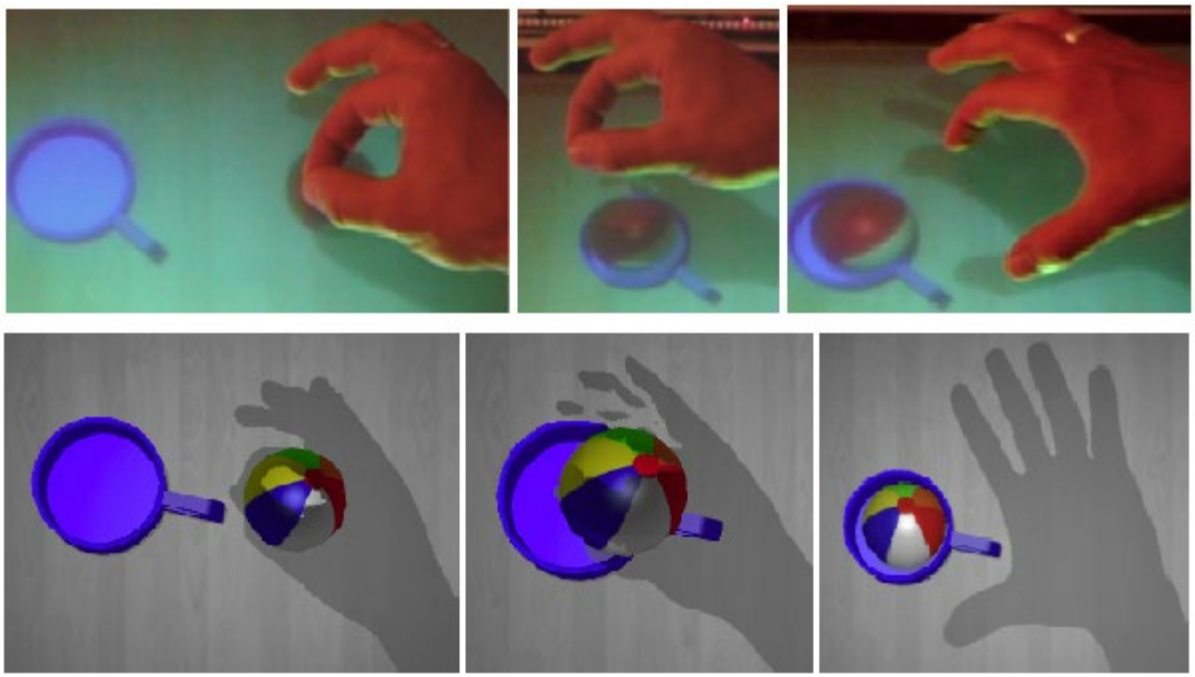


Figure 3.1. Virtual shadows cast by user's hand [9]

cannot be represented in the air (unless some sort of holographic or virtual reality technology is used), therefore the direct interaction paradigm breaks down and the user must rely on a visual representation of his or her hand in hoverspace. Therefore a representation of a user's hand in the 3D space should be included in the system and it has to be consistent and recognizable. For example, Hilliges et al. [16] used virtual shadows to represent users' hands and to integrate them into the virtual 3D world beneath the touch screen (see Figure 3.1). In my system, I use a circular cursor with a variable diameter to represent the location of a hand. The cursor's horizontal location is directly below the centre of the palm and its diameter is proportionate to the height of the palm above the surface (see section 3.3).

[9] © Otmar Hilliges, used with permission

3.1.2 Layers

The lack of reference in the air also makes it difficult to split the hoverspace into multiple layers, since it will not be apparent where these layers are. Still, defining a number of layers in hoverspace can be useful for increasing the dimensionality of control space. For example, a gesture performed near the surface can have a different meaning from a gesture performed farther away. Care should be taken, however, to define and display layers in such a way that their location is reasonably easy to guess or discover. Spindler et al. [26] explored multi-layer interaction in hoverspace and they made a number of recommendations for interaction design, most important of which are: "use as few layers as necessary" and "provide instant feedback". They report another important finding: the participants of the study significantly preferred to search vertically instead of horizontally. This idea is used in the design of my Menu widget, which allows users browse its items by moving their hand vertically above it (see sub-section 3.2.2.2).

3.1.3 Transition between Hover and Touch

I believe that to create the most natural experience, there should be no separation between interactions in hoverspace and on the surface, and no need to switch the modality of interaction. Gestures above the surface should merge seamlessly into touches on the surface and back. Hoverspace interactions are supplementary to touch-based interactions and the way users interact on the surface does not have to change. Hover and touch data should be considered as a unified space, as proposed by Marquardt et al. [21]. All of my widgets as well as the cursor are designed for a seamless transition between hover and touch modalities of interaction.

3.1.4 Fatigue

Fatigue is a known issue with mid-air interactions and its effects should be considered in the design of above-surface gestures and techniques. For example, using dwell time to indicate selection in hoverspace requires users to hold their hand steady for some time and, in tasks when multiple selections are required, this added wait time has the potential to exacerbate fatigue effects and also reduce users' ability to hold their hands steady. Users should not be forced to hold their hands in the air for prolonged periods of time. Instead, hoverspace should be used for short tasks (like item selection or menu invocation) and once the task is done the interactions should naturally return to the surface. Bimanual interaction can help to remedy this issue by limiting the need to adjust the posture in mid-air, for example one hand can be used to navigate to an item and another to indicate the selection. Ergonomics of the surface and the size of the hoverspace are also important factors to consider, since they may impact fatigue. For example, Subramanian et al. have limited the height of the hoverspace to 16 cm for sitting users to reduce fatigue, based on the results of their pilot study [27]. Spindler et al. also tried to reduce the effects of fatigue in their studies by limiting the height of hoverspace to shoulder height for standing users [26].

3.2 Above-surface Widget Design

I designed a set of four different widgets (two abstract and two practical) and a cursor based on the considerations listed above to be suited for above-surface interaction. These widgets are designed to be used with an open palm, and therefore should be large enough to be visible under one's palm during the interaction. An alternative solution to the widget occlusion problem is to detect the size of the occluding palm and adjust the position of the widget so that it is visible to the user.

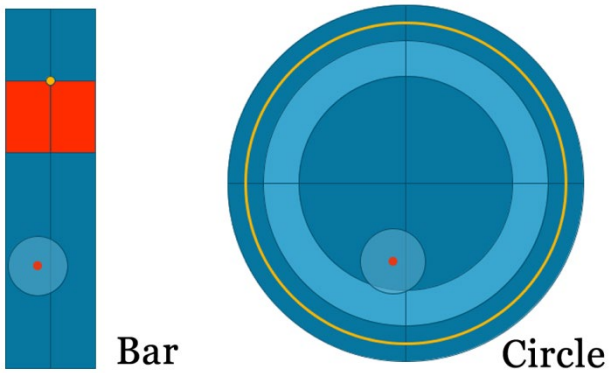


Figure 3.2. Abstract above-surface widgets

3.2.1 Abstract Widgets

The pair of abstract widgets (bar and circle, see Figure 3.2) was made to look as generic as possible; their purpose was to study the effect of widget shape (if any exists) on the users' expectations when interacting with the widgets. Each widget has a target area with a certain width, height and depth that is located above surface. These targets may represent actions (e.g. save, exit), files, images, tools, and so on. Even though I only defined a single target on each widget, for the sake of simplicity in the experiments there may be a larger number of these targets, each in a different location and activated when a user holds a hand at different heights above the surface. A target can be placed anywhere on a widget along the z-axis: from the surface to the furthest reach of hoverspace. This ensures a smooth transition between hoverspace and touch interaction: a user may hold their palm above a widget to reach one target or touch the surface to reach another.

3.2.1.1 Bar

The bar widget (Figure 3.2) is an abstract rectangular control with a small slider moving up and down along the mid-line. The distance of a person's hand above the table determines

the position of this slider along the vertical axis. This slider visualization remedies the problem of non-physical interaction and helps to navigate the hoverspace. There is a “target area” in the main rectangle which turns red when the slider is placed inside. A bar widget can be used to represent lists of items, such as file menu items: open, close, save, save as, etc. It can also represent arrays of objects, such as movies or documents, which can be navigated by moving a hand up and down. In my experiment, for the sake of simplicity, I designed the bar widget to display the position of a hand closest to the surface, therefore only one user can interact with a bar widget at a time. In a more realistic application, an alternative mode of selection for controlling the palm could be used; one of the possible solutions is described in sub-section 3.2.2.2 Menu.

3.2.1.2 Circle

The circle widget (Figure 3.2) is similar to the bar widget in functionality, but shaped differently. The slider is represented as a ring and the “target area” is represented as a differently coloured band, which turns red when the cursor is placed inside. Similar to the bar, circle widget can be used to display lists of items. However, the circular shape is better suited for collaboration, since its orientation will be the same for all users standing around the table. This widget can be used to control the volume of music or zoom level of a map, for example.

3.2.2 Practical Widgets

The second pair of widgets was designed to be more practical and to be adapted for use in more realistic applications. Both widgets have counterparts that are used in WIMP and multitouch interfaces.

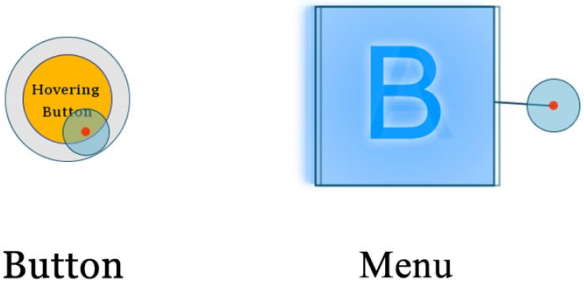


Figure 3.3. Practical above-surface widgets

3.2.2.1 Button

The button widget (Figure 3.3) is an abstract version of a standard circular GUI button that is acquired by moving one's hand to a predetermined 3D position above the table. The circle changes colour when this correct location is acquired to indicate this hover state (similar to mouse-over state in WIMP interfaces). To enhance the guessability of the button's location, its height above the surface is represented by a grey band around the inner circle. The width of this band is the same as the radius of a cursor, when the cursor's height equals to the height of a button.

There are multiple reasons for putting a button into hoverspace instead of keeping it on the surface of the table. For example, the entire surface of a screen may act as canvas and buttons may be placed above it to avoid accidental activation. A button might also be placed in front of a display if it is too far from a user to touch it directly (e.g., while watching TV). An exit button may be placed high above the table to make it inaccessible for small children, but within easy reach of an adult caretaker.

3.2.2.2 Menu

The menu widget (Figure 3.3) is an abstract version of a menu in a GUI, represented by a stack of five squares marked with the letters A to E; the current menu item is controlled by moving a hand up or down (similar to the Spatial menu used in the LightSpace system described above [33]). Items that are located below the current item are blurred to simulate the way a human eye focuses on objects and items located above are made semi-transparent, so that they do not obscure the current item, but are still visible. I experimented with a number of ways to visualize the stack of items, see Appendix 1 for details.

The menu widget uses a special algorithm to determine which cursor should control it. First only the menu's *anchor* is displayed, represented as a circle. When a hand is held above or touches the anchor (the height does not matter, only the horizontal position), the menu is displayed and the cursor is marked by the system as a "control cursor". The system draws a line from the menu to this cursor to indicate it as the "control cursor". Whenever the cursor is moved too far from the menu or is no longer tracked by the system (i.e. when the hand moves out of the tracking range), the system finds all cursors within a certain distance from the menu and assigns the closest one of them as the new "control cursor". Since the cursor that invokes the menu may already be at a certain height, it gives experts a chance to jump directly to an item they are looking for instead of searching through the whole menu. The current item is determined in real time from the height of the controlling cursor using a simple formula:

$$CurrentItem = \lfloor CursorHeight \times 0.9 \times NumberOfItems \rfloor$$

where $CursorHeight$ is a value between 1 (in the highest part of the hoverspace) and 0 (touching the surface) and $NumberOfItems$ is the total number of items in the menu (equal to 5 in my experiments). For example, when a participant holds their hand at a height of 10 cm (approximately equal to .5 of the total height of hoverspace in my system), the current

item is calculated as: $CurrentItem = \lfloor .5 \times 0.9 \times 5 \rfloor = \lfloor 2.25 \rfloor = 2$, which is the third item (count starts from 0). The same control cursor method may be extended to other widgets as well for better support of multi-user environments.

A menu widget can be used to display a large number of items on a small area of a screen. Thumbnails of photos or videos may be used instead of labeled squares to navigate a multimedia gallery, for example. The menu widget virtually splits hoverspace into a number of layers and places an item into each layer. While it may be hard to guess the height of each layer, the menu widget can be easily navigated by moving a hand up and down, and when the desired item is located, a selection action can be performed. As Spindler et al. [26] discovered in their experiments, users prefer vertical search to horizontal search. They also recommend using layers with a depth of at least 1 cm for vertical search and hold tasks. The maximum height of hoverspace in my system is approximately 20 cm, therefore when I split the stack into 5 layers each layer is approximately 4 cm high. This configuration should make it easy for participants to search and target items in the menu widgets. Using layers of sufficient height also reduces the chance of frustration when trying to stay within one layer and drifting inadvertently into the layers above or below.

3.3 Hand Representation

To rectify the lack of reference points during in-air interaction, a user must be provided with a visualization of their hands in hoverspace (see design consideration is sub-section 3.1.1). I designed a special cursor to visualize the location of a hand in 3D space. A separate cursor is displayed for each hovering object and each touch region. The cursor is represented as a semitransparent circle with its centre directly below the geometric centre of a hovering palm and its size proportionate to the height of the palm above the surface (see Figure 3.4). The cursor has a red circle of a constant size in the centre, so when a user touches the

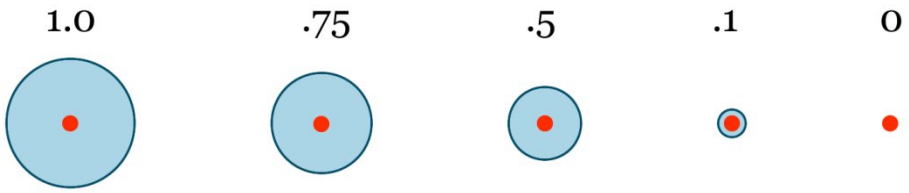


Figure 3.4. Cursor for above-surface interaction with radius proportionate to the height of a hand above the surface: from 1 (maximum height of the hoverspace) to 0 (touching the surface)

surface, the height of the touch region and the size of the cursor's outer circle are equal to zero, therefore touch regions are represented as small red circles without the blue semitransparent outer part. Furthermore, the outer area of the cursor matches in diameter with the outer area of a hovering button: if a button is placed at a height of .5 of the total height of hoverspace, the width of its grey outer ring will be the same as the radius of a cursor when a hand is held at the same height (see Figure 3.3). This approach may help users learn to estimate the height of buttons (or similar widgets). This assumption was not, however, tested empirically.

3.4 Discussion

In this chapter I outlined the design constraints and considerations related to design of gestures for selection above a multitouch surface. They are grounded in the related work and are split into four main categories: non-physical interaction, the use of layers, the transition between above-surface and touch interaction, and issues of fatigue.

I then described four widgets designed specifically for above-surface interaction. The design of these widgets is based on the considerations described above. The same widgets were used in the user studies described below.

Chapter 4
System Description

While there is a multitude of systems that can be used to track a hand above a multitouch table, most of them are expensive or require custom-built components. The main motivation for building my own version of a hoverspace-enabled multitouch system was to utilize the existing hardware without any modifications or additional sensing technology.

In this chapter, I present the design of a system for above-surface hand tracking. I briefly outline the hardware used in the system and then describe the software that includes vision-based height estimation algorithms. In the process of development, I implemented two techniques to estimate the height of hands and objects above a DSI multitouch table, both based on a vision-based tracker. While I found the second approach to be simpler and more useful in my study of target selection, I include the description of the first as well, as it provides more information than just hand height estimate, and could be useful in other applications.

4.1 Hardware: Diffused Surface Illumination

The proposed method uses a standard DSI vision-based multitouch table setup, which was custom-built based on the instructions provided by Peau Productions[10] (see Figure 4.1). The setup utilizes ACRYLITE® EndLighten acrylic that is designed to accept the light through its edge and scatter it evenly throughout the surface. A piece of such acrylic of size 81 cm × 61 cm forms the surface of the table. It is edge-illuminated by 850 nm IR diodes. A rear-projection film is applied to the acrylic's surface to allow it to act as a screen for rear-mounted short-throw projector. A Unibrain Fire-I™ camera equipped with an 850 nm band-

[10] http://www.peauproductions.com/dsi.html

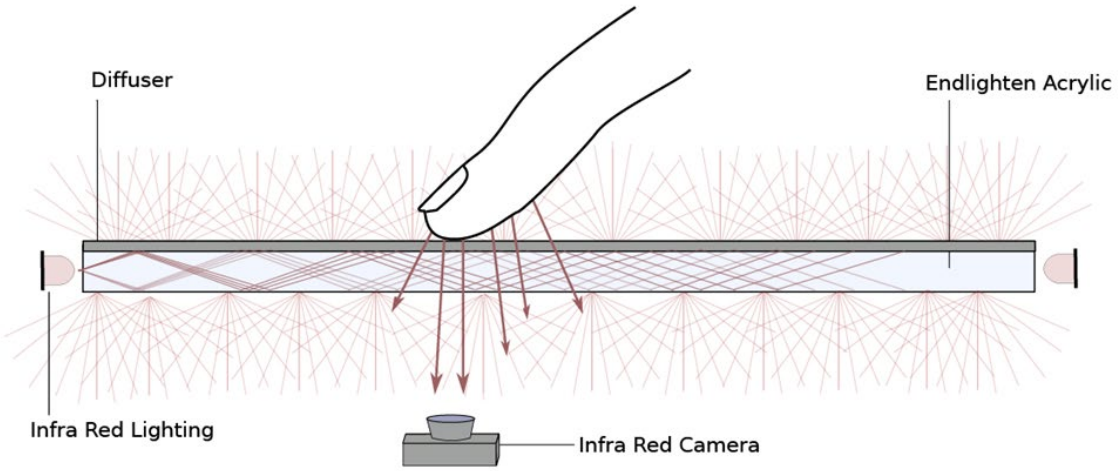


Figure 4.1. The main components of a DSI setup [11]

pass filter is mounted behind the screen. The filter reduces the contamination of the input image with light of other wavelengths and decreases the visual noise in the image. When an object, such as person's hand, approaches the table's surface, it reflects scattered IR light back through it to be detected by the camera. The amount of reflected light is inversely proportional to the distance of the object from the surface and touch regions are considerably brighter than the hovering objects, which makes them easy to recognize for the tracking software. The maximum height at which human palms can be distinguished from the noise depends on the power of IR diodes illuminating the surface, on the amount of ambient light and on the reflective properties of human skin. For my system, the maximum height is approximately 20 cm.

