Handling Ambiguous User Input on Touchscreen Kiosks

by

Christopher K. Leung

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2005

Author .....
Department of Electrical Engineering and Computer Science
May 19, 2005

Certified by .....
Larry Rudolph
Principal Research Scientist
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Handling Ambiguous User Input on Touchscreen Kiosks

by

Christopher K. Leung

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2005, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Touchscreen kiosks are becoming an increasingly popular means of providing a wide arrange of services to the public. However, the principal drawback of these types of systems lies within the elevated error rates due to finger imprecision and screen miscalibration. These issues become worrisome, considering the greater responsibilities and reliance placed upon touchscreens.

This thesis investigates two novel techniques that attempt to alleviate these interaction problems. The first technique, predictive pointing, incorporates information regarding past interactions and an area cursor (which maps the user's touch to a circular area rather than a single point) to provide a better estimate of the intended selection. The second technique, gestural drawing, allows users to draw particular shapes onscreen to execute actions as an alternative means of input that is largely unaffected by issues of miscalibration.

Results from a user study indicated that both techniques provided significant advantages in not only lowering error rates, but also improving task completion times over traditional tasks of target selection.

Thesis Supervisor: Larry Rudolph
Title: Principal Research Scientist

Acknowledgments

I would like to thank my thesis advisor, Larry Rudolph, for his assistance and guidance throughout the course of our work and the development of this thesis. Without his help, I would be still an MEng student as you read these pages.

I would also like give my loving thanks to my gorgeous girlfriend Rita Jou. She always believed in me and was there with an encouraging word whenever I needed it. She also helped me design the experiments used to test out some of the techniques, and sat in lab with me for many hours patiently just reading a magazine or shopping on the internet. Her being there for me meant a lot, and without her help, I'd probably still be a grad student too.

I'd also like to thank my some of my good friends: first, Ishan Sachdev. He indirectly helped me towards finishing my thesis as well. Whenever I was feeling behind, I could just turn around and see him reading ESPN and remind myself that at least there were people who were more behind than I was. Plus, he was usually there to help me out if I got stuck with a problem; Ram Woo, he didn't help me get this done, since he was so far ahead of me that I lost motivation several times. Satish Ramaswamy, who I can thank for countless meals left over from his group meetings, which he never ate. He miraculously finished his thesis while no one was looking and managed to go play golf all the time.

Also, I want to acknowledge my pledge class of 2004 at DU. Unquestionably the best group of guys I've known, even though they didn't have anything to do with this thesis.

I also want to thank Atish Nigam. Unbeknownst to him, I've been using his thesis as a reference for some formatting and organizational ideas. Whenever he comes back to MIT and looks at his thesis and sees that a couple people checked it out, that was just me. Sorry Atish.

In terms of people I don't know, I would like to thank the Boston Celtics, the Red Sox, the San Diego Padres, the other NBA playoff teams, and the makers of the Friends and Sopranos TV shows. Collectively, you were the biggest detriments

towards the completion of my thesis, but you made things interesting and gave me so many much-needed "breaks" between not working on my thesis.

Finally, I would like to thank my loving family for their support. My mom in particular always made sure I was on top of my stuff, which of course I always was. Thanks to my dad for being so laid back but supportive, and my sisters who I don't talk to enough but are always there for me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As service kiosks rapidly become more prevalent in today's increasingly fast-paced world, the accuracy of their use become more important as well. Kiosks can be found as ticket dispensers at movie theaters, financial service provider ATM's, health-monitoring systems at hospitals, and check-in terminals at airports. Kiosks serve a cheap, quick, and convenient method of delivering services and information to the general public, and these finger-operated touchscreens provide a natural and intuitive method of input for novice users. As touchscreens become more common, users will depend on them more heavily and come to demand more of these machines. Although the trend of kiosk popularity has continued for some time, touchscreens[1] are still notorious for their inaccuracy. Even slight problems of miscalibration can lead to consistent errors in selection. These problems of inaccuracy can lead to user frustration, and if ignored, can offset the advantages that a simplistic touchscreen interface provides. This thesis explores two methods that attempt to reduce the limitations and improve the usability of touchscreen kiosks.

1.1 Background

The public access kiosk was first developed in 1977 at the University of Illinois at Urbana-Champaign, and was designed to help students obtain information about

[1]For the remainder of this thesis, the words *kiosk* and *touchscreen* will be used interchangeably.

various student organizations on campus.[4] Remarkably, despite most users' unfamiliarity with computers at the time, over 50,000 users interacted with the touchscreen system within the first 45 days of its unveiling, providing evidence of the touchscreen's ease-of-use as a natural interface. The kiosk represented a huge revolution within the realm of graphical user interfaces.

When kiosks are coupled with touchscreens, they provide a way for users to interact with the underlying system intuitively through simple pointing and button-pressing actions with their fingers. Despite the simplicity of this type of interaction, several weaknesses of touchscreens limit their use for many applications. The principal limitation, in most cases, is the method of interaction itself. While pointing at buttons onscreen is instinctive to most users, the design of an effective user interface for touchscreens that accurately responds to taps of a finger presents a challenging problem.

With the explosive increase in the prevalence of personal computers, most people have become familiar and skilled at using keyboards and mice to interact with their PC's. With kiosks, the methods of user input are severely limited, and often consist of only a touchscreen. While improvements are continually being made to speech interpretation and motion gesture recognition technologies on kiosks, the most common and simplest method for kiosk user input remains pointing with one's index finger. This greatly restricts the user's possible actions to pressing buttons/clicking locations on the touchscreen. This mode of interaction is considerably less accurate than a mouse, which was designed to accurately select items several pixels in size with a very low error rate. With kiosks, the range of error is nearly the size of a fingertip surface, which may be the distance of half an inch or more[53]. On top of this, slightly miscalibrated touchscreens can make these situations even more problematic. These facts imply that high precision pointing and clicking is not always a feasible option. This issue leaves us with an essential question: how does one design a user-friendly kiosk system that overcomes the shortcomings of a finger as an input device?

1.2 Motivation

Self-service touchscreen kiosks represent a rapidly growing sector of industry. The number of players in the interactive kiosk market has risen dramatically from 67 firms in 1996 to over 530 in 2003. Touchscreen executives have estimated that the industry has expanded from a $1 billion to $4 billion industry during the course of 2004.[7] According to another study of companies in the kiosk industry, the number of worldwide kiosk units was estimated at just under 400,000 in 2003, and has expectations to grow nearly 86% to approximately 740,000 units through 2007.

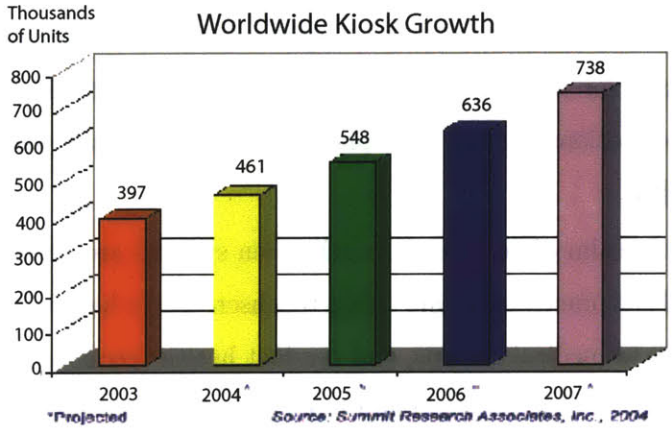

Figure 1-1: Projected Growth of Kiosks through 2007

A similar market analysis study from Frost & Sullivan in 2003 estimated that kiosks were a $2.94 billion industry, and are growing at "a formidable pace", particularly within the retail services industry.[6] A primary reason for the meteoric growth of the kiosk market may be because kiosks provide a way for users to access information and complete transactions in a "non-threatening manner". The primary segment where kiosks are successful is within the retail industry, which accounts for over 50% of the kiosk market. Many companies, including Wal-Mart, have deployed kiosks at their stores after successful pilot testing periods with consumers.

Although retail companies account for the majority of kiosk usage, kiosks are expanding to other segments as well, including healthcare, governmental services, and entertainment.

Kiosks are quickly crossing over into the vending-machine market. Starbuck's intends to deploy kiosks that burn music CDs[35], and McDonald's is in the process of testing kiosks that will allow consumers to rent DVD's.[8] Kiosks are already immensely popular as ticketing systems for transportation, entertainment, and sports ticket sales[64], and are steadily emerging in quick-serve restaurants, where users can take orders and accept payment[11]. These self-service machines are being piloted by many quick-server restaurants, including McDonald's, Subway, and Burger King.

Certain retailer are utilizing touchscreens as a part of the job applications process, allowing employers to check national databases and improve the quality of their applicants. Wal-Mart in particular has begun using over 7,000 machines in the past year specifically for this purpose, and have employed nearly 30,000 kiosks in total, to handle applications dealing with photos, gift registry, cake decoration, and even deli-ordering[64].

Within the realm of healthcare, touchscreen systems are becoming more popular as well. Medical practitioners are using touchscreens to help with patient check-in, provide access to medical records, and conduct health screening surveys. Kiosks can also provide a convenient means of informing patients about important health issues or answering common questions regarding treatment and procedures. Specialized kiosks also include touch-sensitive testing equipment that monitor body temperatures, heart rate, and blood pressure.[3] Attempts have also been made to utilize touchscreen technology for the benefit of allowing disabled persons to access the Internet. These attempts have shown that even among mentally disabled people, touchscreens were still considered very natural and usable.[61]

Touch-based kiosks can also provide a myriad of government-based services. Certain applications, such as driver testing at the DMV, fingerprint recognition in high-security locations, and point-of-sale systems in restaurants are becoming popular. Biometrics technology is making kiosks useful for quickly identifying frequent visitors from foreign countries. One such system, US-VISIT, is already being deployed at border crossings to allow common travellers to bypass the conventional interview process.[1] In addition, touchscreen systems have already begun to make an impact

for secure voting.[2]

Figure 1-2: An Undesirable Situation: Touchscreens Making the Decisions

It is apparent that kiosks are being used to handle increasingly important applications. However, it is not clear that the accuracy of touchscreens are sufficient to consistently perform in these vital applications. According to a recent article chronicling the latest presidential election[41], voters have found problems with early touchscreen systems designed for electronic ballot casting. Several users noticed that although they had made their selections for one candidate, the touchscreen was registering the vote for the other candidate. Although a number of issues may have caused those problems, it is apparent that more accurate touchscreen interaction is necessary. These problems of imprecision become worrisome considering the greater responsibilities and reliance placed upon touchscreens.

1.3 Traditional User Interaction

Graphical user interfaces make interacting with computers much easier, providing a rich visual feedback to users attempting to interact with the computer. The most common interface used today are called WIMP interfaces (interfaces based on windows, icons, menus, and pointing actions), and allow users to achieve a variety of tasks with a few button selections.

This task of button selection can be broken down into two distinct parts. In the first part, the user maneuvers the mouse over the visual representation of the button (which is characterized by a particular shape and location) and clicks on it. In the second part, the underlying O/S receives this event, and if the click resides within the same virtual shape and location, the callback associated with that button widget is executed. Traditionally, there is a direct mapping between these two tasks. Any click within the button widget automatically results in the execution of the callback associated with it. Although this one-to-one mapping is simple, there are advantages in breaking apart this direct mapping. If some amount of processing may be performed between these two levels of interaction, target selection may be improved.

1.4 Goals

In this thesis, we explore the benefits of separating these two parts, and inserting an intermediate level between them that allows for improved interaction. By allowing an additional layer of computation between the application and the operating system, we obtain a more powerful scheme for processing user interaction. We have focused specifically on the design and analysis of two techniques that alleviate problems of sloppiness and imprecision when dealing with kiosk touchscreens. The precision of such touchscreens are often limited by not only the low "resolution" of the user's finger, but problems of miscalibration as well. New methods of reducing error rates (and user frustration) due to these two issues would be invaluable to the continued proliferation of touchscreens. Our approaches adds an additional level of computation between the application and operating system that improves the accuracies of interaction.

1.5 Outline

Chapter 2 of this thesis will discuss relevant background regarding touchscreens and the current state of the art, including similar work researched within the context

of performance-enhancing interaction techniques. Chapter 3 will examine the importance of improving touchscreen interaction, and discuss the implementation and design strategies for the two such techniques. In chapter 4, we will analyze the effectiveness of the described enhanced user techniques through the results of a controlled user study, and in chapter 5, we discuss the implications of these results. This thesis will conclude in chapter 6 with a look at shortcomings encountered with these techniques, and a description of avenues for future work.

Chapter 2

Background and Related Work

This chapter describes the benefits and disadvantages of touchscreens, as well as varied attempts in the past to improve precision of pointing devices.

2.1 Introduction

Nearly all modern human-computer interfaces revolve around a cardinal action–pointing. In most cases, this act of pointing consists of a user maneuvering a cursor to a specified location with an input device (i.e. a mouse), and engaging the target(e.g. a button) with a click. This compound action represents the most popular and intuitive method of user interaction across most systems. On touchscreens, this compound action of maneuvering the finger and touching the screen are combined into a single one.

There are limitations to this paradigm. Within the context of user interfaces, there must be a balance between the size of available targets and the amount of information displayed onscreen. Larger targets are easier to acquire, but take up more screen real estate, and thus prompts additional navigation. If smaller targets are displayed onscreen, this allows more information to be displayed, but also inhibits the user's ability to work with smaller icons.