[11] © Seth Sandler (http://sethsandler.com/multitouch/dsi/)

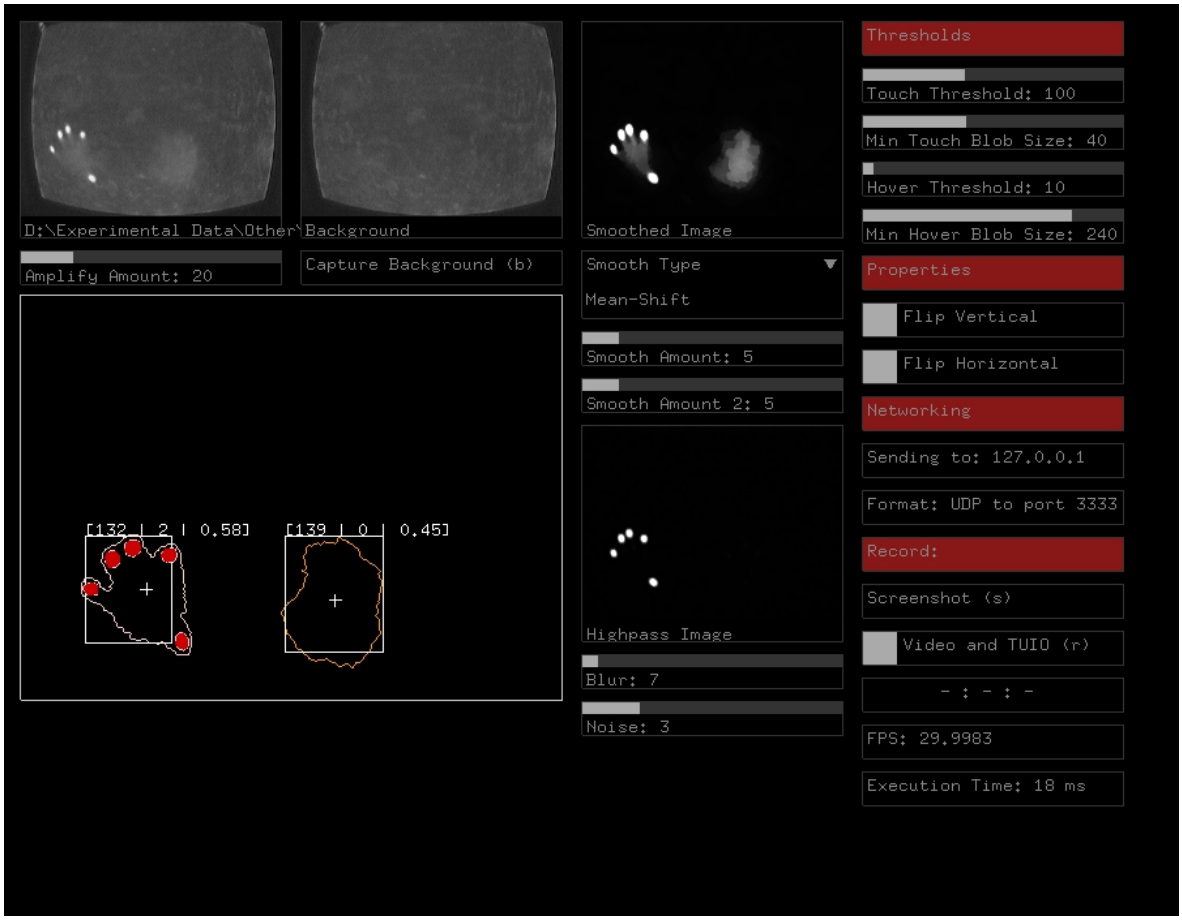


Figure 4.2. Screenshot of the tracker application

4.2 Software: Vision-based Tracker

My system is based on a common pipeline used in other vision-based multitouch trackers, for example, Community Core Vision[12] and reacTIVision[13]. I add one or more additional pipelines (in addition to the touch pipeline) to detect hands above the surface that each use a lower threshold (i.e., detects dimmer blobs) and a Mean-Shift filter [4] (to reduce

[12] http://ccv.nuigroup.com
[13] http://reactivision.sourceforge.net

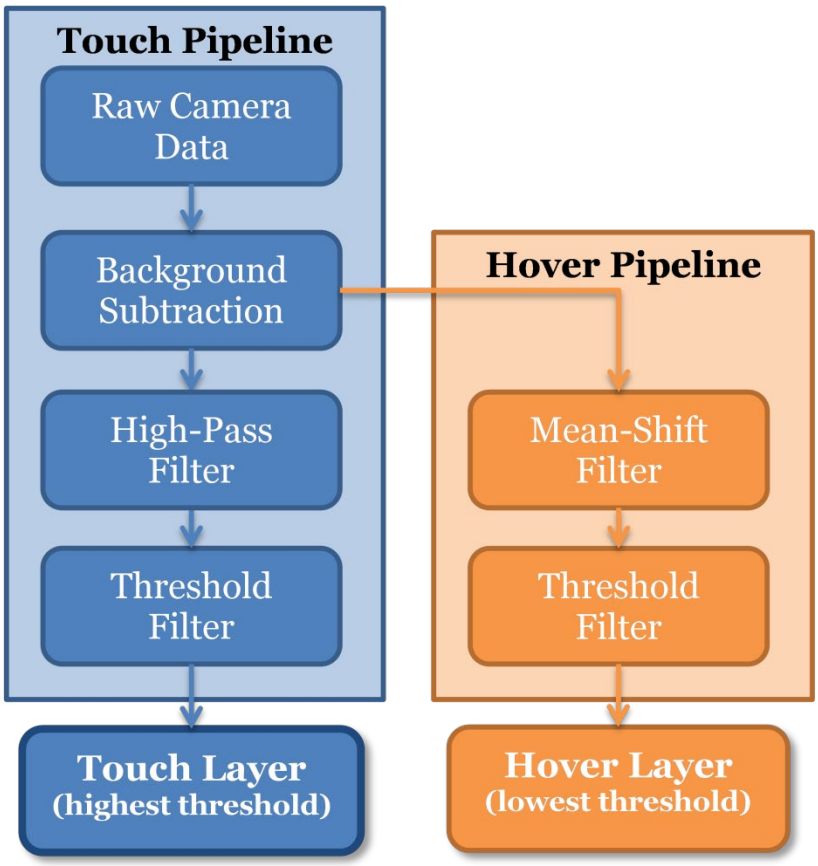


Figure 4.3. Touch and hover pipelines used in the tracker

noise). Standard blob finding and tracking algorithms are applied to the result of all pipelines and the data is sent to client applications using the TUIO [19] protocol.

The **touch pipeline** (see Figure 4.3, left) first analyzes each frame captured by the under-the-table camera, performs background subtraction and amplifies the image. Then, a high-pass filter is used to extract brighter blobs that represent touch regions; a threshold filter is then applied with a high cut-off value (as touches are typically bright).

In addition to this first pipeline, my system also provides a **second pipeline** (see Figure 4.3, right) to detect blobs above the surface. This additional pipeline is designed to detect dimmer blobs that represent hovering objects. After amplification and background

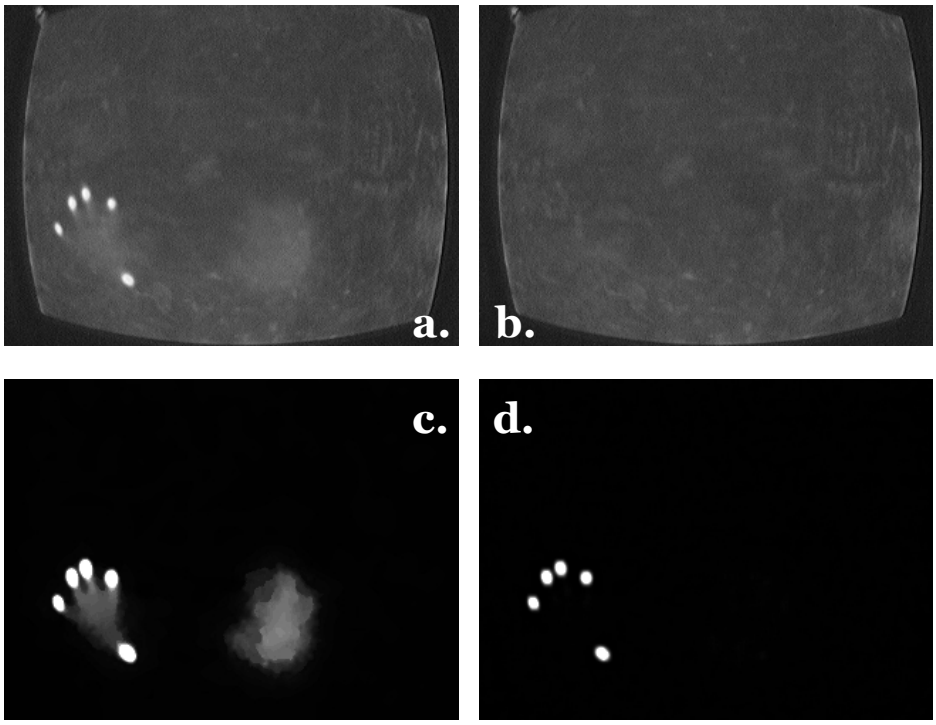


Figure 4.4. Image processing in the tracker: a. original image, b. extracted background, c. hover layer, d. touch layer

subtraction, a copy of the image is made and then this copy is smoothed using a Mean-Shift filter [4] and a threshold filter with a lower cut-off value than for touches is applied.

The choice of the Mean-Shift procedure [4] is based on its inherent ability to cluster data at the local maxima. In the scope of my application this means that after applying this filter to the blurry noisy input image the bright blobs of reflected light acquire sharp outlines and are not merged together, like in cases when a Gaussian smoothing is used. Figure 4.4 shows a single frame processed by this pipeline.

A blob finding algorithm is applied to the result of both pipelines. Each blob is tracked across multiple frames (using a simple nearest neighbour algorithm) and its position, speed, and acceleration are calculated. The tracker then sends this data to multitouch client applications using the TUIO [19] protocol. The ability to switch between 2D and 3D profiles

defined in this protocol allows the application to be compatible with existing client applications.

My tracker requires that a camera calibration procedure is performed when it is installed on a new device. During this procedure an array of points is projected onto the screen and when they are touched, the system records the position of the touch regions as perceived by the camera. As a result, this calibration allows my software to translate camera-based coordinates into screen space coordinates and detect touches correctly.

4.2.1 Hover Height Estimation

Currently, there are many methods that can be used to estimate the height of a palm above a surface: stereo cameras [20,35], depth cameras (like Kinect) [3,33], multiple layers of lasers [28], infrared emitters above a table [5] and so on. What distinguishes my tracker from other methods is the ability to estimate the height of a hovering hand above the surface of the table, without any additional sensing technology. Some computer vision approaches, like shape-from-shading (SFS) techniques [23], may be used to compute the shape of a palm using an image from a regular camera. In the process of software development, I explored two methods of palm height estimation based on the more general SFS problem: the slices method and the centre-weighted average method.

4.2.1.1 Slices Method

My original idea was to use nine additional hover pipelines instead of just one, gradually decreasing in cut-off value, and each representing a different slice of height above the table (the algorithm is adapted from [24]). Each of these slices was processed by a blob-finding algorithm. Since the brightness of the reflected light is inversely proportional to the distance of the object from the surface, the dimmer blobs were intended to represent parts of the

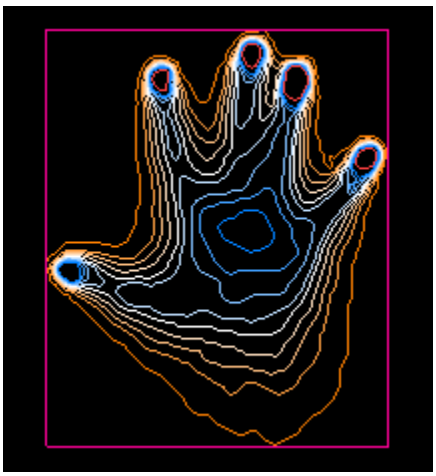


Figure 4.5. Visualization of the height data produces by the Slices method: orange outlines represent the dimmest parts of the image, blue outlines represent brighter parts and red outlines represent touch regions

hand that are further away from the surface (see Figure 4.5). It should be noted, however, that the amount of the reflected light is not only affected by the distance, but also by the intrinsic reflectivity of the object. For example, fingers reflect less light than the middle of a palm; therefore a palm cupped towards the screen may appear brighter than the tips of the fingers, even though they are closer to the screen.

Thus, instead of estimating the height of each slice, I aggregated the information from all slices to get an estimate of the height of an entire hand. Specifically, the higher the hand is above the table, the fewer slices in which it will appear. The software then calculated the height of a hand above the surface based on the number of slices that were visible in the image of the hand.

A potential weakness of this method is the low resolution and high computational demand of the blob-finding algorithm run on each slice of the image. My multitouch setup allowed me to detect hands approximately up to 20 cm above the surface; given 10 layers of the “slices” method, each layer was approximately 2 cm thick. Such low resolution is not

sufficient to detect finer hand motions and to accurately estimate the speed and acceleration of hovering objects.

Despite its potential drawbacks, this method presents interesting opportunities for gesture recognition, because it produces an approximate 3D surface of a hovering hand. This shape might be used to recognize a richer set of interactions and gestures than the second method is able to recognize.

4.2.1.2 Centre-weighted Average Method

Another method that produced better results and required much less processing power is a "centre-weighted average" approach. This method uses only one hover pipeline (thus reducing the processing power requirements, as compared to the previous method). It computes an average value of the 16×16 pixel square located in the geometric centre of each blob detected in the hover layer. The resulting value is then mapped to the height of a hovering hand using a formula derived from the linear regression described below. I experimented with the centre-weighted approach using 6×6, 10×10 and 16×16 pixel squares; however the 16×16 square produced the best fit. Vertical tracking permitted by this method is smoother and more precise compared to the "slices" method described above. Higher vertical resolution (in the range of 0-255 instead of 0-9 of the previous method) means that the system is better at detecting finer hand motions and calculating vertical speed and acceleration of hovering objects.

A potential weakness of this method is the choice of which pixels are used in it. Perhaps computing an average value of all pixels in the blob would produce a better estimate of the height. However, that approach would be much more computationally intensive than my current method. In my current approach I simply select a rectangular region of interest

(ROI) in the centre of the blob and calculate its average value; this can be computed extremely fast in OpenCV framework[14], which is used for image processing in my tracker.

4.2.1.3 Shape from Shading Problem

The task of approximating a distance from an object based on its image can be considered a special case of a more general problem called "Shape from Shading" (SFS) [23]. SFS problem is to compute the shape of a surface using a single greyscale image as an input. My proposed height estimation methods are based on the more general SFS problem and therefore inherit its difficulties and limitations, such as concave/convex surface ambiguity and the fact that the source image is heavily influenced by the properties of the surface (such as reflectivity) and lighting (direction and magnitude). To address these issues I have chosen to solve the problem empirically, not analytically.

I have recorded 5-second videos of 16 participants holding their dominant hand at 10 different heights above the surface (from 2 cm to 20 cm with a step of 2 cm). To assist the participants with the task of holding their hand steady, a ruler was placed on the surface of the table with a sliding clip to indicate the height. The participants were asked to place their palm parallel to the surface right under the clip and the height of the clip was adjusted in such a way that the bottom of the palm was within required distance from the surface. It should be noted, however, that even with this set up, participants of the study were unable to hold their palms perfectly still and some measurement error should be expected. I think that this error in negligible, though, because jitter of human palm is expected during the normal use of the system. The recorded videos were analyzed using the slices method and centre-weighted average method (using squares of 3 sizes: 6×6, 10×10 and 16×16 pixels) and the resulting values were recorded for each frame. Videos of one of the participants were

[14] http://opencv.willowgarage.com/

removed from the analysis due to an error during the experiment, which possibly resulted in data being mislabelled. At 30 fps each video has approximately 150 frames, which in total gave the following: 15 participants × 10 heights × ~150 frames = 25426 data points. Scatter plots of the relationships between the measured values and the actual height were plotted (see Figure 4.6); an exponential function was then fitted to these plots (using Matlab's Curve Fitting Toolbox™[15]) and the goodness of fit values (R^2) of the resulting fits were then used to compare the accuracy of height estimation methods. The centre-weighted average method with 16×16 pixel square produced the best fit of $R^2 > 96\%$ and the equation of the fit was used in the system to estimate the height of a hand based on the computed value:

$$f(x) = 5.971^{-0.07258x} + 20.26^{-0.006696x}$$

To evaluate the predictive power of the fit, I have analyzed the errors of prediction. I calculated an error of prediction for each data point using the fit function above ($Error(x) = |f(x) - actualHeight_x|$). The mean value of error for the whole set of data points is .92 cm ($SD = .66$, $max = 2.86$). To get better insight about the performance of the height estimation function, I have analyzed it in segments: at each of the measured heights (see Figure 4.7).

Overall, my height estimation function seems to produce fairly good result with average prediction errors of 1-1.5 cm. This precision is comparable to the human ability to hold a hand steadily in the air and it should be sufficient for above-surface interaction.

In the future, better evaluation and comparison of the height estimation methods is required, since I used the training dataset for evaluation. More height data should be collected and compared to the predictions of the fit function.

[15] http://www.mathworks.com/products/curvefitting/

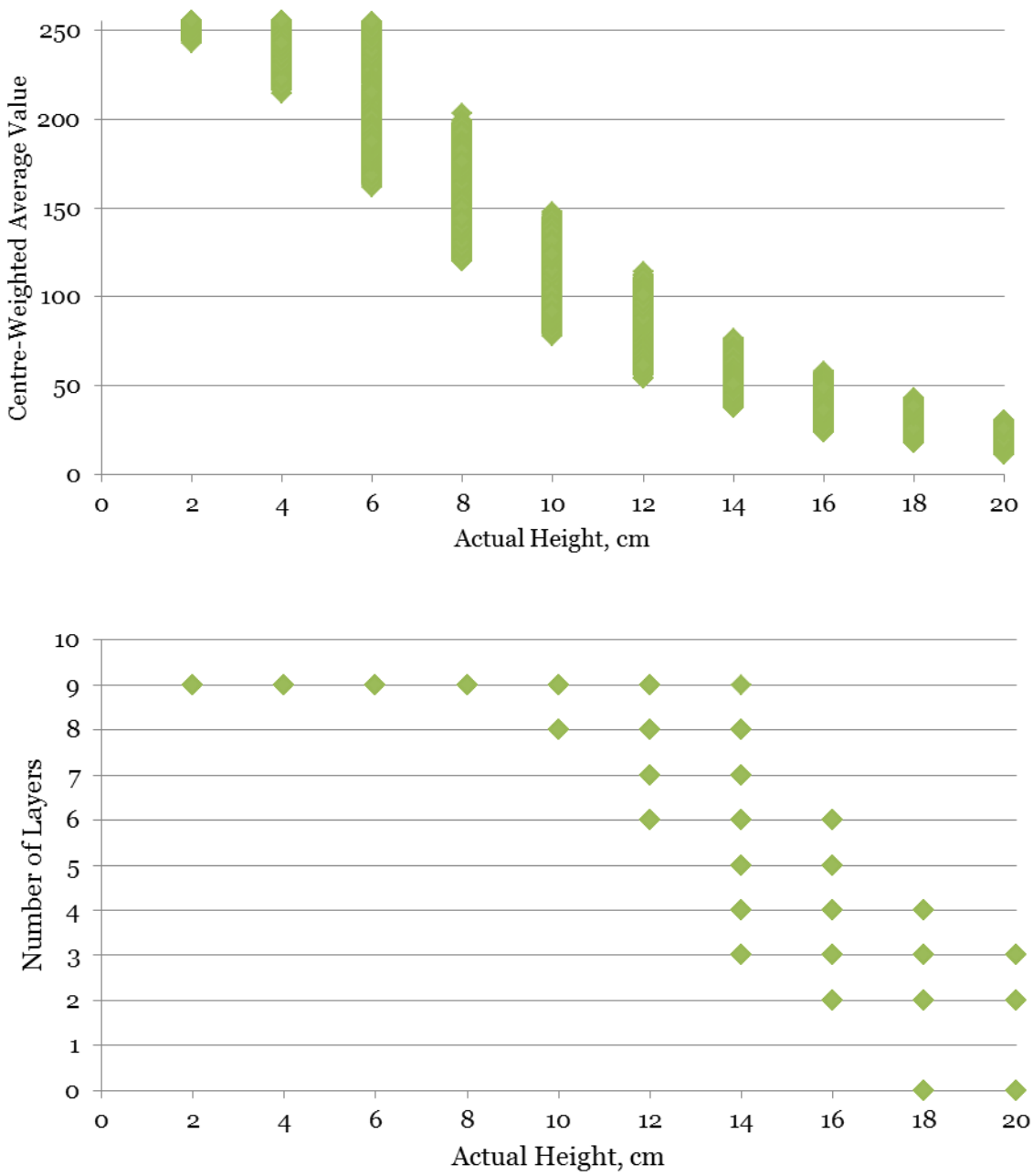


Figure 4.6. Scatter plots of relationships between the measured values and the actual height of a hand above the surface

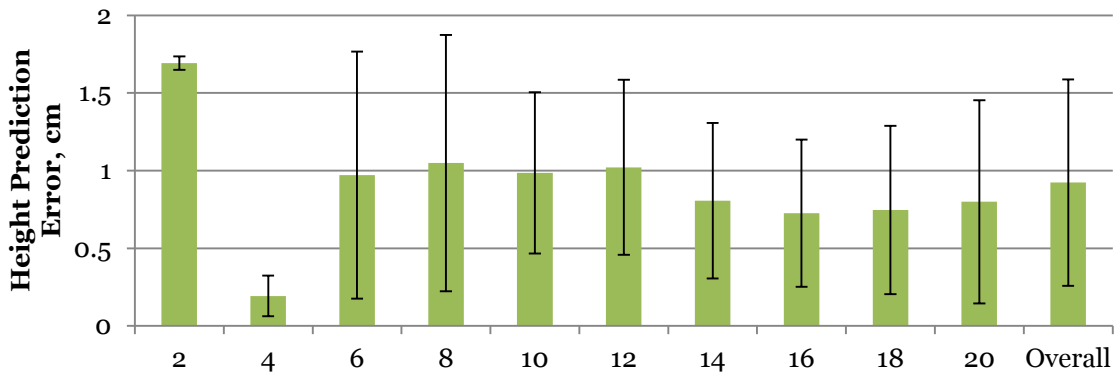


Figure 4.7. Mean error of prediction at each height (from 2 to 20 cm) and overall (error bars represent standard deviation)

4.3 Discussion

In this chapter I presented the system that was developed to enable researchers to study the area of above-surface item selection. This system cannot compare in precision to a high-end motion capture system such as Vicon, but it is cheap, simple, easy to set up and use. My system does not require any modifications to the existing DSI table, it uses the standard TUIO protocol to communicate with multitouch clients and it is able to track human palms at a height of up to 20 cm above the surface. The following chapter describes a pair of studies that were performed using this system.