2.2 Fitts' Law

Fitts"s Law is a model of human psychomotor behavior developed in 1954. This law dictates a formal relationship between speed and accuracy of a rapid, aimed movement (i.e. moving a mouse cursor to a button onscreen). According to Fitts' calculations, the time it takes for a user to acquire a target is a logarithmic function of the distance to and the size of the target:

$$M = a + b \cdot log\{D/W\}$$

where M is the movement time, a and b are empirically determined constants dependent on the input device used, D is the distance of movement from the initial location to the center of the target, and W is the width of the target. The log term above is often noted as the index of difficulty (ID). When Fitts' Law is interpreted from a mathematical relationship into real-life consideration in the design of usable GUI's, it implies that:

- Larger targets are generally acquired more quickly than small targets;
- Closer targets are acquired more quickly than farther targets;
- The ID increases by one unit through a doubling of the distance to target or a halving of the target width;

2.3 Problems with Fitts' Law

Fitts' Law has inherent flaws when considering tasks that involve movement in two dimensions, since Fitts' original experiments were one-dimensional tasks. Within the scope of two-dimensional target acquisition tasks, several other interpretations of the effective width were considered[37]:

- Discarding the height and consider only the width;
- Interprets the width as the sum of the height and width of the target;

- Interprets the width as the area of the target;
- Selects the smaller of the height and width;
- Uses the length of the line between the points where the target boundaries intersect the approach line of the cursor;

Research has shown that selecting the smaller of the height and width provides the best correlation over the other proposed models [37] in estimating the index of difficulty. Despite these shortcomings, Fitts' law does allow us to model user performance in rapid, aimed movements for two-dimensional tasks. We can apply the implications of this law to help design more usable user interfaces, especially within the context of touchscreen kiosks. Several additional general principles exist when applying Fitts' Law to designing layouts for GUIs:

- Actions which are performed more frequently should be assigned a larger-sized target. Although this concept seems intuitive, having a panel of buttons of different sizes detracts from consistency of the interface, so this principle must be carefully applied.
- Actions which are performed more frequently should also be easy for the user to reach, so that popular targets are closer to the center of the screen. This principal must be carefully applied as well, since an interface layout arranged dynamically based on frequency of button/widget use may be counterintuitive to the user who may expect a logically-arranged layout.

2.4 Benefits of Kiosks Touchscreens

There are a wide variety of pointing devices that may be used to interact with computers. Some of these devices are better suited for certain types of interaction. For instance, for most desktop machines, a mouse or trackball is often the most convenient and effective means of input. In most laptops, touchpads or pointing sticks are the favored devices, since these built-in pointing devices require little to no maintenance,

and do not contain any moving parts that might break through normal wear and tear with a mobile machine. However, with public kiosks, where users often don't need such sophisticated methods of input, touchscreens are ideal.

None of the other devices are as natural to use as the touchscreen. Even outside the context of computer interaction, pointing with our fingers is an instinctual, natural method of indicating interest. People point to direct attention towards a certain direction, and to indicate selection. Touchscreens enable software designers to take advantage of this natural convention, and allow the users to physically select items of interest directly, without maneuvering an intermediate device such as a mouse. Several studies have shown [17][50] that touchscreens are the fastest method of input compared with mice and keyboards. This ease-of-use contributes towards the end of convenience. Users can go through a transaction at their own pace, and avoid long lines for service with a human agent by stepping up to a kiosk.

Oftentimes, information kiosks and other electronic displays are used by people who have little to no experience with computer interaction. For these types of people, user-friendly touch-screens may be less intimidating than other input devices. When information is presented simply and logically, even inexperienced users can quickly navigate their way through a transaction or to a specific piece of information by touching the display and following the directions.

Additionally, since touchscreen-operated kiosks are operated in public spaces (for the most part), there are several other important factors that come into play when considering benefits of kiosks. First, touchscreens do no require any additional space that another type of input device would require (a mousepad or surface for the mouse to be maneuvered). Second, touchscreens are very durable and have no moving parts, and thus must be replaced less frequently than other types of devices. Third, touch-screens may be built in to the kiosk, so that it is much more difficult to simply remove the input device.

2.5 Fallbacks of touchscreens

Despite the numerous upsides of touchscreens, there are several drawbacks as well. The principal difficulties of touchscreen interaction are outlined below.

2.5.1 Imprecision

Perhaps the greatest criticism of touchscreens is the fact that it is often difficult to precisely select items, since the area of the fingertip is relatively large, when compared to the cursor pointer of the mouse. This makes it difficult to select precise pixels on the screen, given the large error factor of the fingertip. The size of the pointer (in this case the finger) also partially obscures visual feedback behind the finger, and may lead to errors in selection.

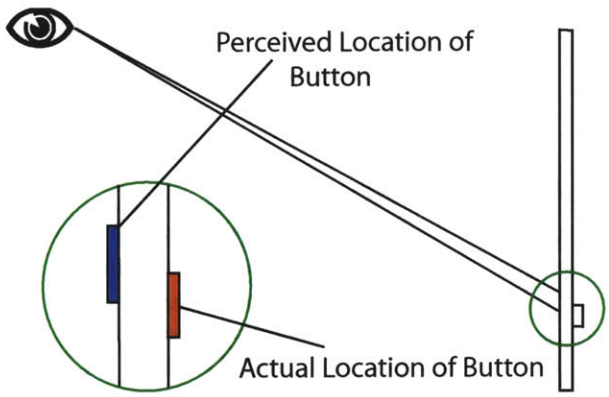

Figure 2-1: Selection Imprecision Caused By Parallax

On top of this, it is can often be difficult for users to precisely maneuver their fingers to precise locations due to the phenomenon of parallax. (Parallax occurs if a significant difference exists between the physical location of a touch by the user and the registered location of the touch by the system. This can be attributed to the fact that the user is operating the touchscreen at a viewing angle, rather than straight-on.)

2.5.2 Miscalibration

The proper calibration of a touchscreen is vital to functionality. Mice do not need to be calibrated; the location of the cursor translates exactly to the location of the click. On touchscreens, miscalibration is a frequent and important problem, and the location of the touch does not always reflect the location of the eventual click.

When a touchscreen is properly calibrated, a touch on the sensor device is interpreted at the same location as on the underlying display screen. However, this alignment between the touch sensor and the screen is not always perfect. Because of this, most touchscreens allow the user to calibrate the machine during startup, so that the devices are correctly aligned. However, if these devices are not properly calibrated, this poses a significant problem. When touchscreens are not properly aligned, they consistently interpret input incorrectly, skewing it to one direction or another. Even slight miscalibrations can cause the users to unintentionally make undesired selections, which can be very aggravating.

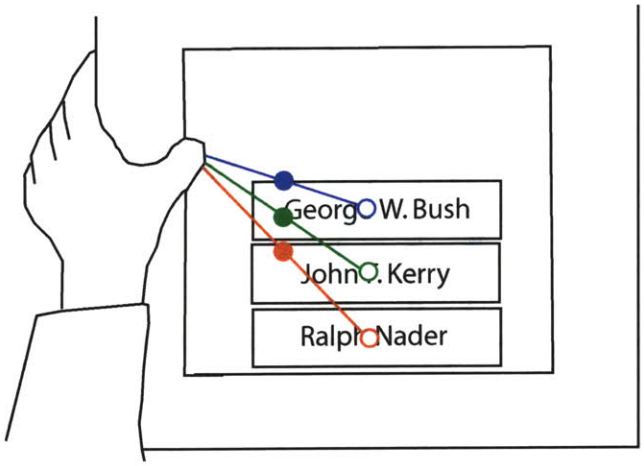

Figure 2-2: Errors caused by Accidental Miscalibration

While this type of miscalibration is troublesome, it can often be avoided by rerunning the calibration step. (Although a more robust solution would be preferred over recalibrating the touchscreen for every user.) A more intractable problem arises from accidental miscalibration. This situation emerges when users grasp the edge of the touchscreen with one hand whilst using the other hand to interact with it. If

the user's thumb accidentally rests upon the screen, this effectively "throws off" the touchscreen, as diagrammed in Figure 5-3.

While a vote for George Bush (in above figure) would likely be registered correctly by the system, votes for John Kerry or Ralph Nader are likely to be interpreted incorrectly. The system takes the locations of the two touches given, and produces an average location as the interpretations. Clearly, in most situations of this variety, the touchscreen will not behave correctly. This problem may also arise if the user's palm (or one of their other fingers) touches the screen simultaneously. Oftentimes, the users do not notice themselves accidentally touching the screen while interacting with the system, which can leave users frustrated and confused. These are common problems that occur with touchscreens[38], and must be resolved.

2.5.3 Fatigue

Users may also experience fatigue whilst operating a touchscreen, since there may be no place for their hand to rest while interacting with they system. This is especially true if the user is standing up. A user with a tired arm is less likely to make selections on the kiosks as accurately, and could easily lead to frustration, and may even force the user to give up and walk away without having completed his transaction. User fatigue can be reduced by decreasing decision-making and searching tasks.

2.5.4 Limited Input Ability

Most kiosks only provide interaction in the form of onscreen buttons that may be pressed. When compared with the variety of inputs currently available on a desktop computer, the kiosk pales in comparison. Careful design principles must be applied in order to effectively layout buttons and screen sequences, and even then, the user must methodically plod through the menus, even if they are frequent users of the system. There is no difference between the kiosk interaction of a novice user and an experienced user. Novice users must be carefully guided through a transaction or query, but experienced users know exactly what to expect, and thus their interaction

could be enhanced greatly by allowing them to complete their transactions as swiftly as possible. If kiosks are to become a more prevalent technology in the future, they must be improved to the point where information can be quickly and effortlessly accessed, for both experienced and novice users of the system.

Although these weaknesses of touchscreens may seem trivial, they are crucial to the success of kiosk touchscreens. If a user experiences frustration due to fatigue, or a high error rate whilst interacting with a kiosk, they will be much more likely to avoid kiosk interaction in the next time around, and perhaps opt for some other method of inquiry or transaction. The principal redeeming quality of touchscreen kiosks lie in the fact that they provide the user with a quick and convenient means of obtaining information or completing a simple transaction. If this process is aggravated due to poor performance of the touchscreen during the interaction, the whole purpose of the kiosk is defeated.

While all of the aforementioned pitfalls are important to consider, this thesis deals specifically with techniques that attempt to reduce finger imprecision, errors from miscalibration, and limited input ability. Although fatigue is an important issue to address, it is not a focal point of this thesis. Several past approaches to alleviating these issues, as well as improving target acquisition within GUIs in general, are discussed below.

2.6 Dealing with Selection Imprecision

Graphical user interfaces make interacting with computers much easier, providing rich visual feedback for users attempting to interact with the computer. The most common interface used today are called WIMP interfaces (interfaces based on windows, icons, menus, and pointing actions), and allow users to achieve a variety of tasks with a few mouse clicks at choice locations. One major issue that GUI designers must tackle is striking a balance between making GUI targets accessible and at an appropriate size, while trying to present as much information coherently as possible. Invariably, as more information is displayed on the screen, the icons must be made smaller to

compensate for lack of space. There have been several approaches taken in the past to help improve the acquisition of small targets (such as buttons) on user interfaces, which are discussed below. Unfortunately, many of these strategies are focused around using the mouse as input devices, but they provide guidance towards what strategies may work on touchscreen kiosks.

2.6.1 Simple Approaches

When considering this problem of target acquisition, three naive solutions are possible, and are discussed below.

One common method of improving target acquisition involves lining icons or widgets around the borders of the screen (i.e. Windows Taskbar[9]). This technique makes target acquisition easier since users do not have to decelerate the pointing device to stop on the target, since the cursor is bounded by the edge of the screen. This makes it easier for users to select these buttons, as it effectively makes one of the dimensions of the widget infinite in size, and impossible to overshoot. While this approach may be effective in some cases, not all icons can be placed along the side of the screen due to limited space. Not only would this be a non-traditional (and therefore likely to be a non-intuitive) GUI layout, this strategy does not work for touch-screen kiosks, since the user's finger does not have a bounding box. Placing targets near the border of the screen may cause users to inadvertently touch the frame of the screen instead of the active touchscreen region. This approach also fails to respond to the concerns of miscalibration, fatigue, and limited input ability.

Another method that has had mixed success is arranging button positions to account for popularity[5]. This has been implemented in menu bars, where more popular actions are displayed near the top, and the menu bar dynamically re-sorts actions as the user interacts (the more popular a particular choice is, the closer to the top of the menu bar it is). Although at first, this approach may seem to be beneficial, many users were frustrated when widgets would move around, increasing the error rate. This technique destroys the consistency of the GUI, and does not address problems with imprecision, miscalibration, or input ability.

A straightforward solution is making targets bigger. The use of larger icons on-screen reduces the index of difficulty, and allows users to make selections more quickly and accurately. Making icons (and buttons in general) larger reduces the amount of information displayable onscreen, and more user navigation is necessary. This is an undesirable feature, because additional navigation and selection tasks undoubtedly lead to a greater probability of error during the course of a transaction, in spite of the benefit larger icons may provide. Although this attempt generally tries to improve problems of imprecision, miscalibration and fatigue, displaying larger buttons and icons is not always feasible, since it locks up precious screen space.

The above methods also are somewhat impractical, since they require considerable changes to the user layout, which may result in a decrease in the efficiency of interactions. In addition, none of these approaches attempt to add an additional layer of computation between the application level and operating system. Any individual application must employ the above techniques in order to obtain their benefits. The following methods are more sophisticated, and explore the possibilities available to improve user interaction once this layer of computation is added.

2.6.2 Goal Crossing

Crossing-based interfaces are a newer development intended to replace traditional pointing tasks within GUIs today. Traditional pointing tasks have users manipulate the cursor to stop over a widget and click on it. Goal-crossing is achieved by positioning the cursor on one side of an icon and moving the cursor over the icon to the other side. Accot and Zhai [16] discovered that goal-crossing does in fact conform to Fitts' Law, although the width is essentially infinite because users do not have to stop the cursor, and a "click" occurs instantaneously upon the completion of the goal-crossing.

MacKenzie [42] analyzed the effectiveness of another similar technique, called the Stroke-Through method. This method is characterized by a mouse-button down action on one side of the target icon, followed by a dragging of the cursor through the icon, and finally a mouse-button release on the other side of the icon. According

to a theoretical model to determine Fitts' constants, the Stroke-Through method was found to be 40% faster than the traditional point and select, despite the fact that the user has to deal with the additional task of holding down one or more of the mouse buttons. Even though the Fitts' Law empirical constants a and b are increased noticeably for goal-crossing, the infinite value of the effective width more than compensates this effect.

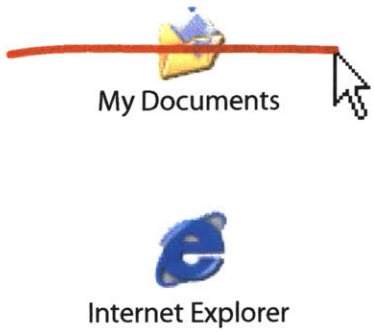

Figure 2-3: An Example of a Goal-Crossing Task

There is one important problem with this goal-crossing technique when applied directly to touchscreens. In order for goal-crossing to be implemented successfully, there must be sufficient empty space on either side of each icon so that the user's finger can land on this empty space. If the icons are clustered together, this would force the user to touch an adjacent icon in order to execute the goal-crossing task. To preclude this possibility, the layout must space the buttons further apart, which decreases the amount of screen space available for buttons (and other widgets), a clearly undesirable feature. Although this technique marginally improves problems with miscalibration and imprecision, it actually restricts input ability even more than conventional methods by forcing targets to take up more space.

2.6.3 Target Gravity/Sticky Targets

Another method of making target acquisition easier is the development and the use of "sticky icons" or semantic pointing. When users move their cursor over a sticky icon,

the gain of the cursor movement is reduced, so that it is easier for users to stop the cursor on the icon, thus reducing target acquisition time [67] [58]. With a lower gain, a larger physical movement of the pointing device is necessary to move the cursor an equivalent distance as compared with a higher gain. This approach effectively makes the icons larger without taking up additional space onscreen. Blanch et al performed a number of experiments with semantic pointing and determined that the index of difficulty of a pointing task is dependent on the size of the target in motor space, as opposed to the visual size of the target.

Another method similar to this concept involves use of warping gravity wells. Gravity wells can be thought of as a larger circle that surrounds the target icon or widget. This surrounding circle represents a region where a gravity towards the icon acts on the cursor. Once the user moves the cursor within this circle, it becomes subject to a "gravitational" pull towards the center of the icon, and is automatically moved towards the center of the widget. These gravity wells have been shown to improve target acquisition time by 10-20%. [54][36]

It is evident that this strategy does not work well on touchscreens (except for those that implement the lift-off strategy), since on many kiosk touchscreens, the cursor is invisible to the user. When the cursor is invisible, the users do not try to maneuver the cursor around the screen, but rather force the cursor to jump from place to place onscreen with each successive touch. This type of interaction precludes the possibility of adjusting the cursor gain or adding a "gravitational pull" on the cursors to enhance precise target acquisition. This technique, applied directly, fails to improve upon any of the inherent drawbacks of touchscreens. However, the concepts behind sticky targets may provide some benefits, since this technique utilizes historical user interaction to better estimate probable actions. This idea may allow us to similarly approximate intentions based on the frequency of past actions.

2.6.4 Area Cursors

Area cursors are a cursors that operate with a larger activation area. While most cursors can effectively click only a single point onscreen, area cursors are designed to

"click" on an area of points. A few prior studies have shown that the performance of area cursors for target acquisition tasks is better than the performance of normal single-point cursors, especially in the case of selecting small targets. Worden et al conducted a study that showed that area cursors significantly improved movement times in target acquisition tasks [67].

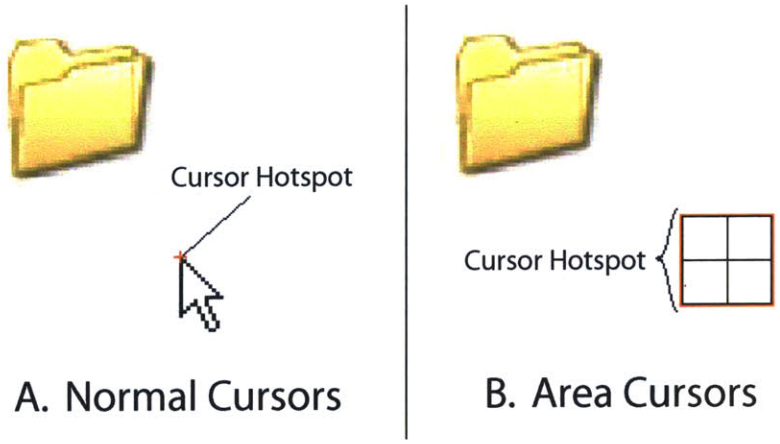

Figure 2-4: The active regions of a normal cursor and an area cursor

Area cursors are not as effective at selecting icons or widgets that are located very close to one another. In most cases, area cursors "react unpredictably" if a cursor highlights two distinct widgets simultaneously. Fortunately, area cursors do provide a possible avenue of improvement within the realm of touchscreens. Since it is difficult to pinpoint a precise point on a touchscreen, a larger area cursor (the size of a finger) may be a more appropriate method of modelling a user's fingertip. This cursor expansion may help immensely in alleviating imprecision and miscalibration, by providing a larger active area. However, further refinements must be carefully made in order to take advantage of area cursors, without also acquiring the problems that area cursors create (such as simultaneous activation of targets).

2.6.5 Enlarging Targets Dynamically

One prevailing approach to improving target acquisition has been the expansion of UI targets dynamically. Research has shown that this target expansion technique

reduced pointing times even if target expansion occurs very late into the movement [48], or if target expansion occurs unexpectedly from the viewpoint of the user [68]This technique can be applied to one of several types of user interface widgets, including toolbars, buttons, or icons. When a user moves the cursor near or on top of these widgets, they grow in visual size. Expanding the size of the widgets allows users to select them more easily when they are interested in selecting them, while consuming less visual space when users are not interested in selecting them.

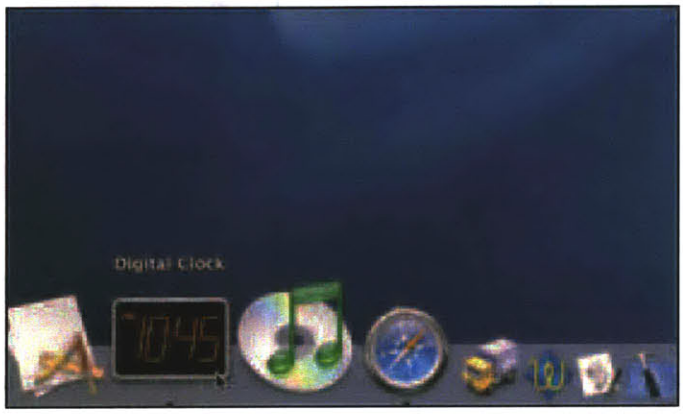

Figure 2-5: A screenshot of the popular Mac OS X Dock Toolbar and enlarging icons

One particularly popular instance of this approach can be seen on the MacOS X Dock (Figure 2-5). This Dock is a toolbar located on the desktop, which is filled with the icons of a number of frequently-used applications. When the users move the mouse over an icon located on the Dock, the icon enlarges in size, pushing nearby icons further away.

There are a few difficulties that arrive with this type of technique. It can be difficult for users to select targets accurately on the MacOS Dock since the icons move when users attempt to move their cursor onto the toolbar at an angle. [28] Additionally, this technique doesn't prove useful within the realm of improving kiosk user interfaces, since users do not usually see the cursor, but rather interact with the system by pressing virtual buttons onscreen. Thus, there is no way for users to "mouse over" a target, and so this technique is not directly applicable to kiosks. However, the concept of dynamically enlarging targets does provide improvements

to generally combat issues of imprecision and miscalibration. In addition, enlarging targets do help to alleviate user fatigue, as the index of difficulty of acquiring targets is reduced.

2.7 Issues with these Approaches

There have been a number of novel techniques investigated before towards alleviating the problem of target selection. However, the crucial difference between these strategies and a successful one for touchscreens relies on the absence (on touchscreens) of cursor position feedback. Although touchscreens have an advantage over other input devices as more natural and direct, they also suffer from the lack of feedback as to the location of the cursor. With other devices, such as mice or trackballs, users receive constant visual feedback from the screen as they move their cursors. As the users maneuver the input device, they can see the exact position of the cursor. With touchscreens, users are not provided any feedback until the touch is made.

It is possible to display a cursor on the display of a touchscreen. However, given the direct nature of touchscreens, allowing users to see the touchscreen forces the user to subconsciously focus their efforts on attempting to maneuver the cursor, rather than selecting objects of interest directly. However, research has shown[53] that it is possible to make very precise selections (with targets smaller than the finger size but larger than 4 pixels). Nevertheless, this approach takes considerably longer than direct touchscreen interaction, since dragging times take longer than pointing times, due to the additional task of holding the finger to the screen.

2.8 Touchscreen-Specific Strategies

There have been far fewer attempts in past research to improve the precision of touchscreen specifically. However, the approaches that follow provided crucial insight towards feasible touchscreen strategies.

2.8.1 Simplistic Touch Strategies

A study by Potter et al [60] described a technique called *Take-Off*, where users would drag the touchscreen cursor to the desired target, and lifts the finger off the screen to select the target. When compared with the traditional *Land-On* method (where the user simply touches the desired location to select the target), *Take-Off* was significantly slower, but was considerably more accurate. The researchers also documented the case of another technique, *First-Contact*, which was a combination of the ideas of both *Take-Off* and *Land-On*. With *First-Contact*, users made selections by dragging their fingers along the touchscreen. Once the cursor makes contact with a selectable item, that item is automatically selected by the system. The performance of this technique fell in between those of the *Land-On* and *Take-Off*, with respect to both speed and error rate.

While these strategies were insightful, they were not deemed as advantageous over the simplistic *Land-On* strategy, since both techniques described took longer than the *Land-On* approach.

2.8.2 High Precision Touch Strategies

An interesting study to note revolves around the investigation of several novel strategies that attempted to improve high precision touchscreen interaction.[53] These strategies were demonstrated to have success with selecting individual pixels on touchscreens without the aid of zooming.

The two high-precision techniques developed were *Cross-Keys* and *Precision-Lever*. The *Cross-Keys* technique establishes a rough starting point when the user first touches the screen. Once this starting point is established, a crosshair appears, flanked by four graphical arrow keys. The users utilize these arrow keys to maneuver the crosshair over the target, if necessary, and tap over the crosshair to make the selection.

The *Precision-Handle* technique employs the use of a handle and crosshair. The user's first touch on the screen results in the placing of the crosshair and the handle.

The handle can be dragged in any direction, which maneuvers the crosshair at the tip on a much smaller scale, which increases the precision of this technique.

Both of these techniques worked well for the selection of targets as small as 1 or 2 pixels in size, with relatively low error rates (less than 8%). However, despite the very high precision of these techniques, they are only useful for applications where pixel-level precision is necessary. In addition, the mean completion times for *Precision-Handle* and *Cross-Keys* are over 2.5 seconds per target acquisition, which take significantly longer than completion times for *Land-On*, which average less than half the time (1.182 seconds per target)[60].

These high-precision techniques require the constant visual feedback from specialized cursors designed to allow precise control with a user's finger, which eliminates problems of miscalibration. While these techniques work well for the selection of small targets, they do not provide practical benefit for hasty users, since each target acquisition task takes several seconds. This lengthened task completion time exacerbates the level of user fatigue, which may offset any benefits of precision. Nevertheless, important practical concepts can be gleaned from both of these approaches.

The following chapter of this thesis delves into two novel approaches to improving user interaction with touchscreens. Both of the developed methods utilize insights gleaned from prior work done on improving target acquisition on traditional GUIs as well as touchscreens in particular.

Chapter 3

Problem Description and Design Strategies

We have examined past efforts at improving user interaction, and have strived to extract the advantageous properties of these strategies while eliminating their deficiencies. Ideally, touchscreen interaction should try to reduce the effects of imprecision, miscalibration, user fatigue, and limited input ability. This chapter describes our attempts at developing new interaction techniques that look to resolve these problems.

The ultimate goal of kiosks is to provide the general public access to a universally available shared resource. Kiosks should allow people to obtain information or perform a relatively simple transaction with great ease. Most users will attempt to perform highly similar, but personalized transactions. There are numerous transactions that occur each day that could be easily handled by a well-designed kiosk (purchasing tickets, finding directions, or ordering items), and touchscreens are the ideal method of interaction–they require no external devices, and most users can operate them naturally. Many of these types of transactions may not only be automated, but they can be more cost-effective for service providers and not susceptible to human error (on the service side) at the same time. However, kiosks cannot eliminate human error completely, as long as people are interacting with them. This chapter explores several methods of coping with error and incorporating a degree of "sloppiness" into interpreting users' actions.

Fingers are not precise input devices. Although fingers are very skilled at certain tasks, they are not meant to be precise pointing devices. Users can use their fingers to indicate the general area of interest very accurately and quickly, but when the sizes of areas of interest fall below a certain threshold (the size of the surface of the finger), fingers are wildly unpredictable. When this task of small target acquisition is coupled with a slightly miscalibrated screen, user interaction can be nightmarish.

These observations have resulted in a more precise model for fingers. Fingers come with inherent unpredictability with regard to this respect, and that unpredictability can never be eliminated. However, it is possible to design systems that can accommodate these shortcomings. In order to take advantage of the strengths of fingers, it is necessary to understand the actions fingers are intended for, which include pointing, touching, and drawing. How is it possible to develop a more robust input system to accommodate the "sloppiness" of the finger? The following sections probe two interaction techniques that attempt to respond to this question.

3.1 ATM's as a Paradigm for Kiosks

Financial institutions know the benefits of self-service technologies like ATM's and online banking: they reduce costs, provide greater flexibility and help customers to help themselves. Not surprisingly, interactive kiosks have become increasingly popular for many of the same reasons.

Many people are very familiar with automatic teller machines (ATM's). Not only do ATM's help save people from a trip to the bank, they save the banks enormous amounts of money as well–there are no employees to pay, and the cost of maintaining a fully functional ATM is significantly lower than that of maintaining a full-service bank branch. ATM's serve as the model of optimization for self-service kiosks today–they allow users to complete a specific type of transaction quickly, simply, and successfully. ATM's, and most kiosks in general, were designed with the convenience of the user in mind.