Chapter 5
Gesture Elicitation and Evaluation Studies

5.1 Experiment 1: Expected Selection Gestures

The purpose of the first study was to elicit expectations of how objects should be selected above a multitouch surface. I used the methodology of Wobbrock et al. [38] to elicit these gestures. Participants were asked to acquire targets in the space above a table, and then asked to perform the gesture that they expected would make a selection.

5.1.1 Apparatus

We used the hardware setup described in Chapter 4 and the centre-weighted average method (sub-section 4.2.1.2) to detect the height of a person's hand above the table.

5.1.2 Participants

Sixteen paid participants (6 female) took part in the study. All were right-handed and the average age was 24.8 years ($SD = 5.08$). Since one of the objectives of the study was to minimize the effect that concern for reliable recognition has on the design of gestures (as might happen when designers create both the system and the gestures for it), I recruited participants who had no knowledge of the sensing technology used to track hands above the multitouch surface. Nowadays, it is nearly impossible, however, to find participants who have had no exposure to multitouch technology whatsoever. In this study 13 participants had previously used smartphones equipped with a multitouch screen, and 7 had previously used tablet PCs and public multitouch devices such as bank machines or airport check-in kiosks. None reported having any in-depth understanding of the sensing technology, however.

5.1.3 Design

I used a within-participants factorial design with the following three factors:

- Widget (bar, circle, button, menu)
- Number of hands (one, two)
- Anchoring (screen, cursor)

5.1.3.1 Task

Participants were shown one of the four widgets (described in section 3.2) and asked to interact by moving their hand in the 3D space above the multitouch surface. When the system detected a hand hovering above the surface, a cursor was displayed. The cursor is represented as a semi-transparent circle with its centre directly below the geometric centre of a hovering palm and its size proportionately related to the height of the palm above the surface. Participants were then asked to move this cursor to the target (which varied by widget) and then demonstrate what gesture they would use to select that target. The software did not attempt to recognize or act on the gestures performed by the participants; it only recorded activity above the surface. I also video recorded participants' hands using a camera positioned above the screen.

5.1.3.2 Widgets

Participants were asked to perform gestures using four different visual widgets. These four widgets were designed to be both abstract representations of targets (bar and circle, see sub-section 3.2.1) as well as closer approximations of widgets that could be used in an application (button and menu, see sub-section 3.2.2). Each visual widget is controlled in the same way: by moving one's hand to a predetermined target in the 3D space above the table.

5.1.3.3 Number of Hands and Anchoring

Participants were asked to demonstrate a gesture using both one hand and two hands separately. The widgets were also shown to be anchored either to the centre of the screen (i.e., remained stationary in x and y), or to the cursor (i.e., the x and y position of the widget moved with the hand).

5.1.3.4 Procedure

The order of events for each participant can be described algorithmically as follows:

1. The idea of hover space selection was introduced
2. For each widget (× 4):
    a. For each combination of hands and anchoring (× 4):
        i. Practice using the widget for as long as desired
        ii. When ready, demonstrate a gesture to select the corresponding target

The order in which the widgets appeared was counter-balanced using a random Latin square. The number of hands and target anchoring parameters were combined into a single 4-value parameter and counterbalanced using random Latin squares (one for every widget). Since the software had no means to recognize a gesture, participants were asked to indicate verbally when their gestures were complete. The experimenter then pressed a button to indicate to the software when to stop recording. With 16 participants, 4 visualizations, 1 or 2 hands and 2 anchoring methods, a total of $16 \times 4 \times 2 \times 2 = 256$ gestures were performed.

5.1.4 Gesture Classification

Once the gestures were collected I analyzed the videos recorded by the above-table camera. I manually classified the gestures performed along 3 dimensions: *palm shape*,

magnitude and *motion*. Palm shape was specific to one hand; therefore the shape of the second palm was also analyzed in two-handed gestures. Examples of the *palm shape* category are: open palm, closed fist, and closed fist with an extended index finger. The gesture *magnitude* describes how much of the palm was involved in the gesture; it ranged from full palm gestures to single finger gestures. The gesture *motion* category describes the path that the hand or a finger followed during the gesture. Depending on the magnitude of the gesture, the motion can describe the path of the full palm or a single finger.

Similar to the findings reported by Wobbrock et al. [38], I noticed that participants did not attach significance to which fingers were used in a gesture and how many fingers were involved. Some performed gestures using their index finger interchangeably with their middle finger or thumb.

Gesture	Description	Freq. 1 hand	Freq. 2 hands
Off-hand tap	Tapping the screen with a single or several fingers of the off hand	N/A	37.5%
Grab	Grabbing or pinching gesture with one or both hands	35.2%	23.4%
Push with a finger	Downwards motion of a single or several fingers	26.6%	10.2%
Snapping/Clapping	Clapping hands together or snapping fingers (sound-based interaction)	9.4%	7.0%
Spread/Expand	Both hands moving horizontally from the target to the edges of the surface	N/A	6.3%
Push	Downwards motion of a full hand, by bending wrist, elbow or shoulder joint. A version of the gesture was performed by bending all fingers downwards.	9.4%	3.9%
Tap	Tapping the screen with a single or several fingers	7.8%	0.0%
Dwell	A hand is held steady in the same place for a set period of time	6.3%	0.0%
Shake hand	A hand is held in the same place and shaken	3.9%	2.3%
Swipe	Horizontal motion above the surface	0.8%	6.3%
Other	Other gestures, such as rotating a palm or bumping palms together	0.8%	3.1%

Table 5.1. The frequencies of gestures demonstrated in the first experiment

5.1.4.1 Agreement Scores

I have grouped gestures with similar *palm shape, magnitude,* and *motion* into 11 groups (see Table 5.1). Group size was then used to compute an agreement score A that reflects, in a single number, the degree of consensus among participants (this process was adopted from [38]). The formula to calculate the agreement score is:

$$A = \sum_{P_i \subseteq P} \left(\frac{|P_i|}{|P|} \right)^2$$

Where P is the set of all proposed gestures for a certain condition (defined by widget, number of hands and anchoring) and P_i is a set of identical gestures from P.

5.1.5 Results

The average agreement score for each condition was .23 (SD = .008). This small variability indicates that the study factors had very little effect on agreement (see Figure 5.1). The overall agreement for one-handed and two-handed gestures was .22. In contrast, the agreement scores of most gestures selected for the user-defined set in the results of Wobbrock et al.'s study [38], were between .30 and 1.0 for a single hand and between .30 and .60 for both hands. My results indicate that there is no clear agreement between participants about how selection should be performed above a surface.

While the agreement scores were low, there were several gestures that were performed more often than others. With one hand, the *grab* gesture was performed 45 times (35.2%) and *push with a finger* was performed 34 times (26.6%). With two hands, *off-hand tap* (37.5%) and *grab* (23.4%) were performed more often than the others. Overall participants preferred one-handed gestures, as indicated either verbally or in the post-study survey. This finding agrees with the results of Wobbrock et al's gesture-elicitation study [38].

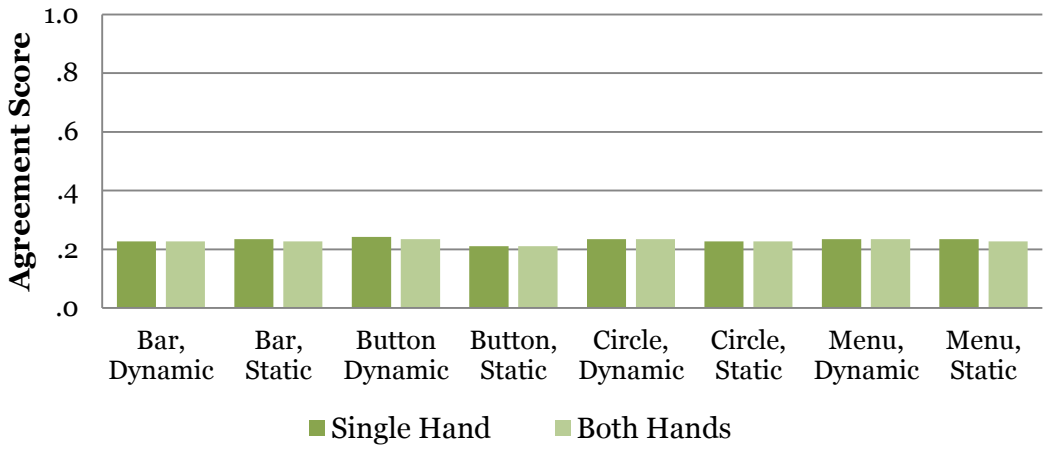


Figure 5.1. Agreement for each condition

I noticed that most participants (12 out of 16, or 75%) consistently used the same one-handed gesture and the same two-handed gesture for each experimental condition. Based on this observation, and in order to reduce the complexity of the factorial design, I opted to use only one visualisation in the second experiment, instead of testing all four.

5.2 Experiment 2: Gesture Performance

From the first experiment, I was able to identify *off-hand tap, grab* and *push with a finger* as possible candidates for a selection technique for targets above a multi-touch surface. The second experiment was designed to measure the performance of these candidate methods of above-surface selection, as well as some techniques I suspected might be effective. I was interested in comparing three properties of each gesture: how fast it can be performed, how accurately it can be performed, and how difficult it would be for the computer to recognize and disambiguate the gesture. To do so, I designed an experiment where the participant had to first *acquire* a target in hover space (above the screen), and then *perform* one of the gestures to select the target (like a mouse click). Unfortunately, the

system I used was not accurate enough to recognize some of the gestures due to the low resolution and ambiguities inherent to shape-from-shading techniques. Too many factors can change the way a palm looks in greyscale apart from its height; for example the reflectivity of the skin on the top and bottom of the human palm is different, so the system would not be able to differentiate between a hand held palm-down higher above the table and a hand held palm-up closer to the surface. The *grab* gesture appears to the software as indistinguishable from lifting one's palm higher above the surface. *Push with a finger* could not be recognized because the change in the image of the hand was too small to be differentiated from noise. For the same reason I was unable to include such designer-defined gestures as *SideTrigger* [2] or *ThumbTrigger* [29]. However, I hope that these limitations will be addressed in future work using this research as a foundation.

As a result of these limitations, I decided to focus my second study on evaluating the performance of gestures that were practical to implement with my minimal hardware. I chose *push* (as it is a close approximation to *push with a finger*) and selected the most common two-handed gesture suggested by the participants: *off-hand tap*. In contrast to user-defined gestures, *dwell* and *droptap* (move hand down rapidly and tap the screen) were also included in the experiment as those I expected to perform well based on my design experience (i.e., designer-defined), even though they were infrequently chosen by participants in the first study. Given the success of the Kinect sensor and its use of the dwell gesture for in-air selection, it is reasonable to expect that people familiar with that system may transfer their experience to item selection in a space above a horizontal surface.

5.2.1 Apparatus

We used the hardware setup described in Chapter 4 and the centre-weighted average method (sub-section 4.2.1.2) to detect the height of a person's hand above the table.

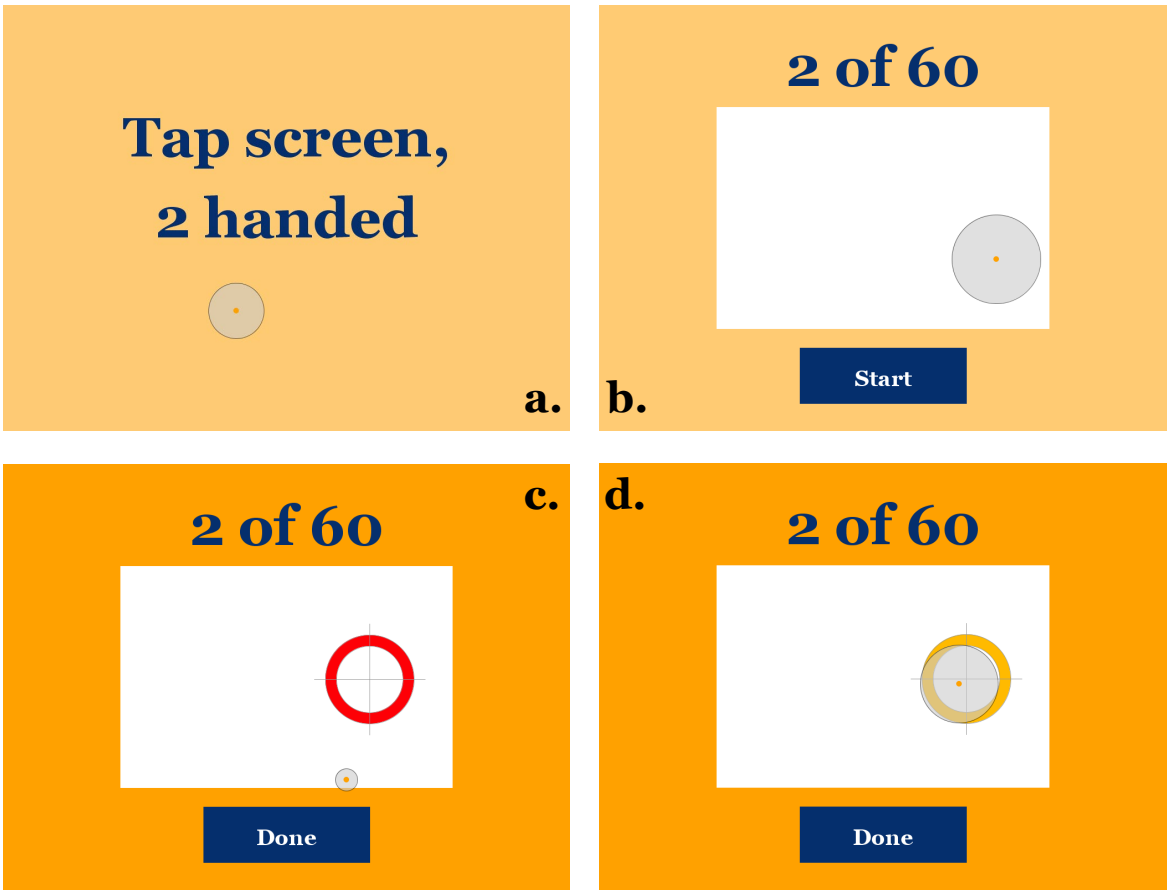


Figure 5.2. Screenshots of the experimental application: a. splash screen indicating the study condition; b. "parking area" is displayed when condition starts; c. a target is displayed after the Start button is pressed; d. the target changes colour when acquired

5.2.2 Participants

Sixteen paid participants (5 female) took part in the study. A single participant was left-handed and the average age was 24.1 years ($SD = 3.17$). Some of the participants took part in the first study as well. All participants were students of a local university and most of them majored in computer science or engineering.

5.2.3 Design

I used a within-participants factorial design with the following two factors:

- Gesture (dwell × 3, push, droptap, off-hand tap)
- Location (dominant, middle, non-dominant)

5.2.3.1 Task

To begin each trial, the participant was asked to touch a specific "parking area" of the screen labelled "Start" with their finger (see Figure 5.2 b). When the participant touched this parking area, a target was displayed (see Figure 5.2 c). The target is identical to the circle widget used in the first study, except for an adjustment in color scheme (see Figure 5.3). The participant was asked to acquire this target, using the same cursor as in the first experiment, by moving their hand so that the centre of their palm was directly above the centre of the ring (i.e., at the crosshair), and the height of their hand made the cursor radius match the target radius. Since the target was virtually located above the surface, the participants had to not only match its x, y position on the surface, but also its z position, or height above the surface. Once acquired, the participant then performed one of six selection gestures: *short dwell*, *medium dwell*, *long dwell*, *push*, *droptap*, or *off-hand tap*.

The three dwell gestures required participants to hold their hand above the target for 500ms, 1000ms, and 2000ms, respectively. The push gesture required participants to move their hand rapidly in the downward direction, and was detected when the speed of the hand was above ~10 m/s. To perform the droptap gesture, a participant had to move their hand down rapidly (with the same speed as the push gesture) and then touch the screen. The last z position of the cursor prior to the start of this rapid motion was saved, and when a touch event was detected, it was used to determine if the target was activated successfully. Off-

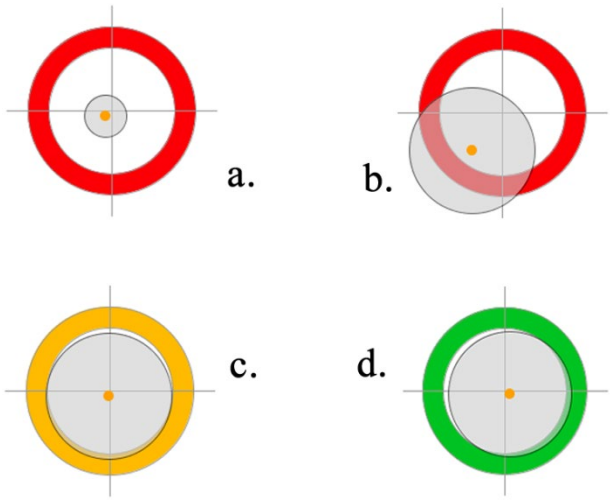


Figure 5.3. The targets (circular rings) and cursor used in experiment 2. Targets first appears red (a. & b.), turns yellow when acquired (c.) and green upon selection (d.). The radius of the cursor is determined by the height of the participant's hand above the table.

hand tap gestures were completed when the participant touched the table anywhere with a finger on the non-dominant hand.

Targets appeared in one of the three x, y locations: on the dominant side (right for right-handed participants, and left for left-handed), in the middle, or on the non-dominant side of the screen. The height and distance of the target from the "parking area" was kept constant, so that participants' hands had to be at 14 cm above the table and 25.4 cm from the start position along the table's surface. The participant was asked to acquire the target and perform a gesture as quickly as possible.

To indicate the state of the target, the following color scheme was used: initially the target was red, when the centre of the cursor was within 1cm of the crosshair, the target changed to yellow, and when a gesture was completed, the target changed to green (see Figure 5.3). While I did not explicitly screen participants for colour-blindness, none reported being unable to distinguish between the colours used in the study. For both the *push* and *droptap* gestures, movement beyond the target boundaries was required to perform the

gesture, and so once the target was first acquired (yellow), the target would not return to its non-acquired state (red).

5.2.3.2 Procedure

The order of events for each participant can be described algorithmically as follows:

1. The idea of hover space selection was introduced
2. For each gesture (× 6):
    a. The experimenter demonstrated the gesture
    b. The participant performed a practice trial
    c. For each trial (3 locations × 3 repetitions = 9), each participant was asked to:
        i. Move their hand to the “parking area”
        ii. Acquire and selected the target

As the dwell gestures all required only one explanation, the three dwell gestures were presented together (one after the other) in random order. The order of dwell, push, droptap, and off-hand tap was then counterbalanced using a random Latin square. The order of the 3 locations was randomized.

The application recorded the path of the hand and the timing of events, as well as target loss and successful/failed gestures. The trial was considered successful when a gesture was recognized while the target was acquired. If a gesture was performed outside of the target or a gesture was never performed, the trial was marked as failed.

After each block of 9 trials, a participant answered three 7-point Likert scale questions about the ease of navigation to the target, performing the gesture and the overall experience. With 16 participants, 6 gestures and 9 repetitions, a total of $16 \times 6 \times 9 = 864$ gestures were performed.

5.2.4 Results

Due to the shape and ergonomics of the experimental multitouch table, most participants had difficulty acquiring targets on the side of the table opposite their dominant hand. The table was shaped as a coffee table and its small height (51 cm) forced participants to bend over the table or kneel next to it; which meant that to reach the left side of the screen, right-handed people had to rotate their torso and/or shoulders, making the target acquisition awkward and uncomfortable (and vice versa for the left-handed participant). I performed a one-way analysis of variance (ANOVA) on the location factor, which showed a significant effect ($F(2,30) = 32.19, p < .001$). Post-hoc pairwise comparison of the location factor showed a significant difference between the non-dominant location and both other locations ($p < .001$), while the middle and dominant locations were not significantly different ($p = .113$). I thus removed data from the non-dominant level of the location factor from the remainder of the analysis. Therefore, a total of $16 \times 6 \times 6 = 576$ gestures were considered in the analysis. I analyzed three main dependent measures: gesture speed, gesture accuracy, and participant preference.

5.2.4.1 Gesture Speed

The time to perform a gesture can be broken down into three parts: acquisition time, jitter time, and selection time.

Acquisition time was measured as the time it takes to move a hand from the starting area to the target. Using a 6 (gesture) × 2 (location) repeated-measures ANOVA, I found a significant main effect of gesture on acquisition time ($F(5,75) = 5.528, p < .001$). In particular, the *short dwell* gesture had significantly smaller acquisition times than all other gestures (*medium dwell*: $p < .01$, *long dwell:* $p = .024$, *push:* $p < 0.001$, *droptap:* $p < 0.01$; *off-hand tap:* $p < 0.001$). Acquisition in the medium dwell was also significantly faster than

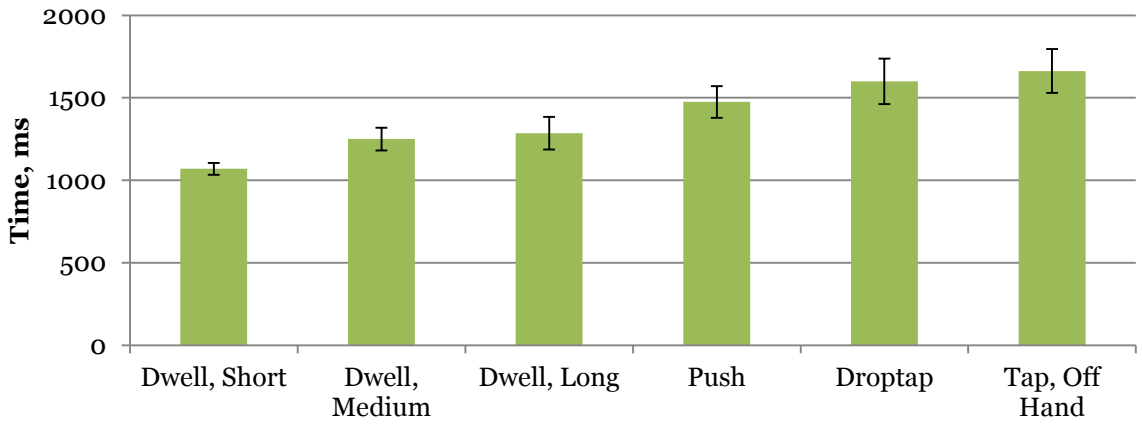


Figure 5.4. Acquisition times for each gesture

push ($p = .035$), *droptap* ($p = .028$), and *off-hand tap* ($p < .01$). The *long dwell* was also significantly faster than *off-hand tap* ($p = .039$). Acquisition times for *push*, *droptap,* and *off-hand tap* were not significantly different ($p > .05$). There was also no main effect of location ($F(1,15) = 2.630, p > .05$), nor interaction between gesture and location ($F(5,75) = 1.544, p > .05$).

The differences in acquisition times were surprising, and cannot be easily explained, since the acquisition task did not differ for any of the gestures; all trials required acquiring a target at the same distance from the starting location. This indicates that people adjusted their behaviour depending on the gesture they were performing. Specifically, people moved more quickly toward targets that required only a dwell. It is possible that this was due to the inaccuracy of, in particular, the short dwell for selection (described below). This inaccuracy perhaps led to a speed-accuracy trade-off. That is, participants may have noticed an inability to accurately select targets, and so increased their speed. However, it should be noted that this same trade-off did not occur for *off-hand tap*. It is also possible that the cognitive complexity of a gesture has an effect on the movement time; for example dwell is cognitively

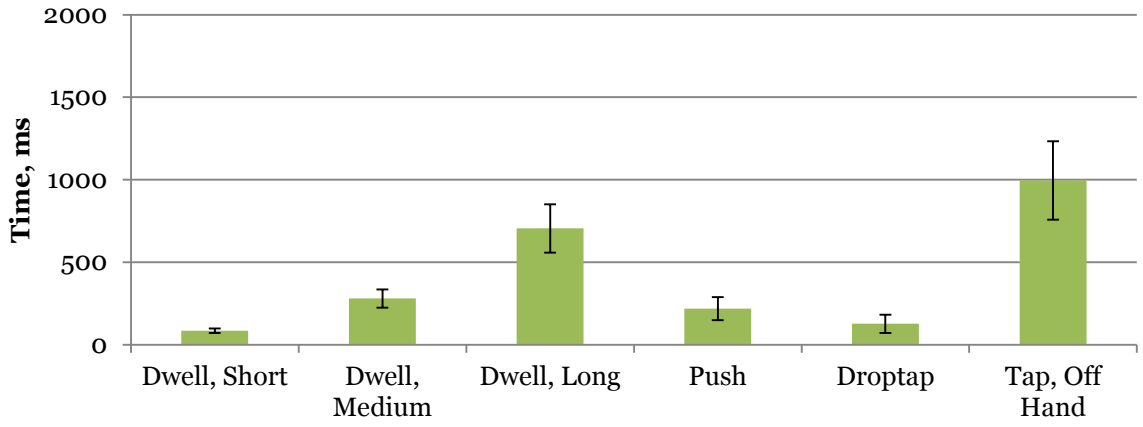


Figure 5.5. Jitter times for each gesture

simple and may result in faster motion. Further studies are required to better isolate this effect.

Jitter time was measured as the time between the initial target acquisition and the final one. In other words, jitter represents a phase when the participant lost and reacquired the target (perhaps several times) before successfully completing the gesture.

I performed a 6 (gesture) × 2 (location) repeated measures ANOVA on the jitter times. There was a significant main effect of gesture ($F(5,75) = 10.275, p < .001$). Post-hoc analysis revealed that jitter times for *long dwell* were not significantly different than *off-hand tap* ($p = .212$), and jitter times of both were significantly longer than jitter times of all other gestures ($p < .012$). *Short dwell* had significantly less jitter than both other dwells ($p < .002$), but similar to *push* and *droptap* gestures ($p > .05$). Jitter times for *medium dwell, push* and *droptap* were not significantly different. There was also no main effect of location ($F(1,15) = 0.065, p > .05$), nor interaction between gesture and location ($F(5,75) = 0.260, p > .05$).

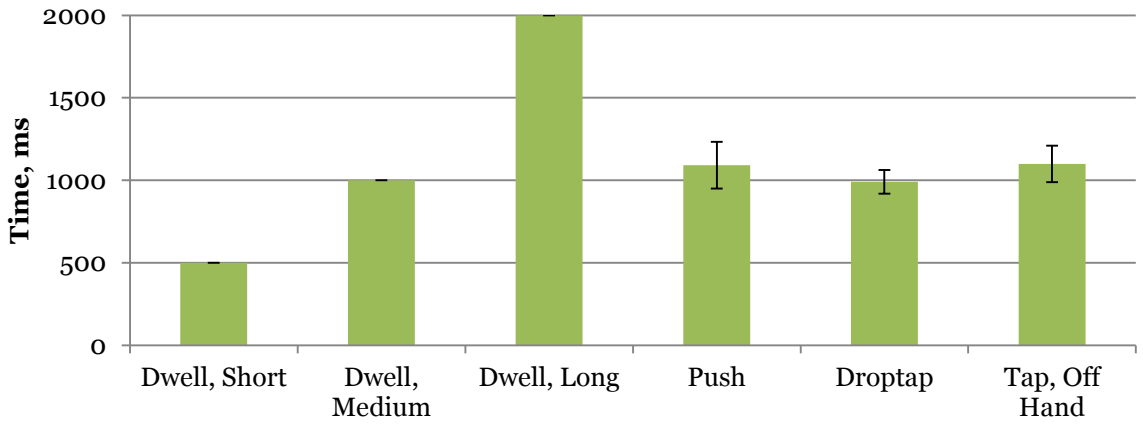


Figure 5.6. Selection times for each gesture

The increase in jitter is expected as the length of dwell increases, since it is difficult for a person to hold their hand steadily in the same place. *Off-hand tap* had more jitter than all other gestures except for *long dwell*, perhaps due to the ergonomics of the table or to the fact that participants had trouble holding their main hand steady while touching the screen with the other hand. The same effect appeared in the number of times a target was lost (see below). The other four gestures had comparable amount of jitter.

Selection time was measured as the time it takes to perform the gesture after the last acquisition (i.e., after acquisition + jitter). For the *dwell* gestures, this selection time is constant, and so was not included in the analysis. I performed a 3 (gesture) × 2 (location) repeated measures ANOVA on the remaining three gestures, but found no significant main effects or interactions (gesture: $F(2,30) = 0.372, p > .05$; location: $F(1,15) = 0.252, p > .05$; gesture × location: $F(2,30) = 0.44, p > .05$).

Overall, although I broke down the analysis by acquisition, jitter, and selection time, Figure 4.7 shows how these times would accumulate in practice. *Short dwell* was the fastest to perform, while *long dwell* was the slowest. *Push*, *droptap*, and *medium dwell* were

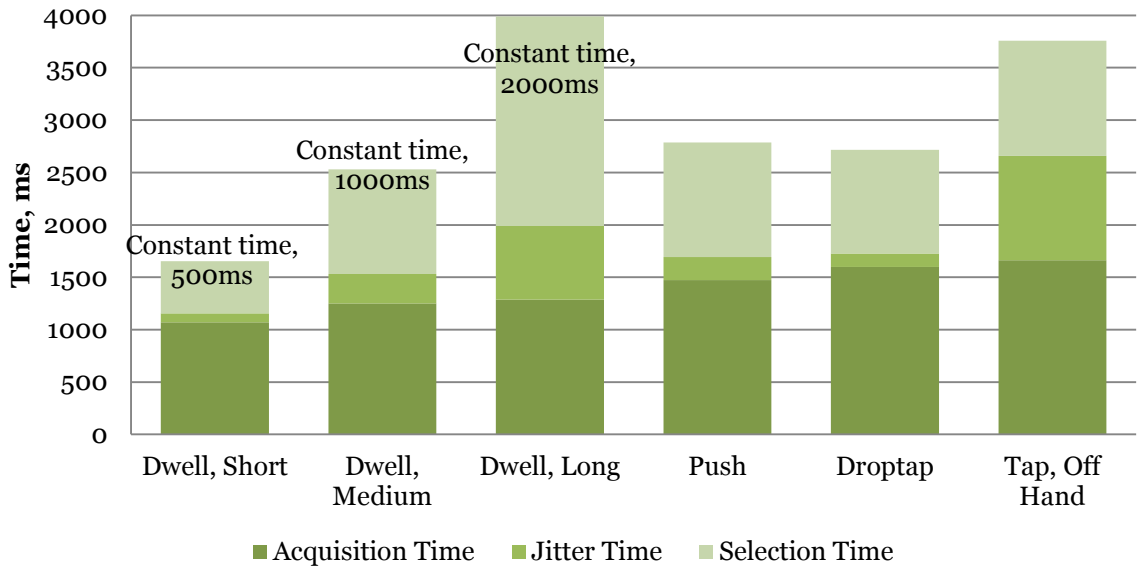


Figure 5.7. Acquisition, jitter and selection times for each gesture

comparable in speed, while *off-hand tap* performed almost as badly as *long dwell* (likely due to the high jitter times).

5.2.4.2 Gesture Accuracy

To evaluate the precision of each gesture I measured the number of times a target was lost when performing a gesture and the overall number of failed trials.

Target lost count was a measure of the difficulty in keeping one's hand on the target while selecting a gesture. This can be thought of as a count of the number of jitters per trial, rather than the time taken for jitter. The 6 (gesture) × 2 (location) ANOVA was performed on target lost count. There was a main effect of gesture ($F(5,75) = 12.947, p < 0.001$). Post-hoc analysis revealed that the target lost count of *long dwell* and *off-hand tap* gestures were significantly different from all others ($p < .034$). The target lost counts of *short* and *medium dwells* were also different from all others ($p < .030$) except for the *push* and *droptap*

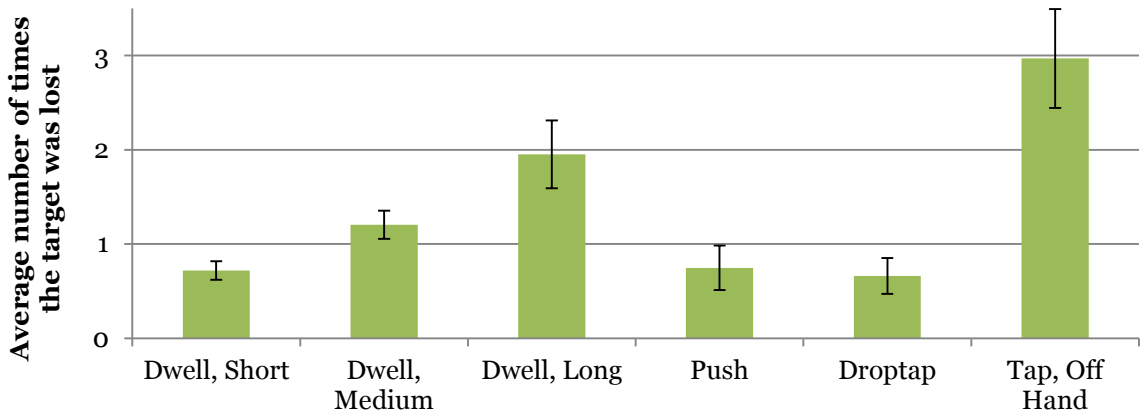


Figure 5.8. Target lost count for each gesture

gestures. There was also no main effect of location ($F(1,15) = 0.181, p > .05$), nor interaction between gesture and location ($F(5,75) = 0.368, p > .05$).

As expected, it becomes harder to stay on-target as the length of the *dwell* gesture increases. *Off-hand tap* again performed the worst for this measure. *Push* and *droptap* performed as well as *short* and *medium dwells*.

Trial failure frequency was measured as the proportion of unsuccessful trials, which were recorded if (a) the target was never acquired, (b) no selection gesture was recorded, or (c) the target was not in its acquired state when the selection gesture was performed. I performed a Cochran's Q test to analyze this binary data (each trial was either successful or not) for the gesture factor, and included each location as a repetition. I found a significant difference between the failure frequencies of the gestures ($Q_{df=5,N=96} = 50.282, p < .001$). A Post-hoc McNemar's test revealed that the *short dwell* resulted in significantly more failures than the rest of the gestures ($\chi^2_{N=96} > 6.568, p < .01$); *long dwell* resulted in fewer failures than *push* ($\chi^2_{N=96} = 8.828, p < .01$); *droptap* also had fewer unsuccessful trials than *push* ($\chi^2_{N=96} = 6.323, p = .012$).

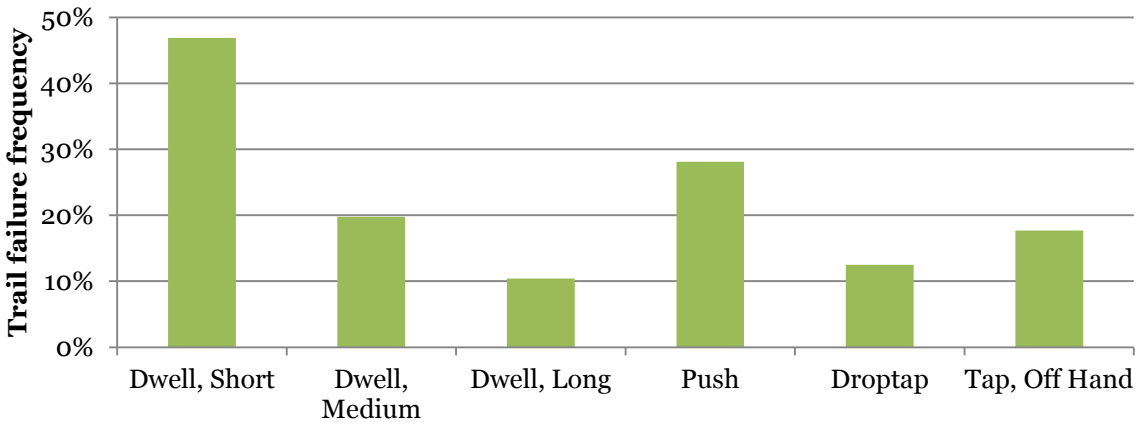


Figure 5.9. The frequency of unsuccessful trials

The number of unsuccessful trials for the *short dwell* gesture was very high ($M = .47, SD = .502$). This high error rate was not unexpected as it has been noted before and dubbed the "Midas Touch" effect [14], which negates any speed benefit noted before by resulting in many unintentional selections. Moreover, the *droptap* gesture resulted in fewer errors than the *push* gesture, and with a similar overall speed for these two gestures, this indicates that *droptap*'s overall performance was better. While the *long dwell* had similarly few errors, the added time for dwell means that *droptap* also outperforms *long dwell*.