Now, with the "Web-enabling" of ATM's, these self-service kiosks may begin to

offer a much wider array of services than before. This services could include more advanced bank services such as online bill payment or the ability to view cancelled checks, as well as peripheral services such as ticket purchasing for events, personalized stock quotes, sports scores, or even directions.

However, even with these additional service ATM's and other kiosk-type machines will provide, the paradigm of kiosks will remain the same: speed and convenience. Despite the expected expansion of services, kiosks will always be intended for quick transactions, regardless of the nature of the transaction.

Considering all the benefits an ATM provides for both the banks and consumers, the model for transactional kiosks is slowly spreading to include other types of services. Nearly all domestic airlines now provide self-service kiosks for flight check-in, cutting down passengers' waiting times and aggravation. Many metropolitan subway systems also allow passengers to purchase passes through kiosks. Grocery stores across the nation provide a self-checkout service for customers in place of standard cashiers. Soon, highly specialized kiosks will become the standard for these types of transactions.

As these types of ultra-convenient, high-speed kiosks become more popular, users will demand more of the kiosk. In the future, most people can be expected to interact with kiosks on a daily basis. As users become more accustomed to interacting with these kiosks, they will undoubtedly become more proficient at using them. In turn, users will attempt to complete the transaction as quickly as possible. This, of course, introduces a higher probability of errors. Users eager to quickly navigate through a transaction will undoubtedly make more mistakes in item selection than a careful, novice user. Inexperienced users tend to make their input selections more deliberately and carefully, which often results in accurate, but slow transactions. Experienced users will maneuver through screens fairly quickly.

In effect, as the current technology exists today, the more proficient a user becomes with interacting with kiosks, the more he is punished, since quicker maneuvering through menus causes a greater error rate. One particular problem of interest that arises is how to deal with ambiguous input to the system. How should these kiosk

systems handle these types of inputs? Fingertip inaccuracy coupled with a quick interaction invariably leads to mistakes. And as kiosks slowly become more powerful (and transactions become more important), these types of mistakes have potential to become more costly. How can we develop a system that can cope with these mistakes? Is it possible to enhance touchscreen interaction with more robust methods of input?

To develop such a solution that enables proficient users to navigate their transaction more quickly, the accuracy of interaction must be improved. At a typical ATM, over 95% of users attempt to perform one of three transactions: make a withdrawal, make a deposit, or conduct inquiries about their balance or bank account.

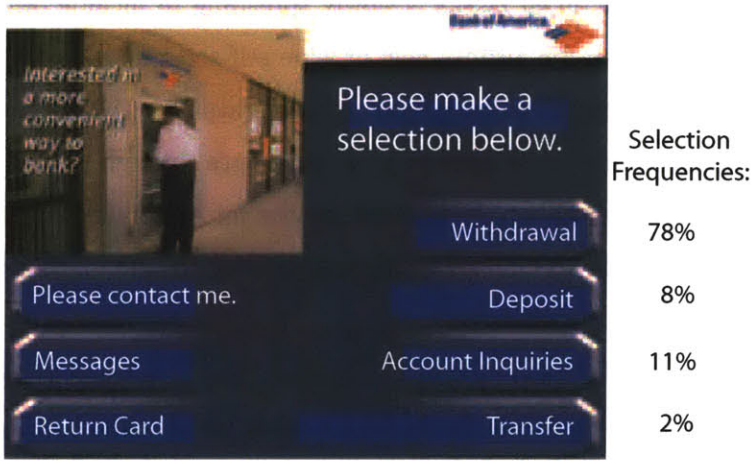

Figure 3-1: A typical ATM screen

While there are other possible transactions users may complete at ATM's, withdrawals, deposits, and account inquiries are the most common. Based on the large number of past transactions with ATM's, we can develop a probabilistic model of user interaction. For example, if 80% of ATM users withdraw money, 10% make deposits, and 10% check their balance, we can develop a model that incorporates those statistics when helping users make a selection. Thus, if a user makes an ambiguous touch on the screen between the withdrawal and deposit buttons, we can assume that around 88.9% (0.8/0.9) of the time, the user is trying to select withdrawal, and the other 11.1%, the user is trying to select deposit. In this case, we can assume, based on all past user actions, that the user is most likely trying to withdraw money, and

take that course of action.

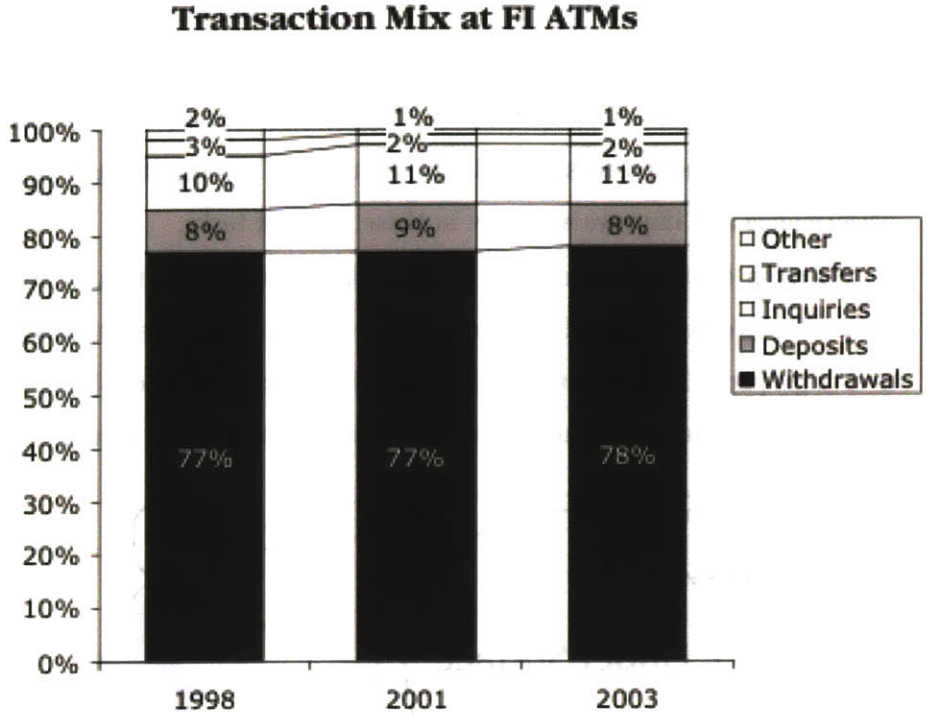

Figure 3-2: ATM Usage Breakdowns

Based on past information of all users, we can make a prediction about what this particular user is attempting to do during this particular transaction. However, we can improve upon the accuracy of this method by storing personalized information about each user. If, for example, this particular user never made withdrawals at ATM's, but frequently made deposits, we could take this into account as well, and if an ambiguous input is made, the system can "guess" that the user intended to select the deposit button, rather than the withdrawal button. Although the concept is simple, it may improve the accuracy of prediction.

The following sections describe two techniques designed to provide more robust methods of interaction, modelled specifically to account for the imprecise nature of fingers. Predictive pointing aims to reduce the effects of finger imprecision due to finger size, while finger gestures aim to eliminate the effects of miscalibration.

3.2 Predictive Pointing

Predictive pointing is a straightforward method of estimating the true intended selection of the user. This technique takes into consideration the issues discussed in the previous section. As mentioned before, touchscreen kiosks are particularly prone to errors during selection tasks. Predictive pointing is a design feature that aims to aid performance and improve accuracy in these types of tasks.

Selection mistakes arise when users make a touch a portion of the screen that borders or overlaps multiple buttons. While traditionally, these touches are mapped to a single pixel that lies within only one of the buttons, it is possible to make a better estimate as to the correct intentions of the user. Consider the situation in Figure 3-3.

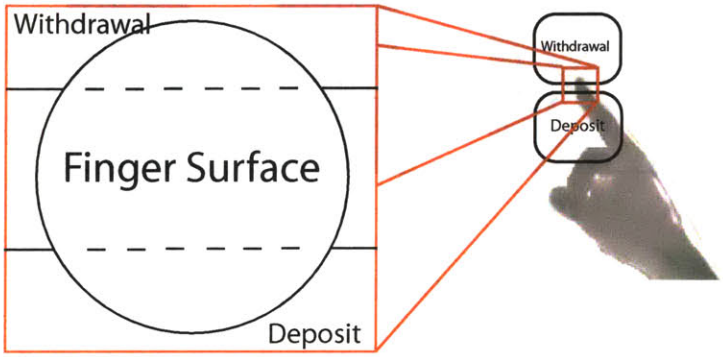

Figure 3-3: An Instance of an Ambiguous Touchscreen Selection

In the situation diagrammed, conventional systems will invariably interpret the touch as a virtual click on a single pixel. In this case, the selected pixel will lie in either Button A, Button B, or the unclaimed space between the buttons. However, if the system has a record of past interactions, it is possible to determine the most likely selection of the user, instead of simply relying solely on the calibration of the touchscreen to interpret the result. For example, if from past interactions of these cases, Button B was selected 80% of the time, and Button A was selected 20% of the time, using this information to determine the most likely interaction would provide some insight as the probable intention of the user. Button B is four times as likely to be selected as Button A in this scenario.

In addition to this, it is necessary to take into account the position of the finger. In this thesis, it is assumed that the actual pixel location registered by the system represents the center point of the circle formed from the surface of the finger touching the screen. Any point within this circle could be the intended selection point. By taking into account both of these pieces of information, a more accurate likelihood estimate can be made.

In essence, predictive pointing models the effect of a fingertip touching a surface more practically. When a user touches the surface of the screen, it makes less sense to map that action as equivalent to a mouse click, since the resolution of the fingertip is much larger than that. With predictive pointing, a touch onscreen is mapped to a circular area that the user has selected, rather than a point. This method of interpreting for a fingertip touch seems much more appropriate, modelling the fingertip as an area cursor.

During the course of implementation, two possibilities were originally considered when determining the shape of the area cursor—a rectangle or a circle. Intuitively, a circular area cursor appeared to be the best choice because it modelled the shape of the finger surface more accurately than any other shape. However, it proved to be difficult to determine the overlap area with circles because the overlapping areas between the circular area cursor and most targets were irregular shapes. It was not immediately clear how the area of the shapes could be approximated.

There are several distinct types of overlap between a circle and a rectangle, and in many cases, the area of overlap was complicated to compute. At this point, the possibility of the rectangular area cursor was explored. Although a rectangle did not approximate the finger area surface shape as well as a circle did, there appeared to be little downside to using them. Calculating the area of overlapping rectangles was very straightforward.

However, we ultimately decided to return to the circular model for finger surface area. Instead of coming up with a complicated algorithm to determine the area of overlap, we decided to take a cleaner route. To determine the area of overlap, several hundred points were randomly selected from within the circle, by choosing a random

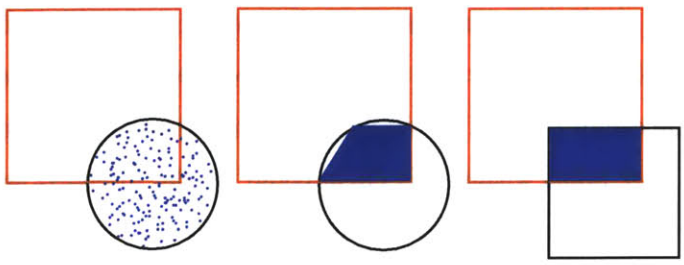

Figure 3-4: The Three Models considered for determining Overlap Area

angle from 0 to 2π radians and a random distance from the center of the circle. The approximate overlap area was determined by finding the proportion of these points that also fell within the boundaries of the rectangular target. This method proved to be extremely accurate, and relatively easy to implement.

Unfortunately, complications can arise with the area cursor method of interpreting finger touches. With single-point interpretations, each effective "click" will always fall within the same target, since the active area consists of a single pixel. With an finger-area interpretation, the active area is much larger. If the active area solely falls within a single target area, the course of action is straightforward. However, with this approach it is possible that an "area click" will trigger multiple targets simultaneously. This poses a realistic problem, since on many kiosks, buttons are located together on a panel, or clustered together on a toolbar onscreen. With ambiguous touches near borders between adjacent buttons, it is difficult to determine which of the buttons was intended to be hit.

A predictive pointing approach handles these ambiguous situations by incorporating information about known probabilities of selection from past user interactions. Each time a finger touch is registered, the system checks to see if the active area overlaps multiple targets. If multiple targets are within the active region, the system then takes into account the "weight", or popularity of each target. These two factors, the overlap area and the weight, are combined to form a likelihood score for each target. These scores are compared, and the most likely target is selected.

Although this technique may be able to provide a better guess as to the true intentions of the user, it may potentially bring up other problems. If a user has a

choice between two buttons, one of which is selected 99% of the time, and the other is selected only 1% of the time, a touch that is largely within the unpopular target but just barely grazes the popular box may cause the system to inadvertently select the popular box. This problem can be especially worrisome if the buttons are very small and clustered together, and the touch area of the finger is larger than the target. In this situation, it may be very difficult to select an unpopular button sandwiched in between two popular buttons. This is obviously undesirable, so measures must be taken to avoid these occurrences.

Determining Optimal Configuration Constants

A few preliminary predictive pointing trials were conducted, where the popularity score and the area of overlap were weighted equally. Unfortunately, the error rates were unusually high. As the previous paragraph indicated, many of the more popular targets were essentially "pulling" selections away from more unpopular ones, if the area cursor overlapped at all. Two strategies were considered in an attempt to alleviate this problem.

The first strategy was to make the size of the area cursor slightly smaller. Originally, the area cursor was a circle 80 pixels in diameter, taken from an estimate of finger size from a current kiosk system[66]. After some experimental trials with a variety of fingers and area cursor sizes, the diameter of a circle was brought down to 60 pixels. This provided a better approximation for the area of the finger surface when in contact with the touchscreen, and translated into a lower and more acceptable error rate. The larger area cursor was causing a much higher incidence of accidental overlap on adjacent targets, even if precise, careful selections over the center of targets were made. Reducing the size of the area cursor helped to curb this possibility.