5.2.4.3 Participant Preference

I also analyzed participants' preferences based on three 7-point Likert scale statements. The first was: "It was easy to ***move*** to the target", the second was: "It was difficult it to ***select*** the target" and the last was: "The selection technique is easy ***overall***". The words "easy" and "difficult" were used alternatingly to avoid influencing the responses; for clarity, I present the results of the responses to the scales with the word "difficult" backwards (i.e., answers 1 and 7, 2 and 6, 3 and 5 were interchanged). One of the participants did not rate one of the gestures.

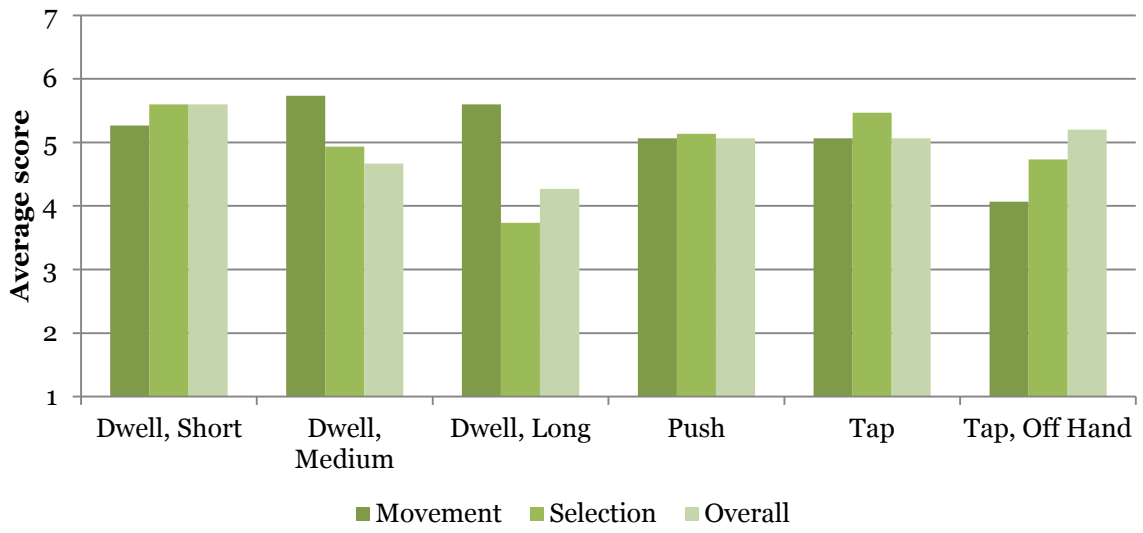


Figure 5.10. Participant preferences

A Friedman's test was performed on each of the three scales. Significant effects were found in the ***move*** and ***overall*** categories ($\chi^2_{N=15} = 17.509, p < .01$ and $\chi^2_{N=15} = 12.164, p = .033$ respectively). The ***select*** category's differences were only marginally significant ($\chi^2_{N=15} = 10.719, p = .057$). Post-hoc pairwise Wilcoxon tests were performed which revealed the following significant results:

- For ***move***, *off-hand tap* was rated significantly lower than all other gestures and *push* was rated lower than *medium dwell*.
- For ***select***, *long dwell* was rated lower than all other gestures except for the *off-hand tap*.
- For ***overall***, *short dwell* was rated higher than *medium* and *long dwells*.

The fact that most participants found moving a hand to a target in the *off-hand tap* gesture more difficult is consistent with the findings in performance. Selection using the *long dwell* gesture was rated low as expected, since two seconds is a long time to hold a hand

steady in the same place. *Short dwell* was preferred to other dwells overall, but due to the unacceptably high false activation rate I would not recommend that it be used in a real-world application. *Push* and *droptap* were rated consistently high on all 3 scales.

Unlike the participants of the first experiment, the second experiment's volunteers had to perform a gesture multiple times while worrying about the speed and precision of their gestures. Therefore their preferences may be a better indication of which gestures are more suited for an application.

5.3 Discussion

The results of this pair of studies provide some insight into the design of above-surface selection techniques. When selecting with one hand, people most frequently expect to be able to *grab* on-screen objects from a location in mid-air above that object, and with two hands expect to be able to tap with their other hand. However, this expectation was not agreed upon by all participants (35.2% and 37.5%, respectively; only between 1/3 and 2/5 of participants). Although, due to system limitations I could not easily investigate the preferred one-handed grab gesture, the investigation of a close approximation of their second choice in the *push* gesture and the *off-hand tap* two-handed gesture revealed that they underperformed when compared to the *droptap* gesture, as expected by designers. More specifically, while I found no difference in selection time between *push* or *off-hand tap* and the one-handed *droptap*, participants frequently drifted off of the target when using *off-hand tap*, and frequently missed the target with *push*.

In addition, the common *dwell* technique did not result in a suitable alternative. In particular, a trade-off between dwell time and accuracy was revealed; when the dwell time is low enough to improve speed beyond the best-performing *droptap*, the number of errors increased dramatically.

5.3.1 Design Recommendation

Based on these results, I recommend the use of a single-handed *droptap* gesture for selection of targets in hover space. However, I suggest some caution to designers in this interpretation, as my system was not capable of detecting *grab*, the most preferred selection gesture from the first experiment. Nonetheless, I note that accurate detection of a grab gesture is not simple in any of the existing hardware systems that I are aware of, whereas *droptap* can easily be detected by tracking sudden acceleration and using the existing touch capabilities of an interactive surface. I demonstrate the use of *droptap* to select colours from a colour palette in the example application (see Appendix 3 for details).

Chapter 6
Conclusions and Future Work

6.1 Summary

Above-surface interaction is an exciting new area of research, which extends gesture-based interaction into the space above the surface of a multitouch device and enriches the ways humans can interact with computers. This area presents a set of unique challenges, since the directness of manipulation inherent to on-surface interaction does not apply to the above-surface space. There are a multitude of systems that are able to track hands in mid-air and a significant amount of research has already been done to develop techniques and gestures specific for above-surface interaction. However, prior to this research, no study has given its participants the opportunity to define the ways that targets can and should be selected in this above-surface space. The research presented in this thesis was motivated by this lack of attention to user-defined target selection gestures for above-surface interaction and the belief that users should be included in the early stages of interaction design and that it is important for system designers to understand what expectations people have when they interact with these systems.

Within this thesis I presented a comprehensive overview of work in four areas of research related to the area of above-surface item selection: above-surface movement detection, interaction above a surface, the methods for studying gestures, and the area of target selection (Chapter 2). Grounded in this extensive related work, I defined a set of design constraints and considerations that were later used in the design of a set of widgets specific for above-surface interaction (Chapter 3), and then described the design of a system that allows hand tracking in the air with no additional sensors beyond a standard DSI multitouch

table (Chapter 4). Finally, I presented two studies that were conducted to first elicit gestures for above-surface selection from a group of participants and then to evaluate and compare some of these elicited gestures to several designer-defined gestures. Based on the results of these studies I was able to make recommendations to designers of above-surface interaction applications (Chapter 5).

6.2 Research Contributions

The research performed in this thesis was aimed to address the problem motivated by the lack of attention to user-defined target selection gestures for above-surface interaction. I have divided this main question into smaller and more focused research questions, which were stated in the introductory chapter of the thesis:

6.2.1 Which techniques do people expect to be able to use when selecting items above a multitouch surface?

When selecting with one hand, people most frequently expect to be able to *grab* on-screen objects from a location in mid-air above that object, and with two hands expect to be able to *tap* with their other hand. However, this expectation was not agreed upon by all participants (35.2% and 37.5%, respectively; only between 1/3 and 2/5 of participants). Most participants (12 out of 16 or 75%) consistently used the same one-handed gesture and the same two-handed gesture for each experimental condition; therefore we can assume that our widgets did not affect participants' choice of gesture.

6.2.2 Which techniques are easier to perform?

In the second study I evaluated and compared six gestures that were recognizable with my minimal hardware: short dwell, medium dwell, long dwell, push, droptap and off-hand tap.

To evaluate how fast the gesture could be performed I measured 3 parameters: acquisition time, jitter time and selection time. Based on the sum of these measures, short dwell was the fastest to perform, while long dwell was the slowest. Push, droptap, and medium dwell were comparable in speed, while off-hand tap performed almost as badly as long dwell (likely due to the high jitter times).

To evaluate the accuracy with which each gesture was performed, I measured the number of times the target was lost during the gesture and the frequency of unsuccessful trials when performing the gesture. Long dwell and off-hand tap gestures resulted in more target losses and the number of unsuccessful trials for the short dwell gesture was very high. Moreover, the droptap gesture resulted in fewer errors than the push gesture, and with a similar overall speed for these two gestures, this indicates that droptap's overall performance was better.

Participants of the study were asked to fill in a questionnaire stating their subjective opinions about how easy of difficult they found each gesture. Based on the analysis of these questionnaires I discovered that push and droptap were rated consistently high on all 3 scales.

6.2.3 Other Contributions

Other than answering the research questions above and making a set of recommendations to the system designers planning to use gestures for object selection in above-surface interaction, this thesis provides several other contributions to the field of HCI. One of these contributions is an extensive literature review of existing hardware and

software solutions for detecting hover and selecting targets in hoverspace. Another contribution is the development of a system that can detect the height of a person's hand (or hands) using low-cost hardware already available in or easy to add to many tabletop displays without requiring any additional sensing technology. And yet another contribution is a set of general widgets designed specifically for hoverspace according a set of constraints and considerations specific for above-surface interactions.

6.3 Future Work

One of the most obvious directions for future work is addressing the limitations of my hand-tracking system that made me unable to analyze some of the gestures frequently selected by the participants of the first study and the gestures proposed by other system designers. Future studies could investigate the use of high-end motion tracking systems to enable detection and recognition of the most popular one-handed gesture, grab. It should also be noted that both experiments were conducted in a controlled laboratory environment and a more realistic study (perhaps conducted in a public place and involving more participants) could be a good direction for future work. Another interesting direction for future work might investigate how objects could be used in hoverspace instead or in addition to bare hands.

Another direction for future study would be to explore the mental models that people build when manipulating objects by moving hands above the surface of a multitouch table. In my studies it was unclear how people modelled and conceptualized the idea of manipulation of objects located on the 2-dimensional screen by moving a hand in the 3-dimensional space above it. Do people imagine the object to be virtually above the screen so that it can be manipulated directly, or do they instead consider their hand to be a pointing device controlling a 2D pointer on a flat surface? Both mental models have implications with

regard to people's expectations and behaviour. It is possible that the reason why I observed such a low rate of agreement in the first study was due to the fact that people struggled with these conflicting mental models. Learning to understand and to manipulate these models could lead to better, more natural and more predictable interaction techniques.

Permissions

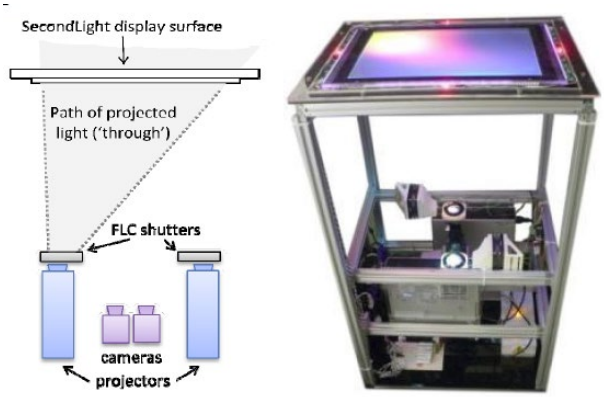
Dmitry Pyryeskin <dpyryesk@uwaterloo.ca> Tue, Sep 11, 2012 at 2:21 PM
To: shahrami@microsoft.com

Dear Shahram Izadi,

I am a Masters student at University of Waterloo, Canada, and I would like to ask your permission to put one of the figures that appear in your paper into my thesis.
The graphic I mean is Figure 10 in paper "Going beyond the display: a surface technology with an electronically switchable diffuser". Please see the graphic attached below.
I will mark the images in my thesis with a footnote: © Shahram Izadi, used with permission.

Thank you in advance,

Dmitry Pyryeskin



Shahram Izadi <shahrami@microsoft.com> Tue, Sep 11, 2012 at 3:28 PM
To: Dmitry Pyryeskin <dpyryesk@uwaterloo.ca>

Dear Dmitry,

Please feel free to include the images in your thesis. All the best,

Shahram

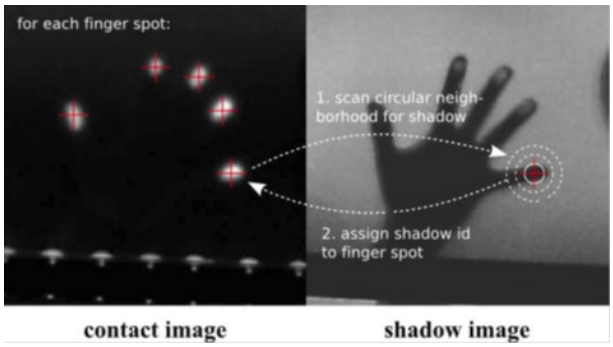
Dmitry Pyryeskin <dpyryesk@uwaterloo.ca> Tue, Sep 11, 2012 at 2:13 PM
To: echtler@in.tum.de

Dear Florian Echtler,

I am a Masters student at University of Waterloo, Canada, and I would like to ask your permission to put one of the figures that appear in your paper into my thesis.
The graphic I mean is Figure 4 in paper "Shadow tracking on multi-touch tables". Please see the graphic attached below.
I will mark the images in my thesis with a footnote: © Florian Echtler, used with permission.

Thank you in advance,

Dmitry Pyryeskin



Florian Echtler <floe@butterbrot.org> Wed, Sep 12, 2012 at 1:52 AM
To: Dmitry Pyryeskin <dpyryesk@uwaterloo.ca>

Dear Dmitry,

Please do - thanks for asking :-) If appropriate, please also cite the paper itself (the one from AVI '08, I assume).

Best regards,
Florian

Sent from my iPad

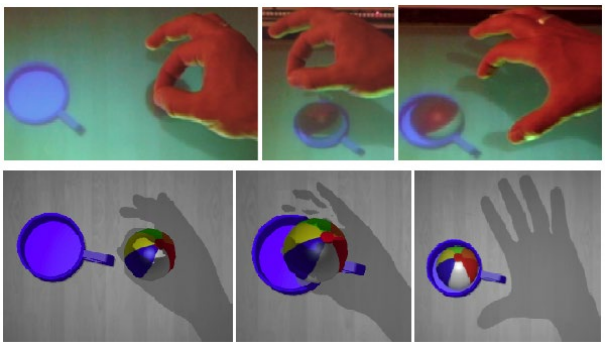
Dmitry Pyryeskin <dpyryesk@uwaterloo.ca> Sat, Sep 15, 2012 at 6:41 PM
To: otmarh@microsoft.com

Dear Otmar Hilliges,

I am a Masters student at University of Waterloo, Canada, and I would like to ask your permission to put one of the figures that appear in your paper into my thesis.
The graphic I mean is Figure 5 in paper "Interactions in the air: Adding Further Depth to Interactive Tabletops". Please see the graphic attached below.
I will mark the images in my thesis with a footnote: © Otmar Hilliges, used with permission.

Thank you in advance,

Dmitry Pyryeskin



Otmar Hilliges <otmarh@microsoft.com> Sun, Sep 16, 2012 at 8:12 AM
To: Dmitry Pyryeskin <dpyryesk@uwaterloo.ca>

Hi Dmitry,

Sure no problem at all. Just out of curiosity – what is your thesis about? And in what context will you be discussing our paper?

Thanks,

Otmar

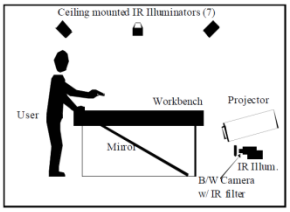
Dmitry Pyryeskin <dpyryesk@uwaterloo.ca> Tue, Sep 11, 2012 at 2:01 PM
To: leibe@umic.rwth-aachen.de

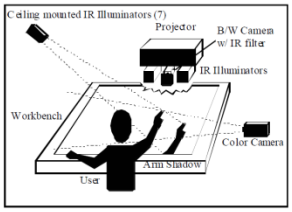
Dear Dr. Bastian Leibe,

I am a Masters student at University of Waterloo, Canada, and I would like to ask your permission to put one of the figures that appears in your paper into my thesis.
The graphic I mean is Figure 1 in paper "The Perceptive Workbench: toward spontaneous and natural interaction in semi-immersive virtual environments". Please see the graphic attached below.
I will mark the image in my thesis with a footnote: © Dr. Bastian Leibe, used with permission.

Thank you in advance,

Dmitry Pyryeskin





Bastian Leibe <leibe@umic.rwth-aachen.de> Wed, Sep 26, 2012 at 6:41 PM
To: Dmitry Pyryeskin <dpyryesk@uwaterloo.ca>

Dear Dmitry,

Sorry for my late response. Yes, no problem about that. Go ahead. Good luck for your thesis!

Best,
Bastian

Bibliography

1. Agarawala, A. and Balakrishnan, R. Keepin' it real: pushing the desktop metaphor with physics, piles and the pen. *Proc. CHI*, ACM Press (2006), 1283–1292.

2. Banerjee, A., Burstyn, J., Girouard, A., and Vertegaal, R. Pointable: an in-air pointing technique to manipulate out-of-reach targets on tabletops. *Proc. ITS*, ACM Press (2011), 11–20.

3. Benko, H. and Wilson, A.D. DepthTouch : Using Depth-Sensing Camera to Enable Freehand Interactions On and Above the Interactive Surface. In *Technical Report MSR-TR-2009-23, Microsoft Research*. 2009.

4. Comaniciu, D. and Meer, P. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24,* 5 (2002), 603–619.

5. Echtler, F., Huber, M., and Klinker, G. Shadow tracking on multi-touch tables. *Proc. AVI*, ACM Press (2008), 388–391.

6. Frisch, M., Heydekorn, J., and Dachselt, R. Investigating multi-touch and pen gestures for diagram editing on interactive surfaces. *Proc. ITS*, ACM Press (2009), 149–156.

7. Grossman, T. and Balakrishnan, R. The design and evaluation of selection techniques for 3D volumetric displays. *Proc. UIST*, ACM Press (2006), 3–12.

8. Grossman, T., Hinckley, K., Baudisch, P., Agrawala, M., and Balakrishnan, R. Hover Widgets: Using the Tracking State to Extend the Capabilities of Pen-Operated Devices. *Proc. CHI*, ACM Press (2006), 861–870.

9. Grossman, T., Wigdor, D., and Balakrishnan, R. Multi-finger gestural interaction with 3d volumetric displays. *Proc. UIST*, ACM Press (2004), 61–70.

10. Han, J.Y. Low-cost multi-touch sensing through frustrated total internal reflection. *Proc. UIST*, ACM Press (2005), 115–118.

11. Han, S. and Park, J. A study on touch & hover based interaction for zooming. *Proc. CHI*, ACM Press (2012), 2183–2188.

12. Hancock, M., Carpendale, S., and Cockburn, A. Shallow-depth 3d interaction: design and evaluation of one-, two- and three-touch techniques. *Proc. CHI*, ACM Press (2007), 1147–1156.

13. Hancock, M., ten Cate, T., and Carpendale, S. Sticky tools: full 6DOF force-based interaction for multi-touch tables. *Proc. ITS*, ACM Press (2009), 133–140.

14. Hansen, J.P., Tørning, K., Johansen, A.S., Itoh, K., and Aoki, H. Gaze typing compared with input by head and hand. *Proc. ETRA*, ACM Press (2004), 131–138.