The second strategy followed was to rebalance the popularity ratings. In the test, the user was to select their favorite M&M color from three choices. This task was designed so that users could nearly instantaneously make a decision as to their preferred color. In addition to this, a priori probabilities were known for color preference, according to an online survey[10], so that predictive pointing could take advan-

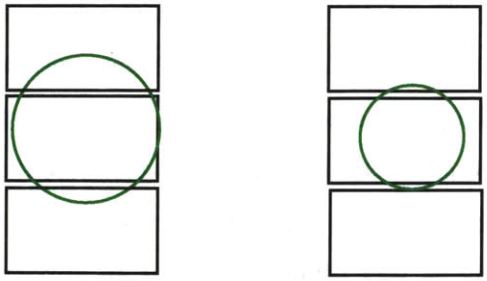

Figure 3-5: Decreasing the Size of the Area Cursor to Reduce Instances of Accidental Overlap

tage of "historical" data. Unfortunately, there were severe differences in M&M color preferences—blue M&M's were approximately six times as popular as yellow M&M's. If a situation arose where blue and yellow choices were adjacent with one another, a touch near the border between the two would invariably lead to the selection of the color blue. These situations were handled by reducing the disparity between the popularity scores by adding in a constant factor to all scores, as well as taking into account the position of the finger more heavily than the popularity score.

After some experimentation, it was determined that instead of simply weighting the overlap area and popularity score equally, the area of overlap would be squared, accounting for finger position more heavily than the popularity. These changes helped to greatly reduce the instances of unintentional selection for predictive pointing. Unfortunately, there is no standard algorithm for dictating the distribution of popularity scores, or the size of the area cursor. For each system, a separate analysis would be necessary to make adjustments for optimal performance.

Learning

Although these constant values of the weights, popularity scores, and finger touch size were determined experimentally, it is useful to think about how the system would benefit from learning from its mistakes. Often, there are a number of physical characteristics that may affect these constants: the size of the screen, the angle of view, the size of the average user's finger, along with several other factors. Even though the increased size of the area cursor helps to provide a better estimate as to finger

position, it also introduces a potentially higher level of error. As indicated in prior sections, a touch mapped to a single pixel, even near the border of adjacent buttons, never considers activating both possible targets. However, with predictive pointing, selections that overlap multiple targets are common. Fortunately, by adjusting the constant values over time according to mistakes, the accuracy of the system can be fine-tuned to produce lower error rates.

This can be accomplished by altering the popularity scores or finger area size. With predictive pointing, it is relatively easy to determine once a mistake was made. If the user makes a target selection, cancels it, and then makes a selection on an adjacent target, this implies that the system incorrectly "guessed" the intentions of the user.

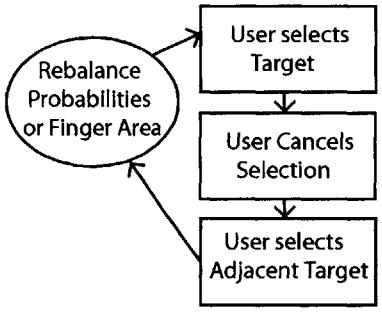

Figure 3-6: How Predictive Pointing Learns from Mistakes

If the system detects this type of interaction, the constants can be altered to reflect this error. If the system is a highly-trafficked ATM, there probabilities of target selection are likely fixed–these popularity scores are very precise. In this situation, it may help to reduce the size of the finger area in an attempt to reduce errors. On the other hand, if the system is a fairly new kiosk with a limited amount of interaction, it may be more beneficial to adjust the probability scores of the targets, since these constants may not have yet reached their static values.

Predictive pointing represents one approach that attempts to improve selection accuracy by incorporating additional, helpful information. The following section proposes another method of user interaction for touchscreens, developed to take advantage of one of the strengths of fingers as input devices, while lessening the effect of

miscalibration on user input. This method, recognizing the user drawings onscreen, attempts to improve selection accuracy as well, by providing an alternative means of selection that is intuitive and resistant from error.

3.3 Finger Stroke Gesture Recognition

When users interacts with kiosks, their means of interaction is rather limited. Past experiments and studies have attempted to use advanced and complicated techniques such as hand motion interpretation or speech recognition to allow the user to interact with the system as naturally as possible. Simple, natural interaction with the kiosk is especially important because many members of the general public have limited experience with these types of interactive systems. Although kiosks attempting to interpret hand motions or speech have been developed to make interaction easier for novices, they can be difficult to master, and may leave new users frustrated.

However, in some instances, there may be expert or highly experienced users of the kiosk system. Cashiers in many fast food restaurant chains or factory workers rely heavily on touchscreens to place orders quickly and meet demand. These touchscreens may be very complicated and the screen may be cluttered with buttons. These operators are using touchscreens for several hours each day, and are completing dozens of transactions each hour. When these types of users interact with the kiosk, they are looking to either obtain specific pieces of information or complete a transaction as quickly and accurately as possible. In these situations, there is no need to guide the user through each transaction. In fact, it may be advantageous to allow the user a set of shortcuts in order to accelerate the process. However, these shortcuts should still be intuitive, and the user should be able to be taught these shortcuts with minimal learning time.

One idea of natural interaction that has largely been overlooked in the past is the notion of gestures drawn on the touchscreen. On many small, portable devices such as PDA's and tablet PC's, gesture recognition of strokes drawn onscreen has become common. Most people have some skill at using their fingers to reliably draw

simple shapes. However, this technique has been tailored more towards handwriting recognition, rather than the execution of actions.

As the number of these new devices lacking mice or keyboards increases, a need to develop new interaction techniques arises as well. Using gestures drawn with either a pen or a mouse to invoke actions has become progressively popular over the past few years. Several browsers (such as Opera[13] and Mozilla Firefox[14]) support the use of mouse gestures as a simple but effective means of navigating through websites, allowing users to browse the Internet efficiently without utilizing the help of the keyboard, menus, or toolbars. This gesture-recognizing support has even been implemented in text editors (such as XEmacs) as well.

In most of these applications, in order to "draw" a gesture, users hold down a designated mouse button, maneuver the mouse in a particular shape to form a unique gesture, and release the button. This shape is then processed and compared against known shapes, and if recognized, the action associated with that shape can be performed. Within these browsers mentioned previously, gestures can perform a number of useful functions, and can be personalized to each individual. In addition, these gestures can easily be extended to recognize and perform a very large number of actions (provided that the user remembers). The success and popularity of such types of gesture recognition systems gives some indication that gesture-based actions may be a viable method of touchscreen interaction as well, since fingers provide a much more natural method of drawing shapes that mice or other input devices.

Benefits of gestures

One prominent advantage of providing support for gestural input is that supporting gestures take up no additional room onscreen. Touchscreens are not usually large, and real estate onscreen is precious. By supplementing existing systems to support gestures, systems can provide another method of robust input without using additional space.

More importantly, gestures are composed of relative movements of the finger on the touchscreen. If the screen causes all selection to be shifted in one direction,

many touches will be registered in correctly. However, if gestures are the mode of interaction, the shape of the symbol is unchanged. Even if the user accidentally rests their free hand on the screen, the relative movements of the finger on the screen can still be interpreted correctly, and thus do not cause problems similar to those of pointing. Although the location of the gesture may be incorrect in these cases, the shape of the gesture nevertheless remains the same. This is a crucial feature of finger gesturing that cleanly and robustly handles the issue of miscalibration.

Developing a finger-based gesture recognition system (FGestures)

Most users are familiar with the process of using their finger to press buttons on kiosks. In spite of its simplicity, this bare-bones approach is highly effective, and very easy for the novice user to pick up. However, despite the simplicity of this convention, it limits the user to the number of actions that are displayable as buttons on the screen. One of our goals was to implement a novel way of executing commands on the kiosk that was flexible and easily scalable. To achieve this, we developed a finger-based gesture recognition system allowed the user to execute commands on the kiosk with a simple wave of the finger.

Mapping Finger movements to direction

Within FGestures, each gesture begins once the user places his finger on the touchscreen. If the finger is removed from the touchscreen without moving across the screen, the touch is considered as a normal mouse click. However, if a user's finger is dragged across the screen, FStroke begins to take notice. FStroke recognizes movement across the screen, and stores each incremental gesture of the user's finger movements as a character appended to the gesture string. Once the user lifts his finger from the screen, the gesture string is interpreted to determine if the gesture was a valid or not. A valid gesture fires off the appropriate action.

Directional Pad Mapping

The primary method of finger stroke recognition comes from mapping each incremental drag of the finger to a direction on a control pad. (A directional control pad can frequently be found on input devices for most current video game consoles.)

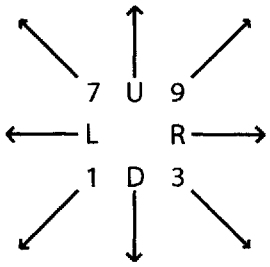

Figure 3-7: A Directional Pad

A fairly simple but effective approach is as follows, modelled after a similar scheme utilized in Mozilla Mouse Gestures[14]. Once the finger is placed on the screen, every subsequent move along the face of the screen is mapped to either a move upwards (U) downwards (D), to the left (L), to the right (R), or diagonally up-right (9), up-left (7), down-right (3), down-left (1), as indicated above in the directional pad depicted in Figure 3-7. Each gesture drawn onscreen is represented by a string of these characters.

Each time the finger is dragged farther than the minimum threshold for movement (which, after some experimentation, worked optimally at 10 pixels) from the past recorded point, it was immediately analyzed to determine the direction of movement. Once the direction is determined, a character representing that direction is appended to the current string. The expression is analyzed to see if fits the format of a defined gesture. The technique used to evaluate the expression created by the gesture is discussed in the following section.

Mapping Gesture Expressions to Configured Gestures

Once the gesture had been completed, the string representation of the gesture was analyzed. Initially, we had planned on interpreting a few number of strokes–we wanted

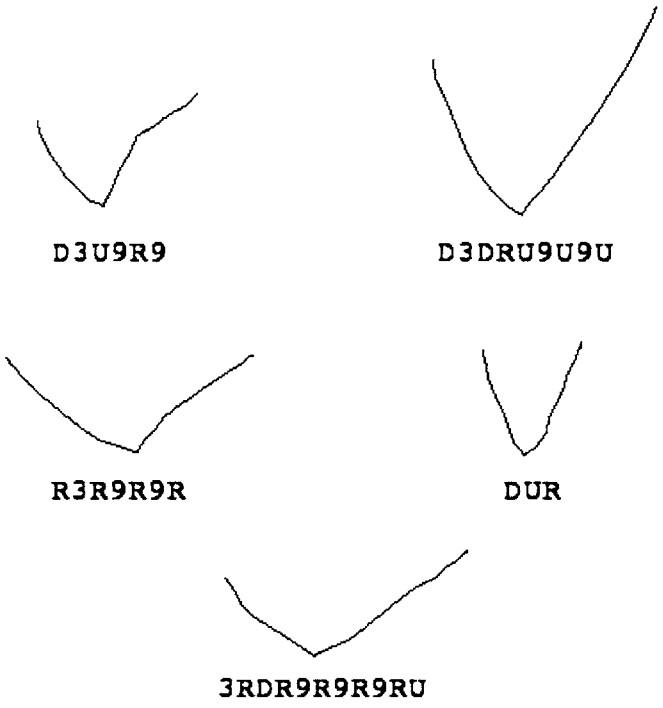

Figure 3-8: Some sample check marks and their interpreted gesture strings

to differentiate between a check mark, a slash mark, and horizontal lines. In a simple world, these gestures could be mapped very simply as these precise strings, respectively: "39", "1", and either "L" or "R". Drawing these gestures precisely with a mouse is very easy to do, since mice can be maneuvered in very precise, controlled movements. Most users can quickly become good at drawing these types of shapes using mice because of the aid of clear visual feedback of the cursor position as the user moves the mouse. Unfortunately, most users are not as skilled at drawing precise shapes with their fingers.

For example, any gesture that represents a stroke which moves down and to the right, and then up and to the right should be interpreted as check mark. However, even a gesture as simple as this one can be drawn seemingly an infinite number of ways: "39", "D9", "DRU", "DRDU", etc. This presented a problem: the more complicated a true gesture was, the larger the number of valid interpreted string

representations there would be. It is evident that matching gestures to a library of possible strings is not a scalable approach, since each additional stroke in a gesture would result in an exponential increase in the number of valid strings.

Identifying Gestures Through Regular Expressions

The solution we came up with was to develop a regular expression to characterize each particular gesture. To continue with the example of the check mark, any gesture that generally moved down and to the right, then up and to the right should be interpreted as a check mark. One possible regular expression that characterizes a check mark could be:

$$[DR3]^{+}[UR9]^{+}$$

Since no two identical strokes can be recorded consecutively (such as RR or DD), this simple expression is powerful enough to recognize a large variety of check marks, without confusion with other types of gestures. By examining the expression more closely, we can decipher that the expression recognized gesture strings that represent at least some movement down and/or right, and then some movement up and/or right. These types of gestures should classify as check marks. In most cases, the regular expression above is robust enough to recognize almost any check mark shape, regardless of position onscreen or sloppiness. Utilizing regular expressions to classify gesture strings is extremely powerful; this relatively simple expression can recognize a range of sloppily drawn check marks from just "39" to "DR3RURU9RU", and all plausible combinations in between.

3.4 Implementation within the SKINNI Kiosk System

The SKINNI Kiosk System is an information kiosk that allows members of the MIT community to share and obtain information regarding daily events, faculty office locations, and directions between rooms. These kiosks are installed in public lounges and lobbies, and utilize touchscreens as the primary means of interaction.