15. Henze, N., Löcken, A., Boll, S., Hesselmann, T., and Pielot, M. Free-hand gestures for music playback. *Proc. MUM*, ACM Press (2010), 1–10.

16. Hilliges, O., Izadi, S., Wilson, A.D., Hodges, S., Garcia-Mendoza, A., and Butz, A. Interactions in the air: Adding Further Depth to Interactive Tabletops. *Proc. UIST*, ACM Press (2009), 139–148.

17. Izadi, S., Hodges, S., Taylor, S., et al. Going beyond the display: a surface technology with an electronically switchable diffuser. *Proc. UIST*, ACM Press (2008), 269–278.

18. Johnson, E.A. Touch display—a novel input/output device for computers. *Electronics Letters 1*, 8 (1965), 219–220.

19. Kaltenbrunner, M., Bovermann, T., Bencina, R., and Costanza, E. TUIO: A protocol for table-top tangible user interfaces. *Proc. ISon*, (2005).

20. Leibe, B., Starner, T., Ribarsky, W., et al. The Perceptive Workbench: toward spontaneous and natural interaction in semi-immersive virtual environments. *Proc. VR*, IEEE Comput. Soc (2000), 13–20.

21. Marquardt, N., Jota, R., Greenberg, S., and Jorge, J. The Continuous Interaction Space: Interaction Techniques Unifying Touch and Gesture On and Above a Digital Surface. *Proc. INTERACT*, Springer-Verlag (2011), 461–476.

22. Nielsen, M., Störring, M., Moeslund, T.B., and Granum, E. A procedure for developing intuitive and ergonomic gesture interfaces for HCI. *Proc. GW*, Springer (2003), 409–420.

23. Prados, E. and Faugeras, O. Shape from shading. In Springer, ed., *Handbook of Mathematical Models in Computer Vision*. 2006, 375–388.

24. Pyryeskin, D., Hancock, M., and Hoey, J. Extending interactions into hoverspace using reflected light. *Proc. ITS*, ACM Press (2011), 262–263.

25. Seay, a. F., Krum, D., Ribarsky, B., and Hodges, L. Multimodal interaction techniques for the virtual workbench. *Proc. CHI*, ACM Press (1999), 282–283.

26. Spindler, M., Martsch, M., and Dachselt, R. Going beyond the surface: studying multi-layer interaction above the tabletop. *Proc. CHI*, ACM Press (2012), 1277–1286.

27. Subramanian, S., Aliakseyeu, D., and Lucero, A. Multi-layer interaction for digital tables. *Proc. UIST*, (2006), 269.

28. Takeoka, Y., Miyaki, T., and Rekimoto, J. Z-touch: An Infrastructure for 3D gesture interaction in the proximity of tabletop surfaces. *Proc. ITS*, ACM Press (2010), 91–94.

29. Vogel, D. and Balakrishnan, R. Distant freehand pointing and clicking on very large, high resolution displays. *Proc. UIST*, ACM Press (2005), 33–42.

30. Vogel, D. and Balakrishnan, R. Occlusion-aware interfaces. *Proc. CHI*, ACM Press (2010), 263–272.

31. Wang, R.Y. and Popović, J. Real-time hand-tracking with a color glove. *Proc. SIGGRAPH*, ACM Press (2009).

32. Wheatstone, C. On Some Remarkable, and Hitherto Unobserved, Phenomena of Binocular Vision. *Philosophical Transactions of the Royal Society of London 128*, (1838), 371–394.

33. Wilson, A.D. and Benko, H. Combining Multiple Depth Cameras and Projectors for Interactions On, Above, and Between Surfaces. *Proc. UIST*, ACM Press (2010), 273–282.

34. Wilson, A.D., Izadi, S., Hilliges, O., Garcia-Mendoza, A., and Kirk, D. Bringing physics to the surface. *Proc. UIST*, ACM Press (2008), 67–76.

35. Wilson, A.D. TouchLight: An Imaging Touch Screen and Display for Gesture-Based Interaction. *Proc. ICMI*, ACM Press (2004), 69–76.

36. Wilson, A.D. Robust computer vision-based detection of pinching for one and two-handed gesture input. *Proc. UIST*, ACM Press (2006), 255–258.

37. Wilson, A.D. Using a depth camera as a touch sensor. *Proc. ITS*, ACM Press (2010), 69–72.

38. Wobbrock, J.O., Morris, M.R., and Wilson, A.D. User-defined gestures for surface computing. *Proc. CHI*, ACM Press (2009), 1083–1092.

Appendix 1
Visualizations of Hoverspace Widgets

I experimented with several ways display graphical elements (such as documents, menus and buttons) that are located at various depths on a multitouch table in an intuitive and aesthetical fashion. The main challenge for this task is the need to display 3D objects on a 2D surface. The use of orthographic projection is not feasible, since multitouch displays can be used by multiple people and the resulting image would not look natural from all angles. Instead, I decided to simulate the ways human eye use various cues to judge the distance to objects. I experimented with the concepts of simulated depth of field[16], transparency, shadows and parallax[17] to provide the visual cues about the depth of an object.

The depth of field is simulated using a simple blur filter: the farther the object is from the focus point, the blurrier it is. This visual cue of distance is based on binocular vision and it only works on short distances, up to a couple of meters, but it is sufficient for a tabletop display.

Transparency is useful to display the items that are above the currently selected item. The amount of transparency can be changed to either increase readability of the selected item or to display the occluding items more clearly.

Parallax provides great depth cue as objects move at different speeds. The effect is based on stereoscopic vision and provides much stronger cues than the depth of field. Parallax might be distracting if overused, but even a small shift of items provides sufficient amount of

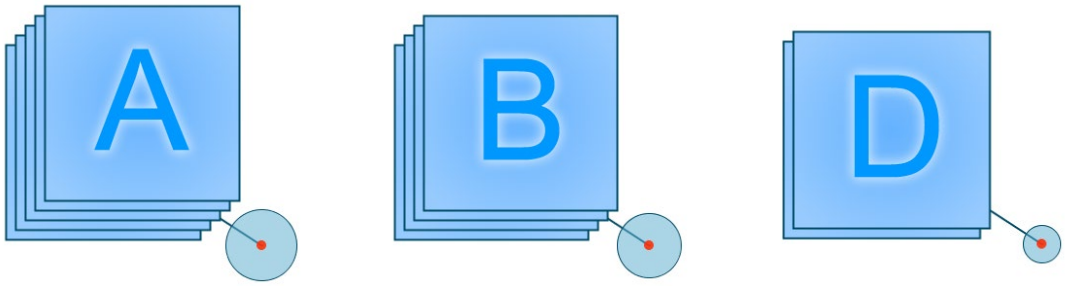
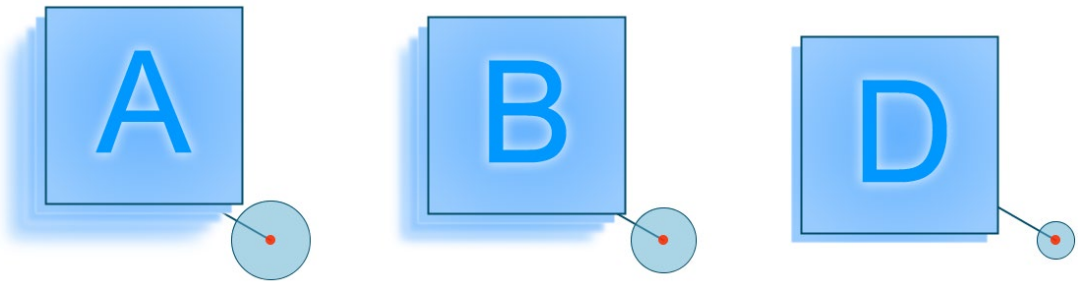
[16] http://en.wikipedia.org/wiki/Depth_of_field
[17] http://en.wikipedia.org/wiki/Parallax

depth information without taking too much screen space or attention. Combined with transparency and blur, parallax also becomes a very aesthetic effect.

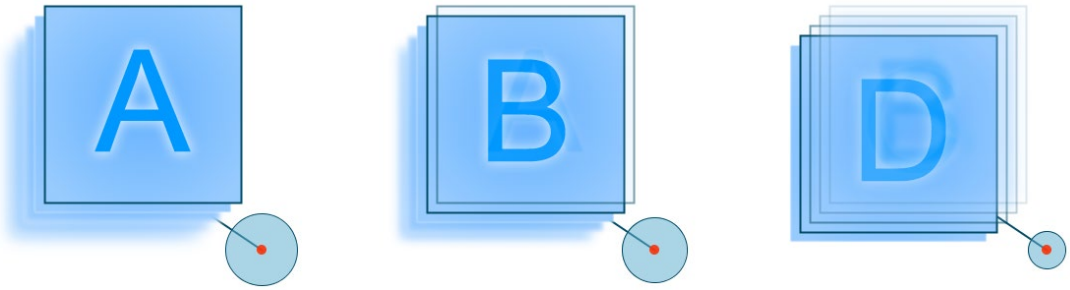
Dynamic shadow is a natural and easy-to-understand depth cue. Combined with parallax it is both effective and aesthetic.

The depth cues described above were inspired by traditional side-scrolling platformer games. These games generally use 2D graphics with effects like motion parallax, shadows and depth of field to separate background from foreground elements and create an illusion of layered background/foreground.

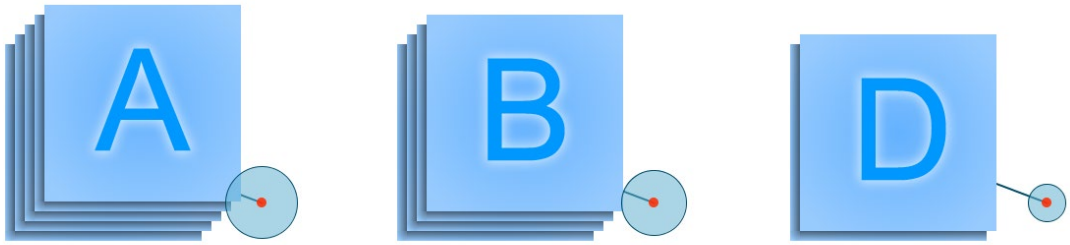
Below is the table of 9 techniques I used to simulate the depth of a stack of square objects. In my studies I used **Parallax with Blur and Transparency** method as it seemed as the most aesthetic and easy to understand.

Basic

Basic with Blur


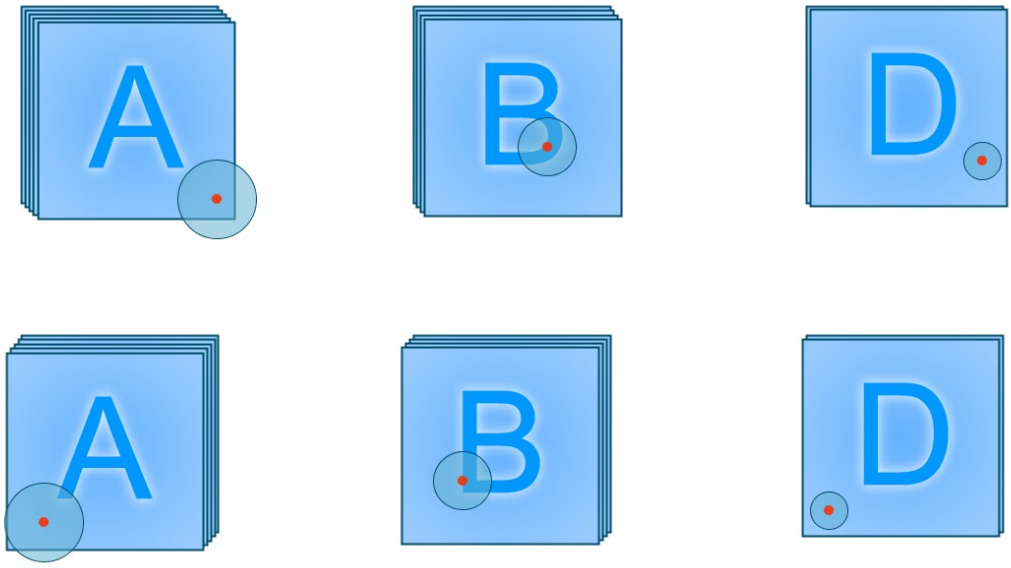
Basic with Blur and Transparency



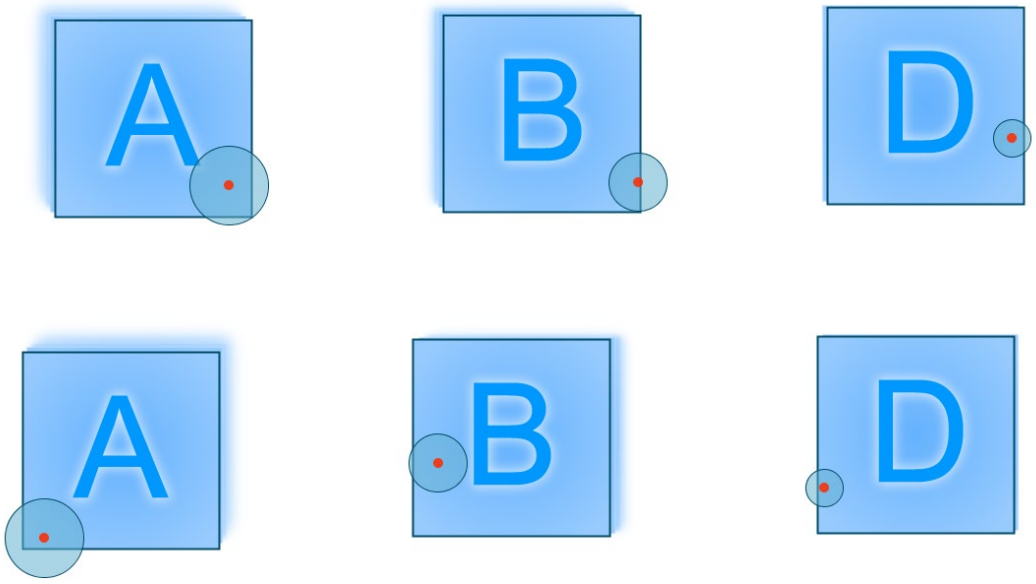
Dynamic Shadow



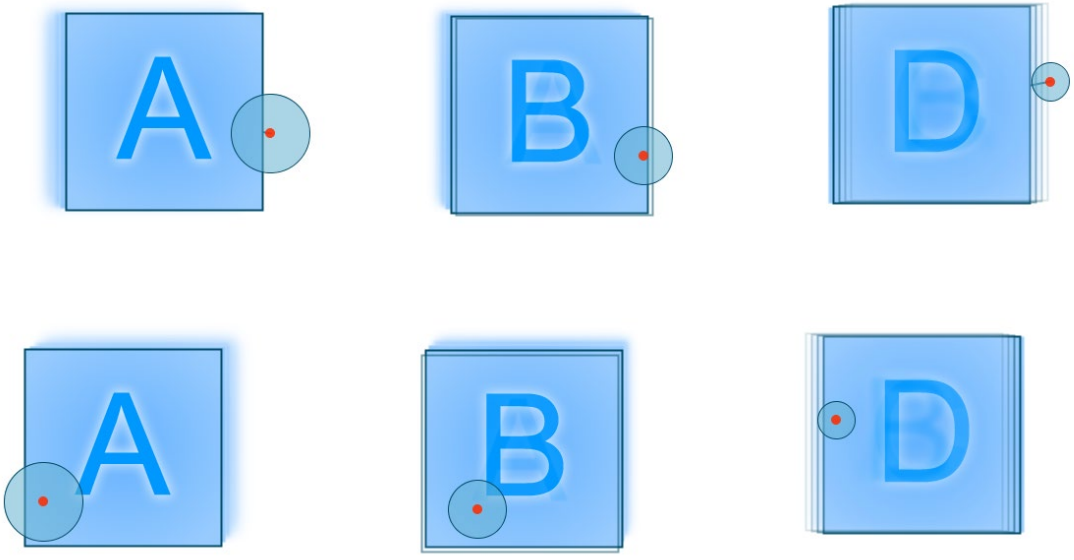
Parallax



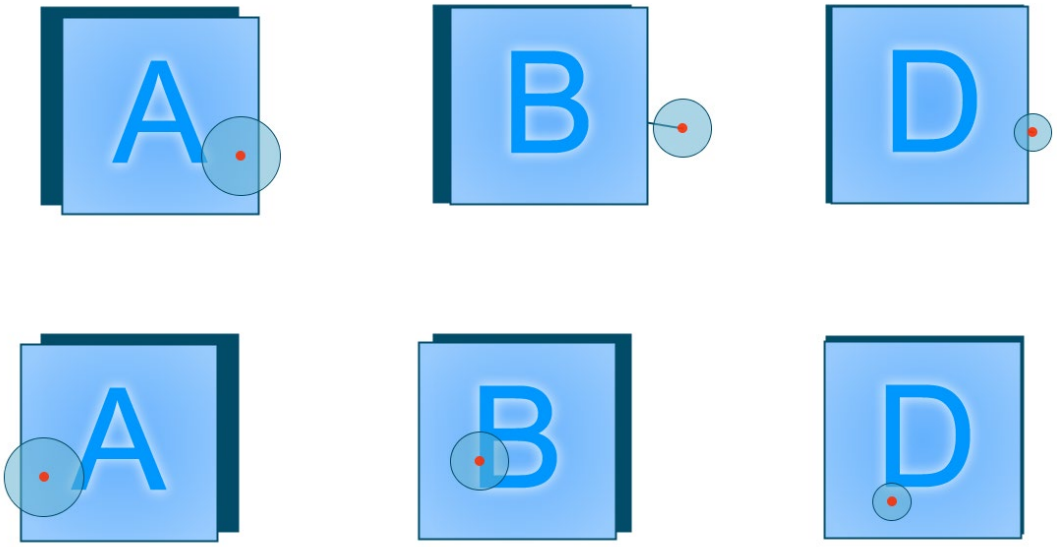
Parallax with Blur



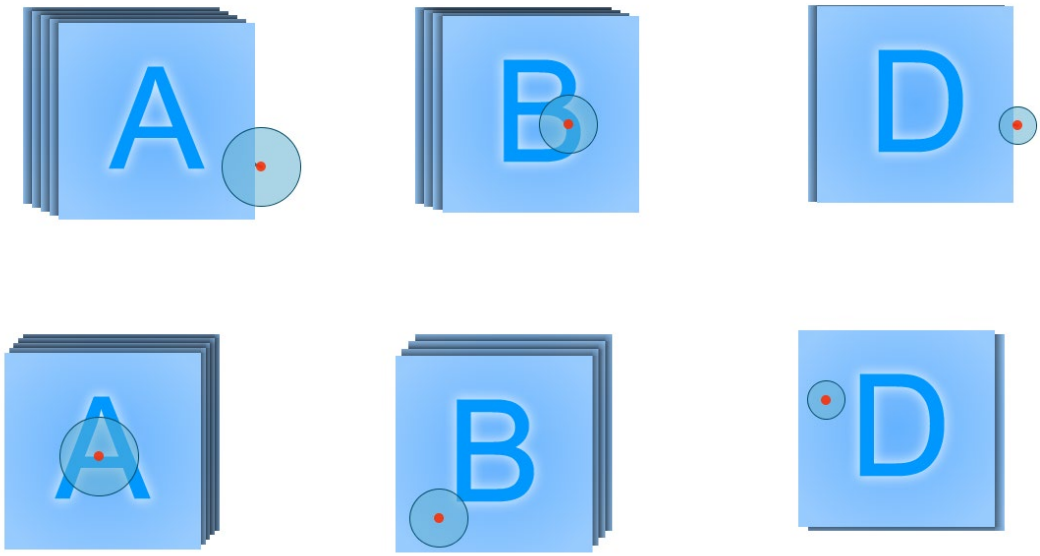
Parallax with Blur and Transparency



Parallax with Shadow



Dynamic Shadow with Parallax



Appendix 2
TestBench – Java Framework for Running Processing-based Experiments

In the process of conducting the user studies, I developed a set of tools to handle tasks common to within-participant studies on multitouch tables using TUIO protocol. These tools were organized into TestBench, a Java framework for running Processing-based[18] experiments with multiple parameters and conditions. The main features and capabilities of the framework are:

- Support of multiple study parameters and conditions
    - Ability to switch between conditions during the experiment
    - Support of unique parameter combinations for each participant (for counterbalancing and to record randomized parameters)
- TUIO listener that enables experimental software to communicate with TUIO-based multitouch devices
    - 2D and 3D protocols are supported to enable both on-surface and hoverspace interactions
    - In the absence of multitouch devices the framework is able to simulate a TUIO cursor using a computer mouse
- Logging/recording of all events happening in the study, including logging of TUIO events
- A set of drawing functions inherited from Processing framework

[18] http://processing.org/

- Several GUI widgets designed for hoverspace interaction

The main classes of the framework are:

- Package *experiments*
  - **TestBench.java**
    - Reads parameters from a text file
    - Keeps track of study conditions
    - Starts the experiment class
  - **AbstractExperiment.java**
    - Contains common experimental functionality
    - Reads current parameters from TestBench class
- Package *hoverspace*
  - **HWTuioListener.java**
    - *Listens to TUIO events*
    - *Stores and records TUIO cursor positions*
    - *Performs various cursor operations: get nearest cursor, get all cursors within square or circle, etc.*
- Package *logging*
  - **ExperimentLogger.java**
    - *Writes into a log file*

A plain-text input file is read by the main class **TestBench.java** when the framework is started. The name of the file can be specified in the code in function *main()*:

```
readParameters("data\\sample.txt");
```

The following line in function *main()*starts the experimental class specified in the input file:

```
PApplet.main(new String[] { "--present", experimentName });
```

The input file is designed to contain all the data required to run an experiment with a single participant; it contains the name of the experimental class, a unique participant ID (for logging purposes), 3 utility variables (Verbose, UseMouse, EnableLogging) and a list of parameter names and values. The main class parses through the input file (ignoring blank lines) and looks for the following pre-defined strings:

- #ExperimentName
    - The name of the experiment class (the class must extend PApplet or AbstractExperiment classes)
- #ParticipantID
    - Unique participant identifier
    - Can be any string
- #Verbose
    - Show/hide debugging console
    - Must be “true” or “false”
- #UseMouse
    - Enable/disable mouse simulation
    - Must be “true” or “false”
- #EnableLogging
    - Enable/disable logging
    - Must be “true” or “false”
- #Parameters
    - Types and names of parameters, separated by commas
    - Must correspond to fields in the experiment class
    - Supported types are: int, float, string, boolean

- #Data
    - Values of parameters named above
    - Each line is a condition in the experiment

After parsing the input file, the main class stores all experimental parameters is a list of conditions, which can be navigated using functions *nextCondition()* and *previousCondition()*.

A sample input file

```
#ExperimentName
experiments.SampleExperiment

#ParticipantID
P1

#Verbose
true

#UseMouse
true

#EnableLogging
true

#Parameters
int intParam, float floatParam, string stringParam, boolean boolParam

#Data
1      0.5    one    true
2      0.33   two    true
3      0.2    three  false
4      1.0    four   false
5      0.001  five   false
```

Note that your own experiment class must extend PApplet class (main class in Processing) or AbstractExperiment class (one of the main classes in TestBench); experiments that extend AbstractExperiment class are able to utilize Verbose, UseMouse and EnableLogging parameters as well as a number of other useful functions.

AbstractExperiment class inherits the functionality of Processing applets such as IO handling, drawing methods, etc. This class is abstract and therefore cannot be instantiated; instead it defines several abstract methods with a postfix *–continuation* that are executed at various stages of applet's lifetime. Another purpose of AbstractExperiment class is to read the current parameters from TestBench class using function *startCondition()*. The main methods of this class are:

- Method *setup()* runs after the applet starts
    - Sets up screen size (1024×768 by default)
    - Reads parameters of the current condition (startCondition() method)
    - Starts logging according to the input file
    - Sets up TUIO listener (by default connected to port 3333)
    - Sets up emulated cursor (mouse position controls x, y coordinates and mouse wheel changes z coordinate)
    - Calls abstract method *setupContinuation()* in the end
- Method *draw()* runs on each frame (at 30fps by default)
    - Updates the position of the simulated cursor to match the position of the mouse
    - Calls abstract method *drawContinuation()*
    - Shows the debugging console
    - Use method *addLineToConsole(String text)* to write to console
- Method *keyPressed()* handles key presses
    - 'm' toggles the simulated cursor
    - '←' and '→' keys switch to previous and next condition accordingly
    - '`' key takes a screenshot of the applet

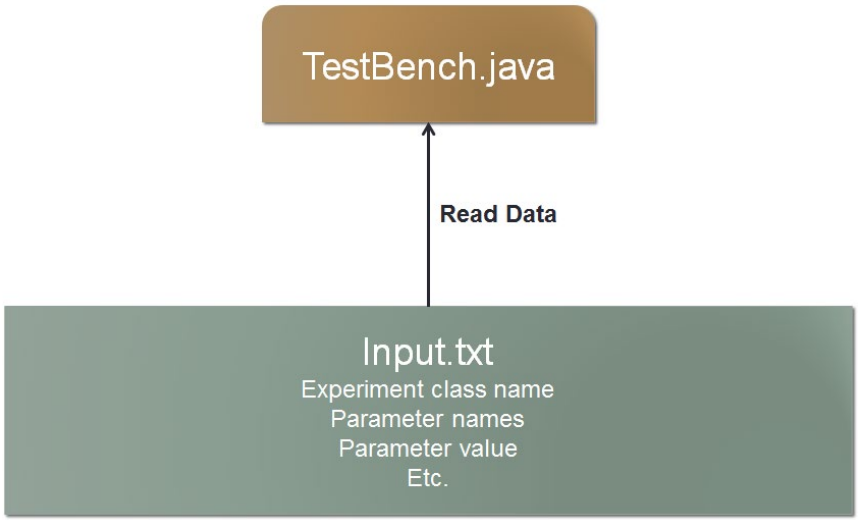
    - Calls abstract method *keyPressedContinuation()* for other keys
- Method *exit()* is called before the applet is closed
    - Calls abstract method *exitContinuation()*
    - Closes the logger
- Method *startCondition()* is called when the condition is switched (using left and right arrow keys, for example)
    - Loads new parameters from TestBench class
    - Calls abstract method *startConditionContinuation()*
- Method *completeCondition()* is called when a participant completes a condition
    - Stops recording
    - Calls abstract method *completeConditionContinuation()*

To write your own experiment class, start by extending AbstractExperiment class, to inherit Processing functionality and the additional functions described above. Override the abstract methods mentioned before to expand the functionality. Note that the study parameters must have the same names and types as the parameters listed in the input file:

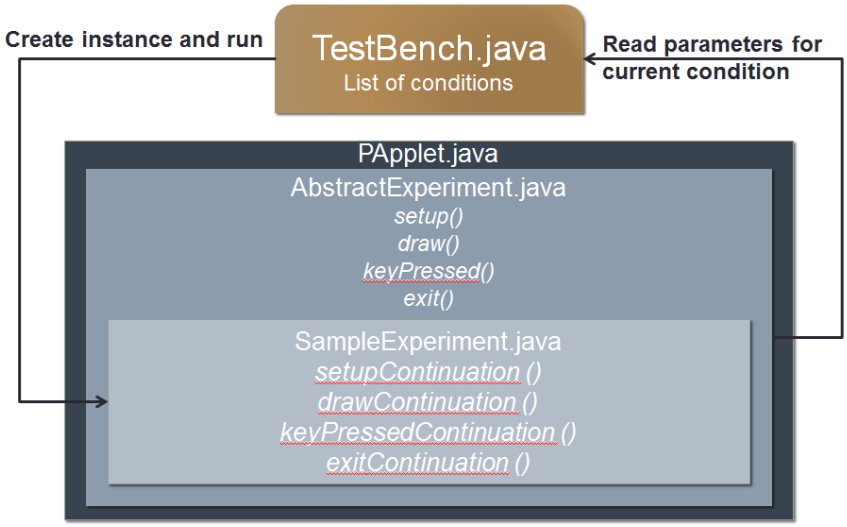
Input file	Experiment class

`#Parameters`
`int intParam, float floatParam, string stringParam, boolean boolParam`
	`// Parameters`
	`public int intParam = 0;`
	`public float floatParam = 0f;`
	`public String stringParam = "";`
	`public boolean boolParam = false;`

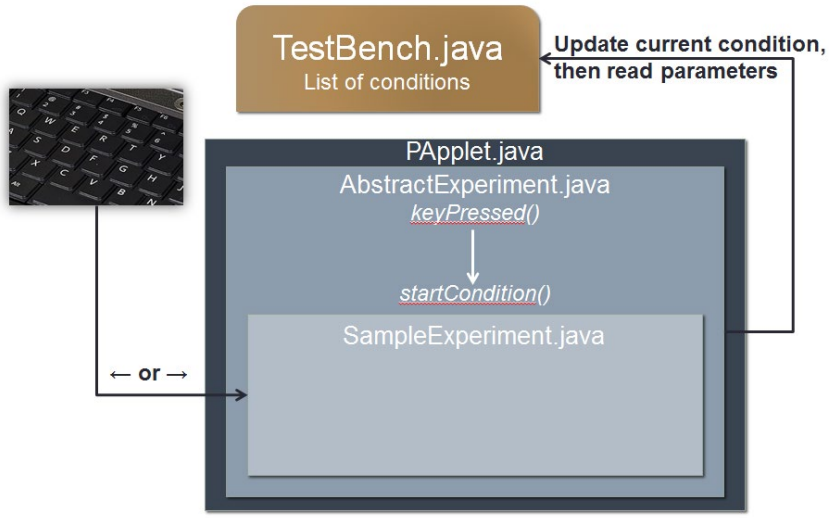
The diagrams below illustrate the flow of data and control in TestBench framework:



First, TestBech class reads in data from a text input file and stores it.



Then, TestBech class creates an instance of an experiment class, which extends AbstractExperiment class and/or PApplet class and runs it. The experiment class implements abstract methods of AbstractExperiment class to extend its functionality.



AbstractExperiment class captures key press events and when left and right arrow keys are pressed, it calls functions *nextCondition()* and *previousCondition()* of TestBench class to change the current condition and then reads the new set of parameters from TestBench.

HWTuioListener class is used for communication with TUIO-enabled multitouch devices. Its main features are:

- Handles add/update/remove TUIO cursor events coming from the multitouch tracker
- Keeps a list of all cursors for the current frame
- Records all TUIO events using ExperimentLogger class
- Is able to record and save gestures as text files using TuioLogger class
- Uses the function *draw(PApplet p)* to draw all current cursors using class HoverspaceCursorVisualization
- Contains several useful functions for fetching certain cursors: *getCursorByID*, *getNearestCursor*, *getAllCursorsWithinSquare*, *getAllCursorsWithinCircle*, *getAllCursors*

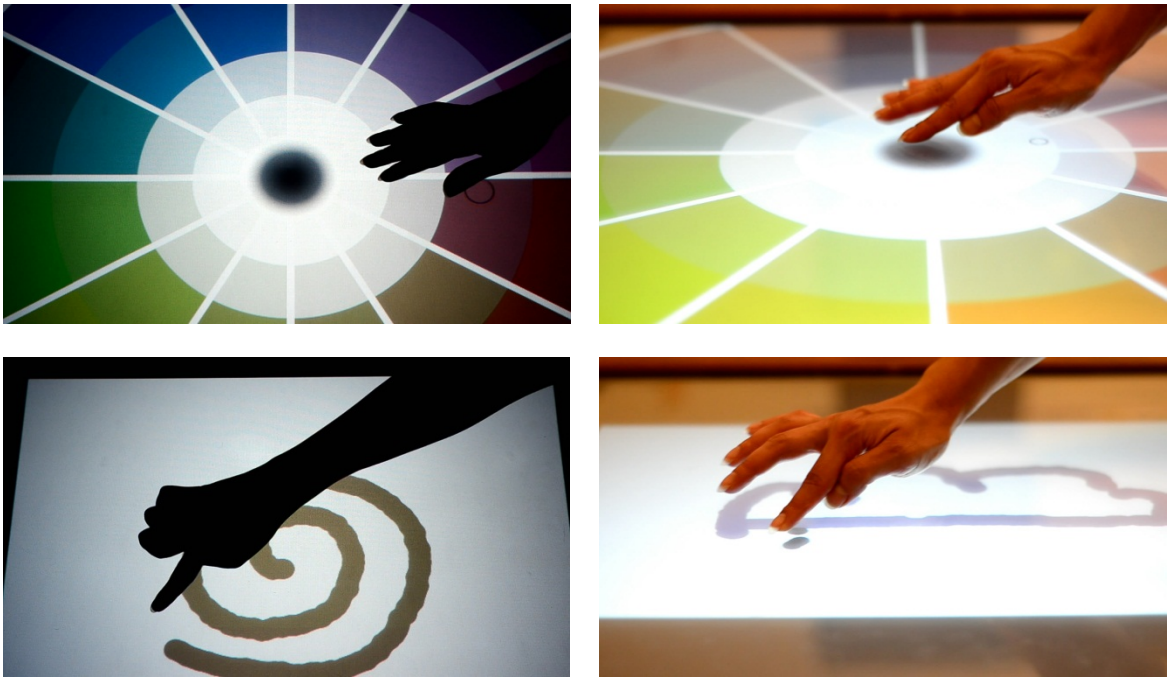
Finally, TestBech framework contains two logging classes: **ExperimentLogger** and **TuioLogger**. ExperimentLogger class creates a log file for each experiment in folder "data/recording" using a timestamp as a name. It can write 5 types of log entries which can be later recognized by log parsers (differentiated by prefixes): Comment, Parameter, TUIO, Event, and Break. Use function *initialize()* to initialize the logger and *close()* when finish writing to log to close the file. TuioLogger is used to record only TUIO events and save them as text files. The output file is formatted in such a way that each line is a trail of a single cursor:

```
Trail <cursorID> [x, y, z, xSpeed, ySpeed, zSpeed, motionSpeed, motionAcceleration, timestamp], […], […],…
```

Appendix 3

Demo Application: HoverPaint

I designed Hover Paint using the results of the two experiments, which allows people to paint on a multitouch surface using their fingers. Hover space interaction is used to control the colour and size of the brush. To activate the colour wheel, a person can lift their hand above the table. The *x* and *y* position of the hand can then be used to control the colour, and the height of the hand can control the brush size; the current selection is displayed as a circular cursor located under the palm. The selection can then be made by moving the hand down quickly and tapping the screen (i.e., the *droptap* gesture).



Screenshots of Hover Paint application

Appendix 4

Study Participant Questionnaires

A. Pre-study background questionnaire

Please fill in information in the following fields:

Your age: _____

Your gender: _____ Male/Female

Are you a student? _____ YES / NO

If yes, then what is the level of your studies: ___ Bachelor / Master/ PhD

Faculty and department: _____

Briefly describe the extent of your familiarity with multitouch interfaces:

If you have any comments about the study, please write them in the space below:

B. Post-study feedback questionnaire

Dwell, Short

It was easy **to move** to the target:	1	2	3	4	5	6	7
It was difficult it **to select** the target:	1	2	3	4	5	6	7
The selection technique is easy **overall**:	1	2	3	4	5	6	7

Dwell, Medium

It was difficult **to move** to the target:	1	2	3	4	5	6	7
It was easy it **to select** the target:	1	2	3	4	5	6	7
The selection technique is difficult **overall**:	1	2	3	4	5	6	7

Dwell, Long

It was easy **to move** to the target:	1	2	3	4	5	6	7
It was difficult it **to select** the target:	1	2	3	4	5	6	7
The selection technique is easy **overall**:	1	2	3	4	5	6	7

Push

It was difficult **to move** to the target:	1	2	3	4	5	6	7
It was easy it **to select** the target:	1	2	3	4	5	6	7
The selection technique is difficult **overall**:	1	2	3	4	5	6	7

Tap Screen

It was easy **to move** to the target:	1	2	3	4	5	6	7
It was difficult it **to select** the target:	1	2	3	4	5	6	7
The selection technique is easy **overall**:	1	2	3	4	5	6	7

Tap Screen, 2 handed

It was difficult **to move** to the target:	1	2	3	4	5	6	7
It was easy it **to select** the target:	1	2	3	4	5	6	7
The selection technique is difficult **overall**:	1	2	3	4	5	6	7

Strongly disagree, disagree, disagree somewhat, neither agree nor disagree, somewhat agree, agree, strongly agree

Appendix 5

Statistical Analysis Details

A. Acquisition Time

a. Tests of Within-Subjects Effects (Factorial ANOVA)

Source		Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta^2
Gesture	Sphericity Assumed	24808321.839	5	4961664.368	5.528	.000	.269
	Greenhouse-Geisser	24808321.839	3.036	8170223.976	5.528	.002	.269
	Huynh-Feldt	24808321.839	3.894	6371352.752	5.528	.001	.269
	Lower-bound	24808321.839	1.000	24808321.839	5.528	.033	.269
Error(Gesture)	Sphericity Assumed	67311671.991	75	897488.960			
	Greenhouse-Geisser	67311671.991	45.546	1477868.166			
	Huynh-Feldt	67311671.991	58.406	1152479.961			
	Lower-bound	67311671.991	15.000	4487444.799			
Position	Sphericity Assumed	1471153.377	1	1471153.377	2.630	.126	.149
	Greenhouse-Geisser	1471153.377	1.000	1471153.377	2.630	.126	.149
	Huynh-Feldt	1471153.377	1.000	1471153.377	2.630	.126	.149
	Lower-bound	1471153.377	1.000	1471153.377	2.630	.126	.149
Error(Position)	Sphericity Assumed	8391452.929	15	559430.195			
	Greenhouse-Geisser	8391452.929	15.000	559430.195			
	Huynh-Feldt	8391452.929	15.000	559430.195			
	Lower-bound	8391452.929	15.000	559430.195			
Gesture * Position	Sphericity Assumed	3948198.048	5	789639.610	1.544	.186	.093
	Greenhouse-Geisser	3948198.048	2.396	1647731.013	1.544	.225	.093
	Huynh-Feldt	3948198.048	2.883	1369424.466	1.544	.218	.093
	Lower-bound	3948198.048	1.000	3948198.048	1.544	.233	.093
Error (Gesture*Position)	Sphericity Assumed	38356389.795	75	511418.531			
	Greenhouse-Geisser	38356389.795	35.942	1067170.597			
	Huynh-Feldt	38356389.795	43.247	886922.388			
	Lower-bound	38356389.795	15.000	2557092.653			

b. Pairwise Comparisons

(I) Gesture	(J) Gesture	Mean Difference (I-J)	Std. Error	Sig.[b]	95% Confidence Interval for Difference[b]	
					Lower Bound	Upper Bound
Dwell, Short	Dwell, Medium	-180.247*	52.943	.004	-293.092	-67.401
	Dwell, Long	-216.644*	86.172	.024	-400.314	-32.973
	Push	-404.976*	90.294	.000	-597.433	-212.518
	Tap	-530.128*	150.038	.003	-849.927	-210.329
	Tap, Off Hand	-592.829*	133.935	.000	-878.306	-307.353
Dwell, Medium	Dwell, Short	180.247*	52.943	.004	67.401	293.092
	Dwell, Long	-36.397	89.780	.691	-227.760	154.965
	Push	-224.729*	97.113	.035	-431.720	-17.738
	Tap	-349.881*	143.688	.028	-656.145	-43.618
	Tap, Off Hand	-412.583*	126.876	.005	-683.012	-142.154
Dwell, Long	Dwell, Short	216.644*	86.172	.024	32.973	400.314
	Dwell, Medium	36.397	89.780	.691	-154.965	227.760
	Push	-188.332	138.045	.193	-482.568	105.904
	Tap	-313.484	175.696	.095	-687.972	61.004
	Tap, Off Hand	-376.185*	166.604	.039	-731.292	-21.078
Push	Dwell, Short	404.976*	90.294	.000	212.518	597.433
	Dwell, Medium	224.729*	97.113	.035	17.738	431.720
	Dwell, Long	188.332	138.045	.193	-105.904	482.568
	Tap	-125.152	186.149	.512	-521.920	271.615
	Tap, Off Hand	-187.854	134.634	.183	-474.820	99.113
Tap	Dwell, Short	530.128*	150.038	.003	210.329	849.927
	Dwell, Medium	349.881*	143.688	.028	43.618	656.145
	Dwell, Long	313.484	175.696	.095	-61.004	687.972
	Push	125.152	186.149	.512	-271.615	521.920
	Tap, Off Hand	-62.701	192.193	.749	-472.352	346.949
Tap, Off Hand	Dwell, Short	592.829*	133.935	.000	307.353	878.306
	Dwell, Medium	412.583*	126.876	.005	142.154	683.012
	Dwell, Long	376.185*	166.604	.039	21.078	731.292
	Push	187.854	134.634	.183	-99.113	474.820
	Tap	62.701	192.193	.749	-346.949	472.352