Both Predictive Pointing and Finger Gestures were implemented on these touchscreens to provide a means of richer interaction. These techniques represented an exploration of the effectiveness of having the intermediate layer between the application GUI of SKINNI, and the underlying operating system. Instead of interpreting touches onscreen directly to the associated callback, a calculation regarding the position of the touch and the selection probability of possible targets is performed to estimate the true intended target. This additional layer of computation allows us to develop a more refined interaction system. The traditional method binds the act of selection and the callback execution together. By separating the two, this allows the system to not only provide better approximations of user intent, but opens up the possibility of learning. Similarly, the interpretation of gestures onscreen allows for a more sophisticated means of interaction. We can interpret particular dragging actions to have specialized meanings. On a traditional WIMP system, this possibility does not exist.

We can develop an engine that contains information regarding various aspects of user behavior, including selection probabilities, information on screen miscalibration, and finger size constants. By isolating the storage and calculation of these metrics into a separate engine, each application can avoid having to implement these interaction techniques themselves.

The SKINNI Kiosk System is not the ideal platform upon which to implement this system. This separation between the GUI event and the resulting callback should be implemented on all touchscreen-based interfaces. The touchscreen is particularly prone to errors from imprecision and miscalibration, and this concept of separation

represents one approach that alleviates these types of interaction problems.

This chapter has outlined the two methods designed to improve both the accuracy of touchscreen interaction and the handling of touchscreen miscalibration. Predictive pointing aims to improve target selection accuracy by taking finger surface position and past interaction probabilities into account. Finger gestures attempt to provide a more robust yet intuitive interface for touchscreen users. The following chapter attempts to concretely analyze the efficacy of these techniques in a defined user study.

Chapter 4

User Study

This chapter presents a user study designed to evaluate the effectiveness of the aforementioned techniques in enhancing touchscreen performance in basic selection tasks. These techniques were compared against conventional methods of user interaction to determine whether any significant improvements in task completion times could be claimed.

4.1 Participants

Thirteen individuals, ten male and three female, participated in the user study. They ranged in age from 19 to 24 (mean age = 22.3 years), and were undergraduate and graduate students recruited from the Massachusetts Institute of Technology. All participants were frequent users of computers, and all had had sizeable experiences with operating touchscreens. Subjects were asked if they would participate in the experiment, but were not compensated for their participation. (Permission to conduct this study was granted from the MIT Committee On the Use of Humans as Experimental Subjects.)

4.2 Apparatus

The experiment was run on a MicroTouch 3M M170 Flat Panel Display Touch Monitor, on a machine running an unstable version (2.6.5) of Debian Linux. The monitor was a 17" screen using active-matrix thin-film transistor liquid crystal display technology, and ran at an optimal full-screen resolution of 1280 x 1024 pixels. Participants were asked to use their dominant hand when interacting with the touchscreen, and experiments were done in full-screen mode, with a neutral gray background color.

4.3 Design and Procedure

The subjects were asked to perform five different tasks, which included four trials of various pointing tasks, as well as one trial of finger gesturing as input. The four pointing tasks were as follows:

- Normal Pointing (NP): Users were presented three buttons, each of distinct colors, and were asked to select their favorite M&M color of the three by touching that button. Color selection was chosen so that users could nearly instantaneously make a decision as to their preferred color. Participants rotated between three screens (See Figure 4-1): a start screen (with one button at the center of the screen they selected in order to begin the current trial), a choice screen (with the three buttons of varying colors where users were asked to make their selection), and a verification screen (where users were asked to indicate whether the system's registered selection was correct). There were three sets of 20 trials conducted for this task. Each set of trials used a different button height, varying between small (50 pixels), medium (75 pixels) and large (125 pixels).

- Predictive Pointing (PP): This task was identical (from the user's perspective) to the NP task above, but utilized the principle of predictive pointing in attempt to better estimate the user's selections.

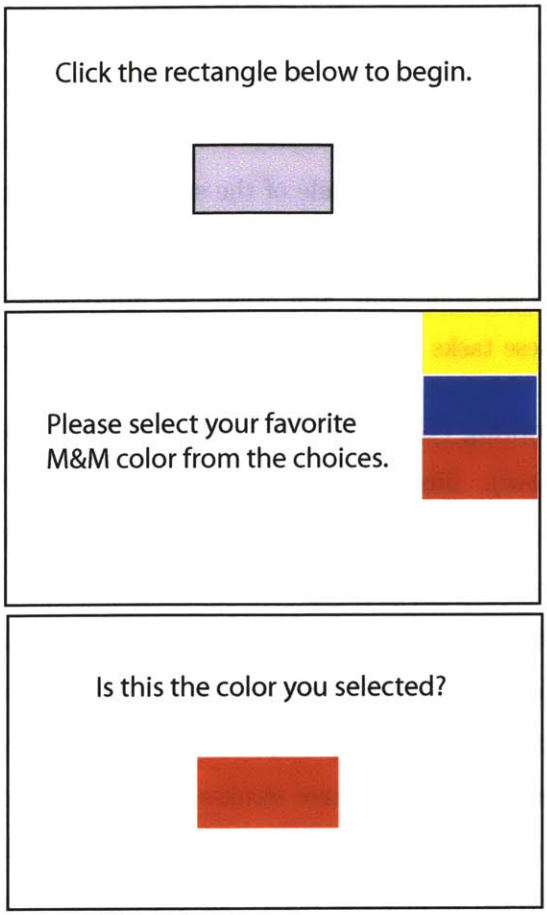

Figure 4-1: The Start, Choice, and Verification Screens of the Normal Pointing Task

- Simple Pointing (SP): This task was designed to provide an estimate for the experimental constants from Fitts' Law, specifically for the touchscreen used. Participants alternated between two screens: the start screen and the task screen, where they were shown a single button of size W and of distance D. These two values were varied (W = 35, 60, 100, 250 pixels and D = 464, 597, 764 pixels) and picked randomly throughout the course of the trials.

- Pointing with Distractors (PD): This task was designed to test the difficulty of selecting a particular button amongst a varying number of distractor buttons. This task was tested primarily to develop a model for a more practically realistic situation that could be compared with the Gestural Drawing test (see below),

and determine the effect of distracting targets to target acquisition time. Participants alternated between a start screen and a target acquisition screen. The target acquisition screen presented a textual command to select a particular button. On the right hand side of the screen, the button was displayed with a random number of distracting buttons, ranging from 1-8 distractors. The user was to read the command, and select the appropriate button. The selection times for these tasks were measured, along with the error rate.

- Gestural Drawing (**GD**): This task was designed to be equivalent to the PD task (as above). Similarly, users alternated between the start screen, and a gesture screen. Once the user clicked the start button, they were asked to draw a random gesture, which they were familiarized with before. These gestures ranged from 1 stroke (horizontal line, vertical line, diagonal slash), 2 strokes (check mark, carrot symbol, greater-than symbol), 3 strokes (triangle, the letter C, the letter Z), 4 strokes (square, the letter W, the letter S), and 5 strokes (the letter E, a square wave). These strokes were chosen primarily because of their familiarity, and users were shown several ways how each shape could be drawn, and were allowed to practice before the experiment began. The times for each gesture was recorded, as well as the error rates.

Before the gestural drawing trials, the users were familiarized with the concept of gestures, and were given a untimed practice session to go over the mechanics of performing gestures. The subjects were told to try out unfamiliar shapes so they could be somewhat accustomed to using them during the trials. Most of the participants were familiar with the majority of the shapes.

In all of the trials, participants were asked to perform each task as quickly and as accurately as possible. Users were instructed to continue retrying the task if nothing was registered by the system. Each time a target was missed, or a gesture went unrecognized, the user was provided visual feedback to prompt them to retry the task. If the target was hit, or the gesture was recognized, the screen reverted back to either the start screen or the verification screen, depending on the task.

4.4 Results and Discussion

4.4.1 Predictive Pointing (PP) vs. Normal Pointing (NP) Tasks

The experimental sessions revealed that there were significant differences between the performance of PP and NP tasks. Throughout the analysis of these results, we utilized standard statistical metrics to determine the level of significance: t-values which represent the ratio between the difference of the sample means and the estimated standard errors, and p-values which provide an objective measure of the strength of evidence which the data supplies in favor of the null hypothesis. The lower the p-value, the greater the confidence level of declaring a significant difference between the two samples, not a difference due to random chance. The calculation of the t-value (and thus the p-value) takes into account the mean, variance, and sample size of the distributions.

There were significant differences when comparing the error rates between NP and PP tasks of selecting small buttons ($t = 6.567$, $p < 0.001$) and medium-sized buttons ($t = 4.963$, $p < 0.001$). NP tasks averaged an error rate of 27.7% (5.53 errors per 20 trials) in selection of small buttons, while PP produced only an average of 8.1% (1.615 errors per 20 trials). For medium-sized buttons, the PP trial error rates averaged 5.75%, while NP trial error rates averaged 19.6%. For largest buttons tested, subjects did not make any selection mistakes.

Mean trial completion times were evaluated for predictive pointing and normal pointing tasks as well. Variance analysis indicated that the task completion times were significantly different between these two tasks for small buttons ($t = 2.95$, $p < 0.01$). On average, subjects navigated the predictive pointing trials 10.4% more quickly than the normal pointing trials, and all but one of the participants navigated through the PP trials more quickly than the NP trials (see Figure 4-2). For medium-sized buttons, subjects navigated through the PP task more quickly as well ($t = 2.744$, $p < 0.02$), on average approximately 2.8 seconds faster than the control NP task.

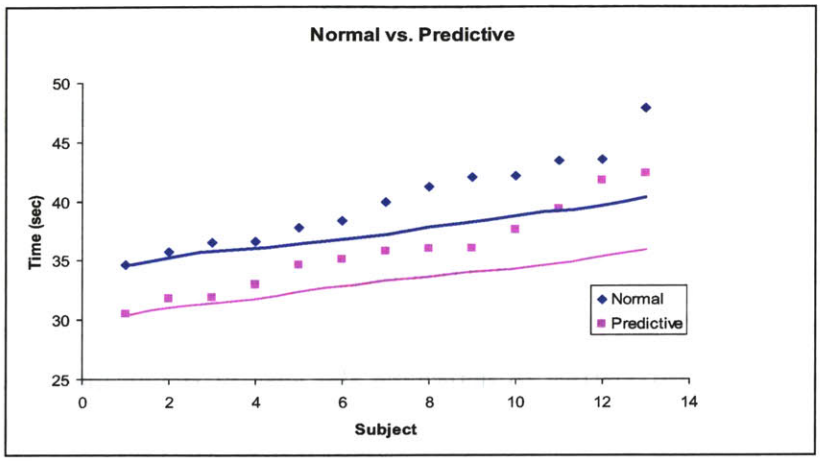

Figure 4-2: A Comparison of Completion Times for Normal and Predictive Pointing tasks

For the largest button size, performance between the two tasks was not statistically different ($t = 0.800$, $p < 0.5$).

In addition to the error rates and completion times of these trials, the frequency of missed clicks was also recorded. (Missed clicks are clicks that are registered by the system, but are not located over any selectable target.) PP also provided a significant reduction in the number of missed clicks (mc) from the NP trials, for small buttons (16.8 mc vs. 61.2 mc, $t = 21.685$, $p < 0.0001$), medium buttons (17 mc vs. 59.77 mc, $t = 19.203$, $p < 0.0001$), and large buttons (17.6 mc vs. 60.69 mc, $t = 20.079$, $p < 0.0001$). In all cases, there were nearly a 250% increase in missed clicks in the NP trials when compared to the PP trials. PP tasks averaged 17.3 missed clicks (per 20 trials), while NP tasks averaged a sizeable 61.2 missed clicks (per 20 trials).

As expected, larger size buttons resulted in reduced error rates, missed clicks, and task completion times. Users felt that the larger buttons felt much "more responsive" than smaller buttons.

In real-world situations, machines may be either poorly calibrated or miscalibrated, causing input from the user to be displaced somewhat. In these cases, predictive pointing would also provide an advantage over normal pointing techniques as well.

Table 4.1: Comparison of the Results of Normal and Predictive Pointing Tasks

Technique	Normal Pointing			Predictive Pointing		
Button Height (pixels)	50	75	125	50	75	125
Error Rate (%)	27.65	19.62	0.00	8.08	5.75	0.00
Time (s)	39.99	37.80	35.98	37.21	36.06	35.33
Missed Clicks	61.15	59.77	60.69	16.77	17.00	17.62

4.4.2 Normal Pointing with Distractors

Movement times for simple pointing tasks followed Fitts' law very closely, correlating with the target-distance to target-height ratio. After a linear regression on the experimental data, the following equation was determined for simple pointing tasks:

$$MovementTime = 374 + 230 \cdot log_2\{Distance/Width\}$$

$$r^2 = 0.9876$$

While movement times of single button acquisition did correlate well with Fitts' Law, they were not directly useful for comparison with other methods, since in nearly all cases of user interaction, users are not asked to select a single button onscreen, but rather to select a single button out of numerous other possible buttons. Thus, it was more appropriate to compare the proficiency of gestural input as compared with real situations, where distracting targets would be present.

For these trials, users were asked to select a single target button from a panel with a number of distractor buttons. For a set button size, results indicated that as the number of distracting buttons was increased, there was an increase in target acquisition time of the form:

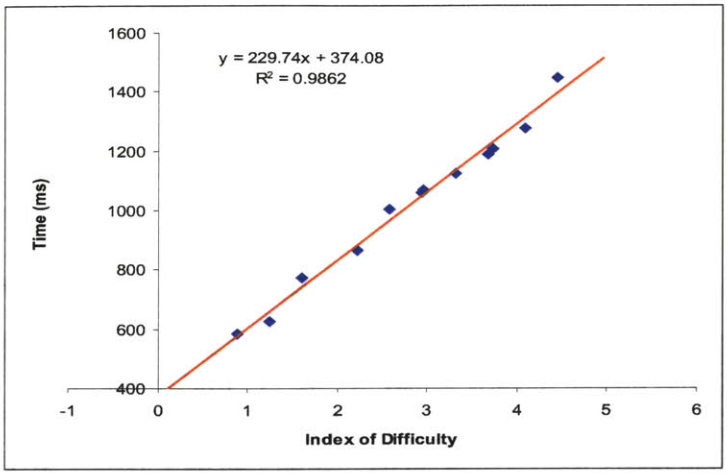

Figure 4-3: Fitts' Law Regression Plot

$$AcquisitionTime = c \cdot log_2\{N + 1\} + d$$

where T_0 is the time for target acquisition with no distracting targets, N is the number of distracting targets, and c and d are experimental constants, where 0 ¡ c ¡ 1. The following constant values were determined from experimental data:

$$AcquisitionTime = 523.77 \cdot log_2\{N + 1\} + 536.11$$

Figure 4-4 displays the relation between distracting buttons and target acquisition times.

Fortunately, the addition of distractor targets appears to be easily explained, and a new expression for an extended (and more immediately useful) variant of Fitts' Law with distractors was determined:

$$M = a + b \cdot log_2\{A/W\} + c \cdot log_2\{N + 1\}$$

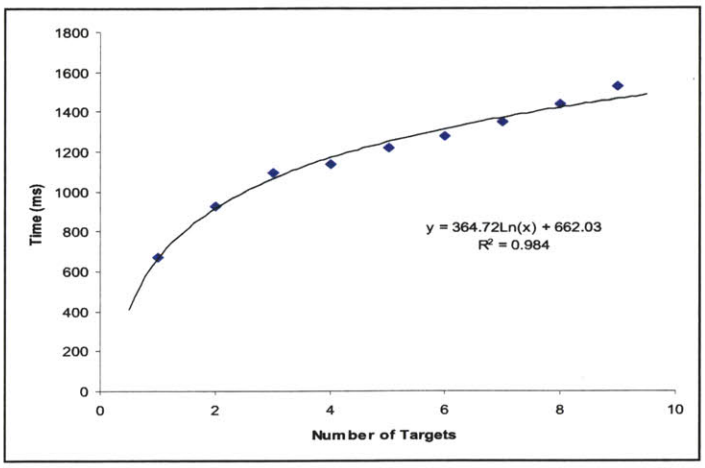

Figure 4-4: The Relation Between Distractor Buttons and Target Acquisition Time

4.4.3 Gestural Input

For the Gestural Drawing tasks, users were initially slower to react during the practice sessions, most likely due to the unfamiliarity of the type of task. However, most participants quickly improved their GD completion times during the actual test.

For gestures, there was no concrete measure of an equivalent "index of difficulty", as there is for traditional pointing tasks. However, it is apparent that as the complexity of the gesture is increased (increasing the number of strokes of the gesture), the longer the completion time of the task should be. Indeed, results supported this assumption:

These results indicated that each additional stroke added a constant offset to the task completion time, implying a linear relationship between the number of strokes within a gesture and movement time to complete that gesture. The correlation of the data with the regression line was very high ($r^2 = 0.9842$) The shapes that were used in this experiment were very common shapes, and most users were very familiar with them. However, if the users were tested on a set of more unfamiliar shapes, this most likely would have a greater effect on task completion time, and could even result in a polynomial relationship between stroke number and completion times.

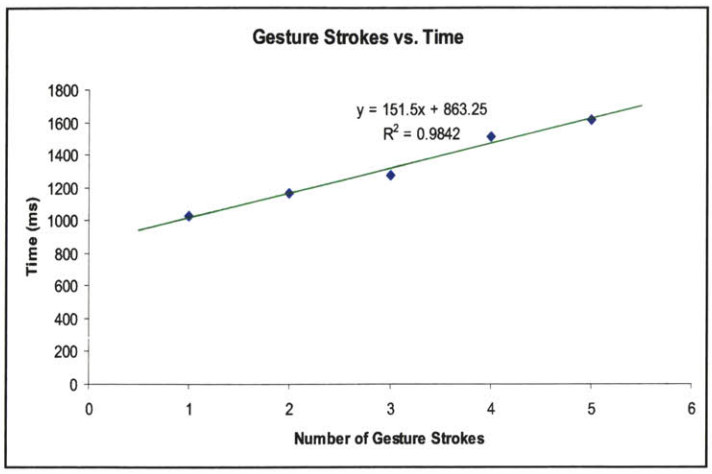

Figure 4-5: The Effect of Number of Strokes in Gesture on Task Completion Time

From analysis of the aforementioned experimental data, the movement time could be described with the relation:

$$MovementTime = a + b \cdot N$$

There appeared to be a lawful regularity between movement time and the number of strokes within the gesture. From the linear regression of the data, the following equation was determined:

$$M = 863.25 + 151 \cdot N$$

where N represents the number of strokes. At this point, it was not immediately clear whether gestures were slower or faster than normal pointing tasks with distractors, since there was no commensurate scale between the two to compare relative indices of difficulty.

4.4.4 Comparison of Normal Pointing with Distractors and Gestural Drawing

Although gestural drawing and normal pointing tasks can both execute actions similarly, as mentioned above, there is no relative scale between the two that allows for direct comparison. Currently, it is not known what commensurate Fitts' Index of Difficulty exists for gestures of any number of strokes. Although an attempt was made to compare the two interaction types, developing a concrete equivalence relation between the two is beyond the scope of this thesis.

However, it is possible to draw certain assumptions as to how gestural drawings can translate to a corresponding instance of the Fitts' Law with distractors. With the implementation of gesture recognition in this thesis, let us assume that a one-stroke gesture may be recognized as one of eight unique symbols. If stroke direction is ignored, then this leaves four distinct recognizable symbols: a horizontal line, a vertical line, a forward slash (/), and a backward slash (\). A user could execute one of four actions with a single-stroke gesture, making it effectively equivalent to selecting a single button out of four. According to experimental results, one-stroke gestures averaged 1028.17 ms completion time, and an error rate of 8.25%, while selecting a button with three distractors averaged a 1135.28 ms completion rate and an error rate of 7.5%. The difference in times were statistically significant ($t = 2.7549$, $p < 0.01$), although the error rates were not shown to be significantly different ($t = 0.1667$, $p > 0.5$).

There are theoretically 56 distinct two-stroke gestures (excluding two consecutive strokes in the same direction) possible with the implementation used in this thesis. However, due to the imprecision of the finger, many of the shapes not depicted in cannot be used, since they are difficult to distinguish from single-stroke lines and the two-stroke shapes which are shown above (See Figure 4-6).

Let us assume that these ambiguities due to user imprecision result in the eight L-shaped gestures depicted in Figure 4-7 as possible two-stroke gestures that are distinct enough to be recognizable. With the same reasoning as used above, a two-stroke

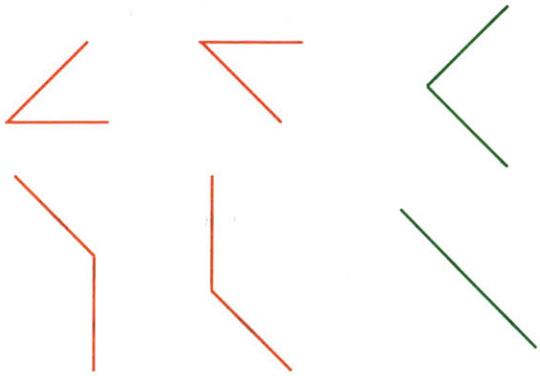

Figure 4-6: For sloppy users, it may be difficult to distinguish the two gestures on the left from the one on the right

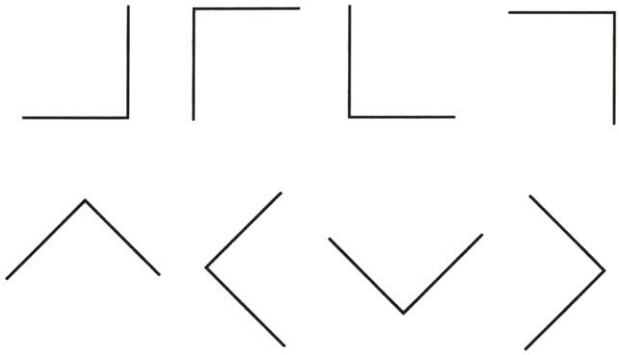

Figure 4-7: Eight Distinct Two-Stroke Gestures

gesture can be thought of as having equivalent "power" as an eight-button panel. When a two-stroke task is compared with a selection task from a panel with eight buttons, there was a significant difference ($t = 3.9874$, $p < 0.001$) in completion time. The average time for a two-stroke gesture averages 1162.95 ms, while the average time for an eight-button search is 1436.55 ms. However, despite this reduction in completion time, gestural drawing did not produce significantly more errors (7.76% for gestures, 7.5% for buttons with distractors). If the reader accepts the supposition that each additional stroke provides an additional four possible shapes, Figure 4-8 shows the relation that compares the performance of the two.

From the results gathered, it appears that under this interpretation, gestures outperform pointing tasks with distractors. For comparison, if the reader believes that each additional stroke is the equivalent of two additional buttons, the resulting

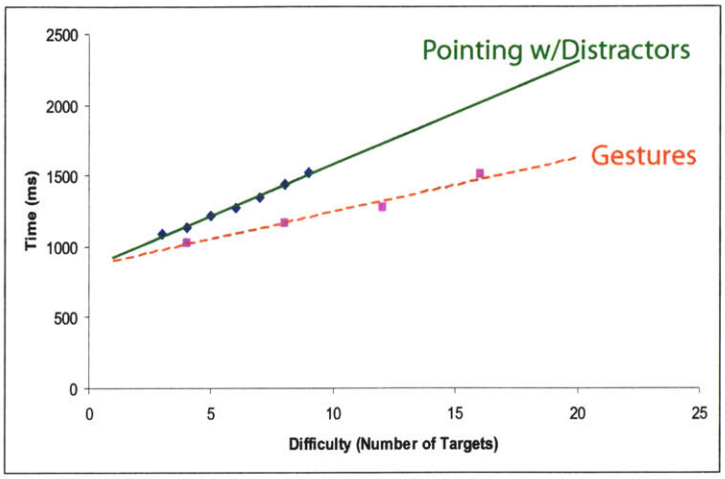

Figure 4-8: A comparison of the Gestures and Pointing with Distractors (4-button stroke equivalence assumption)

relation is shown in Figure 4-9.

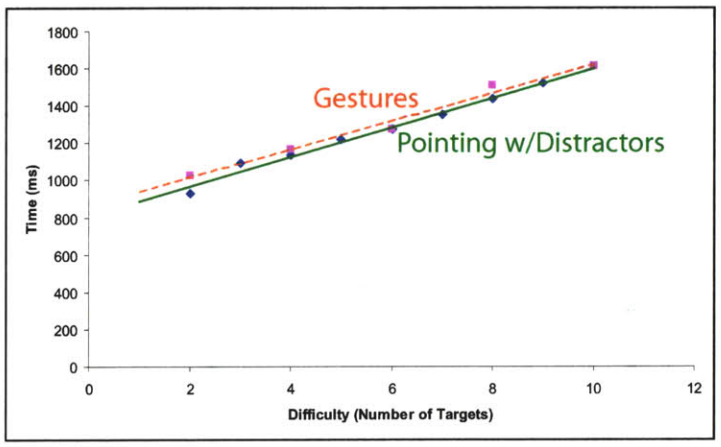

Figure 4-9: A comparison of the Gestures and Pointing with Distractors (2-button stroke equivalence assumption)

Although data was recorded for three-stroke, four-stroke, and five-stroke gestures, many users remarked that the more complicated gestures seemed "slower", and that they would generally prefer buttons over these longer gestures. Additionally, even though more shapes are possible with more strokes, users felt that it would be very difficult to remember the shapes unless they were familiar with them. For simple

gestures composed of a few strokes, gestures may be effective, according to the results of the study. However, it is assumed that for the general case, users are far more likely to use buttons to make their selections rather than drawing complicated, multi-stroke gestures. Most participants indicated the ease of gestures was due to the fact that the shapes were familiar, and very little processing time was necessary to recall the shape. Once this familiarity was lost, the gestures would be difficult to remember, since they would be an arbitrary set of strokes to the user.

4.4.5 Learning Effect

Several of the subjects were randomly selected to participate in an additional two trial runs for both gestural input and the normal pointing tasks with distractors, to determine whether any decrease in completion times could be suggested. Participants significantly improved their performance times over the additional three trial blocks in both cases, although their error rate did not differ significantly between trials. These additional trials were not used in the comparison of the GD and PD trials.

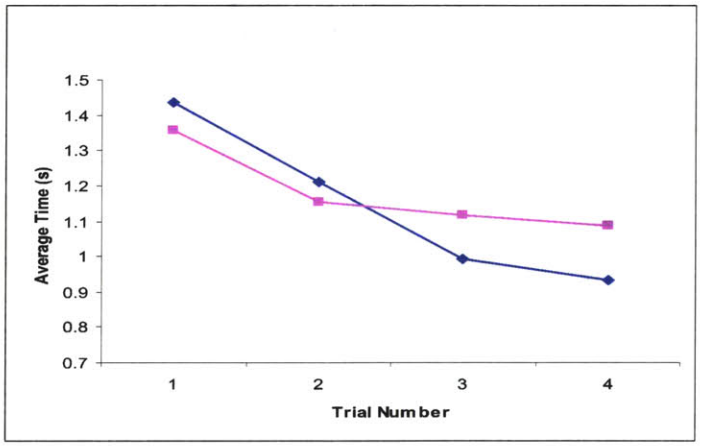

Figure 4-10: A comparison of Completion Time Performance After Additional Trials

As seen above in Figure 4-10, users greatly improved their PD and GD times with practice. Improvement of GD completion times over PD completion times appeared to be sizable. Although subjects were significantly slower on initial trials of drawing gestures, they quickly picked up the method of interaction.

Participants in the additional trials noted that the improvement with buttons was due largely to the memorization of the button layout. However, subjects also indicated that the improvement of gesture operation time was primarily due to an increased familiarity with the interaction technique. All of the participants felt very comfortable with using gestures as an interaction technique upon completing these additional trials. Several of the participants said that they preferred it over searching for buttons, since it felt "more natural". On the other hand, some of the other participants felt more comfortable with buttons, as they had had significantly more experience with them.