Based on estimated marginal means

*. The mean difference is significant at the .05 level.

b. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

B. Effect of Position variable on Acquisition Time

a. Tests of Within-Subjects Effects (One-way ANOVA)

Source		Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Position	Sphericity Assumed	60403591.695	2	30201795.847	32.190	.000	.682
	Greenhouse-Geisser	60403591.695	1.583	38151685.434	32.190	.000	.682
	Huynh-Feldt	60403591.695	1.739	34734310.253	32.190	.000	.682
	Lower-bound	60403591.695	1.000	60403591.695	32.190	.000	.682
Error (Position)	Sphericity Assumed	28146882.163	30	938229.405			
	Greenhouse-Geisser	28146882.163	23.749	1185195.520			
	Huynh-Feldt	28146882.163	26.085	1079033.559			
	Lower-bound	28146882.163	15.000	1876458.811			

b. Pairwise Comparisons

(I) Position	(J) Position	Mean Difference (I-J)	Std. Error	Sig.[b]	95% Confidence Interval for Difference[b]	
					Lower Bound	Upper Bound
Middle	Non-dominant	-605.728*	97.920	.000	-814.441	-397.016
	Dominant	-104.315	61.963	.113	-236.387	27.757
Non-dominant	Middle	605.728*	97.920	.000	397.016	814.441
	Dominant	501.414*	78.221	.000	334.688	668.139
Dominant	Middle	104.315	61.963	.113	-27.757	236.387
	Non-dominant	-501.414*	78.221	.000	-668.139	-334.688

Based on estimated marginal means

*. The mean difference is significant at the .05 level.

b. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

C. Jitter Time

a. Tests of Within-Subjects Effects (Factorial ANOVA)

Source		Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta^2
Gesture	Sphericity Assumed	64277793.758	5	12855558.752	10.275	.000	.407
	Greenhouse-Geisser	64277793.758	2.003	32094557.006	10.275	.000	.407
	Huynh-Feldt	64277793.758	2.312	27806813.209	10.275	.000	.407
	Lower-bound	64277793.758	1.000	64277793.758	10.275	.006	.407
Error(Gesture)	Sphericity Assumed	93834924.340	75	1251132.325			
	Greenhouse-Geisser	93834924.340	30.041	3123515.554			
	Huynh-Feldt	93834924.340	34.674	2706222.539			
	Lower-bound	93834924.340	15.000	6255661.623			
Position	Sphericity Assumed	102057.935	1	102057.935	.065	.802	.004
	Greenhouse-Geisser	102057.935	1.000	102057.935	.065	.802	.004
	Huynh-Feldt	102057.935	1.000	102057.935	.065	.802	.004
	Lower-bound	102057.935	1.000	102057.935	.065	.802	.004
Error(Position)	Sphericity Assumed	23617989.206	15	1574532.614			
	Greenhouse-Geisser	23617989.206	15.000	1574532.614			
	Huynh-Feldt	23617989.206	15.000	1574532.614			
	Lower-bound	23617989.206	15.000	1574532.614			
Gesture * Position	Sphericity Assumed	1401724.388	5	280344.878	.260	.934	.017
	Greenhouse-Geisser	1401724.388	1.508	929772.323	.260	.711	.017
	Huynh-Feldt	1401724.388	1.640	854939.610	.260	.730	.017
	Lower-bound	1401724.388	1.000	1401724.388	.260	.618	.017
Error (Gesture*Position)	Sphericity Assumed	80984988.355	75	1079799.845			
	Greenhouse-Geisser	80984988.355	22.614	3581189.063			
	Huynh-Feldt	80984988.355	24.593	3292957.108			
	Lower-bound	80984988.355	15.000	5398999.224			

b. Pairwise Comparisons

(I) Gesture	(J) Gesture	Mean Difference (I-J)	Std. Error	Sig.[b]	95% Confidence Interval for Difference[b]	
					Lower Bound	Upper Bound
Dwell, Short	Dwell, Medium	-195.013*	51.106	.002	-303.943	-86.084
	Dwell, Long	-619.676*	143.681	.001	-925.925	-313.427
	Push	-133.562	63.718	.053	-269.374	2.251
	Tap	-41.625	56.245	.471	-161.509	78.259
	Tap, Off Hand	-911.219*	235.101	.001	-1412.326	-410.113
Dwell, Medium	Dwell, Short	195.013*	51.106	.002	86.084	303.943
	Dwell, Long	-424.663*	147.782	.012	-739.654	-109.673
	Push	61.452	93.127	.519	-137.044	259.947
	Tap	153.388	86.906	.098	-31.847	338.624
	Tap, Off Hand	-716.206*	243.943	.010	-1236.159	-196.253
Dwell, Long	Dwell, Short	619.676*	143.681	.001	313.427	925.925
	Dwell, Medium	424.663*	147.782	.012	109.673	739.654
	Push	486.115*	141.237	.004	185.076	787.154
	Tap	578.051*	157.076	.002	243.251	912.852
	Tap, Off Hand	-291.543	223.384	.212	-767.675	184.589
Push	Dwell, Short	133.562	63.718	.053	-2.251	269.374
	Dwell, Medium	-61.452	93.127	.519	-259.947	137.044
	Dwell, Long	-486.115*	141.237	.004	-787.154	-185.076
	Tap	91.937	68.917	.202	-54.957	238.830
	Tap, Off Hand	-777.658*	223.446	.003	-1253.922	-301.394
Tap	Dwell, Short	41.625	56.245	.471	-78.259	161.509
	Dwell, Medium	-153.388	86.906	.098	-338.624	31.847
	Dwell, Long	-578.051*	157.076	.002	-912.852	-243.251
	Push	-91.937	68.917	.202	-238.830	54.957
	Tap, Off Hand	-869.594*	241.774	.003	-1384.923	-354.266
Tap, Off Hand	Dwell, Short	911.219*	235.101	.001	410.113	1412.326
	Dwell, Medium	716.206*	243.943	.010	196.253	1236.159
	Dwell, Long	291.543	223.384	.212	-184.589	767.675
	Push	777.658*	223.446	.003	301.394	1253.922
	Tap	869.594*	241.774	.003	354.266	1384.923

Based on estimated marginal means

*. The mean difference is significant at the .05 level.

b. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

D. Selection Time

a. Tests of Within-Subjects Effects (Factorial ANOVA)

Source		Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta^2
Gesture	Sphericity Assumed	704830.049	2	352415.025	.372	.693	.024
	Greenhouse-Geisser	704830.049	1.735	406180.971	.372	.664	.024
	Huynh-Feldt	704830.049	1.942	362883.169	.372	.687	.024
	Lower-bound	704830.049	1.000	704830.049	.372	.551	.024
Error(Gesture)	Sphericity Assumed	28444568.321	30	948152.277			
	Greenhouse-Geisser	28444568.321	26.029	1092806.452			
	Huynh-Feldt	28444568.321	29.135	976316.214			
	Lower-bound	28444568.321	15.000	1896304.555			
Position	Sphericity Assumed	156321.265	1	156321.265	.252	.623	.016
	Greenhouse-Geisser	156321.265	1.000	156321.265	.252	.623	.016
	Huynh-Feldt	156321.265	1.000	156321.265	.252	.623	.016
	Lower-bound	156321.265	1.000	156321.265	.252	.623	.016
Error(Position)	Sphericity Assumed	9317767.645	15	621184.510			
	Greenhouse-Geisser	9317767.645	15.000	621184.510			
	Huynh-Feldt	9317767.645	15.000	621184.510			
	Lower-bound	9317767.645	15.000	621184.510			
Gesture * Position	Sphericity Assumed	39373.076	2	19686.538	.044	.957	.003
	Greenhouse-Geisser	39373.076	1.355	29051.052	.044	.901	.003
	Huynh-Feldt	39373.076	1.443	27291.656	.044	.912	.003
	Lower-bound	39373.076	1.000	39373.076	.044	.837	.003
Error (Gesture*Position)	Sphericity Assumed	13525771.510	30	450859.050			
	Greenhouse-Geisser	13525771.510	20.330	665324.178			
	Huynh-Feldt	13525771.510	21.640	625030.656			
	Lower-bound	13525771.510	15.000	901718.101			

b. Pairwise Comparisons

(I) Gesture	(J) Gesture	Mean Difference (I-J)	Std. Error	Sig.[a]	95% Confidence Interval for Difference[a]	
					Lower Bound	Upper Bound
Push	Tap	101.454	147.596	.502	-213.140	416.047
	Tap, Off Hand	-6.661	158.693	.967	-344.907	331.585
Tap	Push	-101.454	147.596	.502	-416.047	213.140
	Tap, Off Hand	-108.115	110.867	.345	-344.421	128.192
Tap, Off Hand	Push	6.661	158.693	.967	-331.585	344.907
	Tap	108.115	110.867	.345	-128.192	344.421

Based on estimated marginal means

a. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

E. Target Lost Count

a. Tests of Within-Subjects Effects (Factorial ANOVA)

Source		Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta^2
Gesture	Sphericity Assumed	406.711	5	81.342	12.947	.000	.463
	Greenhouse-Geisser	406.711	2.618	155.349	12.947	.000	.463
	Huynh-Feldt	406.711	3.222	126.248	12.947	.000	.463
	Lower-bound	406.711	1.000	406.711	12.947	.003	.463
Error(Gesture)	Sphericity Assumed	471.210	75	6.283			
	Greenhouse-Geisser	471.210	39.271	11.999			
	Huynh-Feldt	471.210	48.323	9.751			
	Lower-bound	471.210	15.000	31.414			
Position	Sphericity Assumed	1.816	1	1.816	.181	.677	.012
	Greenhouse-Geisser	1.816	1.000	1.816	.181	.677	.012
	Huynh-Feldt	1.816	1.000	1.816	.181	.677	.012
	Lower-bound	1.816	1.000	1.816	.181	.677	.012
Error(Position)	Sphericity Assumed	150.789	15	10.053			
	Greenhouse-Geisser	150.789	15.000	10.053			

	Huynh-Feldt	150.789	15.000	10.053			
	Lower-bound	150.789	15.000	10.053			
Gesture * Position	Sphericity Assumed	14.127	5	2.825	.368	.869	.024
	Greenhouse-Geisser	14.127	1.385	10.197	.368	.620	.024
	Huynh-Feldt	14.127	1.481	9.537	.368	.633	.024
	Lower-bound	14.127	1.000	14.127	.368	.553	.024
Error Gesture*Position	Sphericity Assumed	575.614	75	7.675			
	Greenhouse-Geisser	575.614	20.781	27.700			
	Huynh-Feldt	575.614	22.218	25.908			
	Lower-bound	575.614	15.000	38.374			

b. Pairwise Comparisons

(I) Gesture	(J) Gesture	Mean Difference (I-J)	Std. Error	Sig.[b]	95% Confidence Interval for Difference[b]	
					Lower Bound	Upper Bound
Dwell, Short	Dwell, Medium	-.487*	.141	.004	-.787	-.186
	Dwell, Long	-1.233*	.329	.002	-1.934	-.532
	Push	-.028	.188	.883	-.429	.372
	Tap	.058	.212	.788	-.395	.511
	Tap, Off Hand	-2.250*	.482	.000	-3.277	-1.223
Dwell, Medium	Dwell, Short	.487*	.141	.004	.186	.787
	Dwell, Long	-.747*	.312	.030	-1.411	-.082
	Push	.459	.276	.117	-.130	1.047
	Tap	.545	.267	.059	-.024	1.114
	Tap, Off Hand	-1.763*	.492	.003	-2.812	-.714
Dwell, Long	Dwell, Short	1.233*	.329	.002	.532	1.934
	Dwell, Medium	.747*	.312	.030	.082	1.411
	Push	1.205*	.357	.004	.443	1.967
	Tap	1.291*	.410	.007	.417	2.166
	Tap, Off Hand	-1.017*	.437	.034	-1.948	-.085
Push	Dwell, Short	.028	.188	.883	-.372	.429
	Dwell, Medium	-.459	.276	.117	-1.047	.130
	Dwell, Long	-1.205*	.357	.004	-1.967	-.443
	Tap	.086	.260	.745	-.468	.641

	Tap, Off Hand	-2.222*	.465	.000	-3.212	-1.231
Tap	Dwell, Short	-.058	.212	.788	-.511	.395
	Dwell, Medium	-.545	.267	.059	-1.114	.024
	Dwell, Long	-1.291*	.410	.007	-2.166	-.417
	Push	-.086	.260	.745	-.641	.468
	Tap, Off Hand	-2.308*	.515	.000	-3.405	-1.211
Tap, Off Hand	Dwell, Short	2.250*	.482	.000	1.223	3.277
	Dwell, Medium	1.763*	.492	.003	.714	2.812
	Dwell, Long	1.017*	.437	.034	.085	1.948
	Push	2.222*	.465	.000	1.231	3.212
	Tap	2.308*	.515	.000	1.211	3.405

Based on estimated marginal means

*. The mean difference is significant at the .05 level.

b. Adjustment for multiple comparisons: Least Significant Difference (equivalent to no adjustments).

F. Trial Failed Count

a. Cochran's Q Test

	Value	
	0	1
Dwell, Short	51	45
Dwell, Medium	77	19
Dwell, Long	86	10
Push	69	27
Tap	84	12
Tap, Off Hand	79	17

N	96
Cochran's Q	50.282[a]
df	5
Asymp. Sig.	.000

a. 1 is treated as a success.

b. Post-hoc McNemar Test

	N	Chi-Square[a]	Asymp. Sig.	Exact Sig. (2-tailed)
Dwell, Short & Dwell, Medium	96	14.881	0.000	
Dwell, Short & Dwell, Long	96	26.884	0.000	
Dwell, Short & Push	96	6.568	0.010	
Dwell, Short & Tap	96	21.787	0.000	
Dwell, Short & Tap, Off Hand	96	15.848	0.000	
Dwell, Medium & Dwell, Long	96			.093[b]
Dwell, Medium & Push	96	1.361	0.243	
Dwell, Medium & Tap	96			.210[b]
Dwell, Medium & Tap, Off Hand	96	0.033	0.855	
Dwell, Long & Push	96	8.828	0.003	
Dwell, Long & Tap	96			.824[b]
Dwell, Long & Tap, Off Hand	96			.230[b]
Push & Tap	96	6.323	0.012	
Push & Tap, Off Hand	96	2.382	0.123	
Tap & Tap, Off Hand	96			.405[b]

a. Continuity Corrected

b. Binomial distribution used.

G. Participant Preferences

a. Friedman's Tests

Movement

	Mean Rank
Dwell, Short	3.67
Dwell, Medium	4.27
Dwell, Long	3.90
Push	3.37
Tap	3.47
Tap, Off Hand	2.33

Selection

	Mean Rank
Dwell, Short	4.20
Dwell, Medium	3.70
Dwell, Long	2.37
Push	3.27
Tap	3.97
Tap, Off Hand	3.50

Overall

	Mean Rank
Dwell, Short	4.33
Dwell, Medium	3.27
Dwell, Long	2.37
Push	3.53
Tap	3.67
Tap, Off Hand	3.83

Test Statistics

N	15
Chi-Square	17.509
df	5
Asymp. Sig.	.004

Test Statistics

N	15
Chi-Square	10.719
df	5
Asymp. Sig.	.057

Test Statistics

N	15
Chi-Square	12.164
df	5
Asymp. Sig.	.033

b. Post-hoc Wilcoxon's Tests

	Movement		Selection		Overall	
	Z	Asymp. Sig. (2-tailed)	Z	Asymp. Sig. (2-tailed)	Z	Asymp. Sig. (2-tailed)
Dwell, Medium - Dwell, Short	-1.186[b]	.236	-1.569[b]	.117	-2.401[b]	.016
Dwell, Long - Dwell, Short	-.962[b]	.336	-2.722[b]	.006	-2.631[b]	.009
Push - Dwell, Short	-.499[c]	.618	-1.143[b]	.253	-1.282[b]	.200
Tap - Dwell, Short	-.647[c]	.518	-.241[b]	.809	-.925[b]	.355

Tap, Off Hand - Dwell, Short	-2.112[c]	.035	-1.243[b]	.214	-.986[b]	.324
Dwell, Long - Dwell, Medium	-.577[c]	.564	-2.371[b]	.018	-1.459[b]	.145
Push - Dwell, Medium	-2.041[c]	.041	-.052[c]	.959	-.905[c]	.366
Tap - Dwell, Medium	-1.802[c]	.072	-.924[c]	.356	-.838[c]	.402
Tap, Off Hand - Dwell, Medium	-2.684[c]	.007	-.569[b]	.569	-.948[c]	.343
Push - Dwell, Long	-1.807[c]	.071	-2.249[c]	.024	-1.451[c]	.147
Tap - Dwell, Long	-1.378[c]	.168	-2.302[c]	.021	-1.486[c]	.137
Tap, Off Hand - Dwell, Long	-2.732[c]	.006	-.955[c]	.340	-1.451[c]	.147
Tap - Push	.000[d]	1.000	-.633[c]	.526	-.103[b]	.918
Tap, Off Hand - Push	-2.043[c]	.041	-1.028[b]	.304	.000[d]	1.000
Tap, Off Hand - Tap	-2.041[c]	.041	-1.119[b]	.263	-.155[c]	.877

b. Based on positive ranks.

c. Based on negative ranks.

d. The sum of negative ranks equals the sum of positive ranks.

Publications from this Thesis

Material, ideas, figures, and tables from this thesis have appeared previously in the following peer-reviewed publications:

Pyryeskin, D., Hancock, M., and Hoey, J. Extending interactions into hoverspace using reflected light. Proc. ITS '11, ACM Press (2011), 262–263.

Pyryeskin, D., Hancock, M., and Hoey, J. Comparing Elicited Gestures to Designer-Created Gestures for Selection above a Multitouch Surface. Proc. ITS '12, ACM Press (2012), accepted in September 2012.