The following chapter discusses the implication of these results in GUI design, and explores possible reasons for performance differences.

Chapter 5

Implications and General Discussion

As expected, analysis of error rates and missed clicks showed significant benefit of using a predictive pointing model over the normal pointing model, particularly for smaller buttons. The experiment tested the two techniques on small buttons sizes to force ambiguous input, where the subjects' fingers were always bordering multiple buttons. The results indicate that for borderline cases, where input may be ambiguous, predictive pointing proved to be much less error-prone than normal pointing.

However, it was also interesting to not the significant reduction in missed clicks as well. The experiment did not contain any other active areas aside from the buttons involved, and thus missed clicks did not have any negative effects, but it is important to point out that on real touchscreens, other areas may also be activated, so missed clicks may have more undesirable effects.

Another side effect to note from the results of predictive pointing task is the fact that there was a significant decrease in the amount of time required to complete the trial when compared to the normal pointing task. Although a decrease in error rates and missed clicks was expected, a decrease in completion time was not. This result may be explained by the fact that users expect the kiosk to react correctly to their input, and when the system registers an unintended selections, users are confused for a moment, thus taking longer to continue the transaction. In the experiment, users

were not penalized for incorrect touches, but in real situations, users would be forced to go back and correct the unintended action. Not only would this require additional time on the part of the user, on some occasions, this possibility may not exist. This point serves to emphasize the importance of making the correct selection the first time.

Quantitatively, according to the results of the experiment, predictive pointing is not only more accurate than normal pointing methods on small and medium-sized buttons, but proves to be superior in terms of task completion times as well. It seems that in most cases, implementing predictive pointing is preferable to only supporting normal pointing.

The main benefit of predictive pointing lies within the fact that it does not require any additional time or energy on the user's part. From the user's perspective, a system that utilizes predictive pointing is indistinguishable from one that does not. In both cases, the user instinctually interacts with the touchscreen. However, predictive pointing, if configured correctly, can produce a higher accuracy of interaction. This advantage, as discussed above, often leads to a decrease in transaction time as well, an additional bonus. The predictive pointing method enhances the traditional *Land-On* technique, since the user's interaction remains as natural as possible.

The time to target acquisition with distractors, as expected, increases with the number of distractors. Although the relation between acquisition time and the number of button appears to be logarithmic (as shown in Chapter 4 in Figure 4-4), logically, it seems that acquisition time should increase linearly as the number of buttons increases. This is because each button must be checked (assuming constant time) in turn to see if it is the correct one. Measuring this effect made it easier to compare practical selection tasks with gestures more clearly, although a fully concrete conversion could not be determined.

From the data gathered, it did appear that gestures could be of practical use. If the reader believes the assumptions stated above (that one-stroke gestures are equivalent in power to a four-button panel, and two-stroke gestures are equivalent in power to an eight-button panel), this allows for a direct comparison between gestures

and buttons. Based on the similar error rates but a significantly faster action time for gestures, the results indicate that allowing users to execute actions through the use of gestures drawn on screen may be feasible to the end-user.

Gestural input also affords a richer set of inputs. Although several participants in the study indicated that complicated, unfamiliar inputs would not be beneficial (and would probably be detrimental) to task completion time, they also mentioned that self-configurable gestures they were familiar with and defined themselves would be advantageous. Another characteristic of gestures is that several known commands could be executed swiftly in succession, without significant movement of the user's finger.

Figure 5-1: Gestures can be performed over Icons for Compounded Actions. For example, a slash drawn over the speaker icon could instruct the system to increase the volume, while another gesture could mute the system. This convention could allow for a richer set of possible interactions.

An additional characteristic of gestural-based input is the ability for the user to perform gestures over items or icons located onscreen, for a richer action. This study has explored the use of gestures in general compared with pointing with distractors. However, a gesture can be performed at particular locations, or over specific icons onscreen to induce a compound action. In a traditional interface, the user would have to select the item, and select an action to be performed on the item. With gestures, the two actions can be combined by drawing the gesture over the desired item to achieve the same effect.

One principal reason for the improvement in completion times when using gestures comes from the fact that it eliminates the element of searching. When users wish to

select a button, they must look through each one until the desired button is found, and then select it. With gestures, this search is eliminated to some extent. Practically, certain gestures can be universally tied to common actions: a check mark represents an "OK" button selection, while a slash could represent "Cancel". In these cases, all kiosks can choose to represent this reasonable shortcut. If any user wished to affirm a selection, they can simply draw a check mark on the screen. With conventional systems, the act of searching is always necessary, since not all kiosk button layouts will be identical.

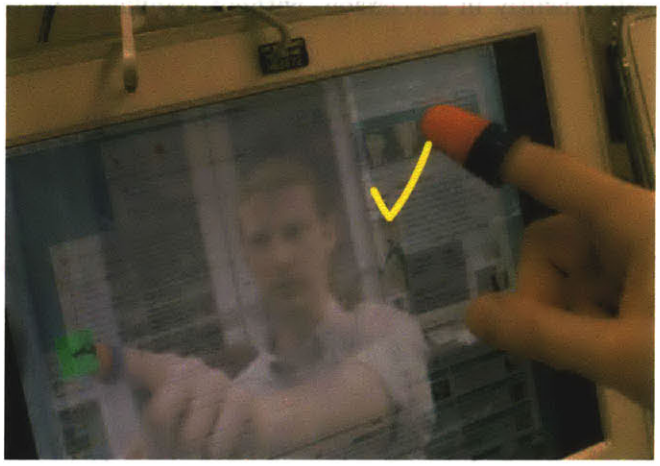

Figure 5-2: A user interacting with a computer through finger gestures.[12]

Not only do gestures significantly improve performance times, they also take up very little screen real estate. Gestures can be supported by the system invisibly, since no button icons must be displayed to the user. This is a crucial advantage over conventional systems, since many touchscreens are cluttered with information. Even if gestures are not adopted as the primary method of interaction on touchscreens, they would undoubtedly be beneficial to users as a supplementary method of input. In fact, in most applications that utilize them, gestures are often a "shortcut" method of input, and can be used as such on touchscreens as well.

Gestures also proved to be much less error-prone than selection in general. The results of the normal pointing study showed that errors were very common, especially with small targets. The touchscreen did appear to be slightly miscalibrated, off by

a number of pixels downwards and to the left. Although predictive pointing proved to be effective at minimizing this problem, gestures helped to soften the effect of miscalibration as well.

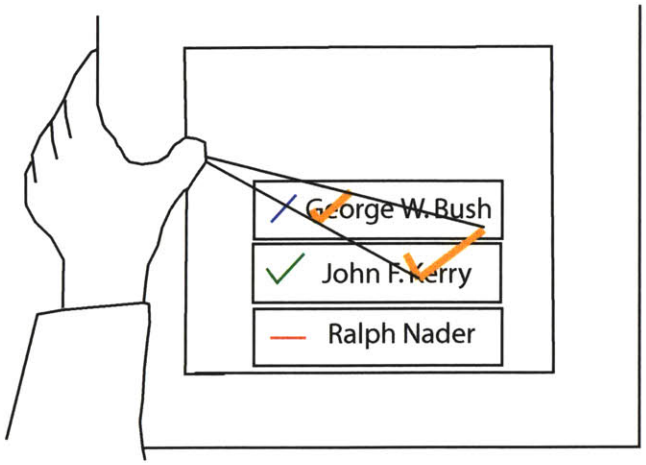

Figure 5-3: Gestures on Accidentally Miscalibrated Touchscreens. Although the user has accidentally rested their thumb on the display, the effective intended shape remains the same.

As Figure 5-3 indicates above, even a miscalibrated screen will recognize the intended symbol, and indication of the robustness of this interaction method. Although the location of the check mark is not correct, the shape remains intact.

As described in previous chapters, a check mark was characterized as a general V-shaped drawing. This means that any two strokes that resemble a check mark will be recognized as such, regardless of the angle of the check, or the length of the strokes. Additionally, the characterizations often allow for some unsteadiness of the finger, so that slight "hiccups" along the stroke path are ignored, and the underlying shape is still recognized.

From the results gathered, and implications discussed, both techniques seemed to offer some improvement over known methods. Even though both techniques were only preliminary steps into an exploration of possibilities for enhancing touchscreen interaction, the results from the user study indicated that with further research, these methods could be improved upon greatly. There are a variety of areas that are left to be explored, and the following chapter outlines these areas, and concludes this thesis.

Chapter 6

Conclusion

This thesis has introduced and investigated alternative interaction techniques designed to better suit the use of users' fingers as input devices on touchscreen kiosks. As evidenced by the results of the user study, both techniques achieved considerable success when compared with conventional methods of pointing. The study of the efficacy of these novel methods of interaction has provided substantial insight towards understanding the capabilities and drawbacks of touchscreens.

The two techniques explored in this thesis were predictive pointing as a means of improving selection accuracy, and gestural drawing onscreen as an alternative means of input, which is largely unaffected by miscalibration. Predictive pointing was designed specifically to be transparent from the viewpoint of the user, while gesture support was developed so that no additional screen space was required. Although this thesis has delved into two original methods of interaction, there are a number of areas left to research. The following sections discuss problems encountered, and suggestions for future experimentation.

6.1 Shortcomings and Problems During the Course of Work

There were a few shortcomings of these methods that were uncovered during the course of work on this thesis. Although both of the techniques intended to improve interaction, there were some problems that were encountered and concerns raised during the evaluation process.

As mentioned in Chapter 3, the predictive pointing technique occasionally made it difficult to select very small, unpopular targets that were surrounded by popular objects. Due to the expanded active region involved with predictive pointing, the selection area would often overlap the nearby popular items. If there were large discrepancies between the popularity scores as well, selections would often be "pulled" towards these popular items, rather than remaining correctly on the unpopular ones.

Fortunately, this problem was easily solved by rebalancing the weight between the importance of finger position and the popularity score. Squaring the "weight" of the finger overlap area helped to eliminate this problem. This implies that finger position is more important that popularity, and helped to decrease the possibility of selection error.

While gestures can be useful in supplementing or replacing buttons, there are a few shortcomings to note. First, inexperienced users will not be able to take advantage of gestures immediately. Oftentimes, kiosk users are first-time and one-time users of the touchscreen display, and will not be accustomed to using these gestures. Indeed, during the course of the user study, many of the users felt less comfortable with gestures than with traditional target selection during the practice round (although several participants quickly became accustomed to gestures.

Thus, although gestures do not negatively affect these novice users in any way (they may choose traditional target selection), they also do not provide any additional assistance. Second, it may be difficult for users (even experienced users) to remember a large set of gestures. While it is possible for users to define their own set of gestures, it will likely be difficult for them to remember all but the most commonly used ones.

Thirdly, it can be difficult to create a robust regular expression to characterize a particular gesture. The more strokes needed to draw a gesture, the more complicated the regular expression will be. When attempting to capture a gesture within a regular expression, it may be hard to develop an expression that does not overlap with other expressions and succeeds in recognizing the gesture consistently.

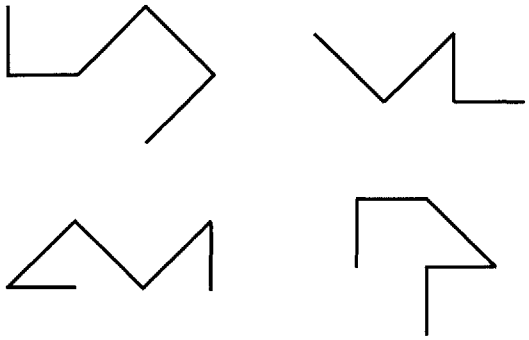

Figure 6-1: Example of Unfamiliar Shapes

An additional downfall lies within the fact that very complicated gestures are unlikely to be useful. While there was evidence that familiar, lesser-stroke gestures provided a performance improvement, many of the participants in the user study noted that arbitrary, unfamiliar gestures (see Figure 6-1) would likely take significantly longer to input. This implies that gestures with more than three strokes would be impractical as input types, since they not only take longer to input, but also come with higher error rates (inherent in more complicated strokes).

6.2 Future Work

This thesis has introduced new methods of interaction on touchscreens. While the techniques explored represented novel research into touchscreen interaction, there remain several other avenues left to explore.

Within the realm of predictive pointing, there are several possible topics left to analyze. During preliminary experimentation with predictive pointing, we decided to implement a circular area cursor. Although a rectangular area cursor was consid-

ered, we believed that the circle represented a better approximation of the fingertip surface area. It may be useful to investigate performance differences between circular and rectangular area cursors. Additionally, the diameter of the area cursor was experimentally determined, and an investigation of an optimal cursor size may be beneficial.

Another area of interest lies with the weighting of popularity scores and area cursor position. It would be curious to examine the error rates after manipulating the weights of popularity and cursor position, to determine an optimal setting, and whether this setting had any relationship to target size and the difference in magnitudes of the popularity scores.

In addition to this, predictive pointing could be enhanced if it were combined with dynamically-updating popularity scores. For our experiments, the weights were taken from the results of a prior survey, and were not updated during the course of the study. In order to make a better predictive system, it may be beneficial to look into the effects of learning. A more elaborate predictive pointing system could take into account mistakes it had made in prediction, and "learn" from them, by adjusting the scores, or manipulating the size of the area cursor to provide more accurate predictions.

As for gestural input, one important area relates to investigating how the complexity of a gesture (the number of strokes) compares directly to an equivalent situation of target selection with distractors. Although a rough approximation of this equivalence relation was proposed, further research into this method of input must be done to establish an explicit relation.

Another area involves looking into the number of gestures that one could feasibly remember. According to some reports[49], the average person can remember approximately seven (plus or minus two) arbitrary items in their short term memory. This mental limitation may prove to restrict the effectiveness of gestures. Even though the task completion times did indicate that certain gestures were more quickly performed than certain target selection tasks, further research must be done to examine whether most people would remember the different gestures and their associated actions dur-

ing the course of a real transaction. In these scenarios, with each gesture mapping to a distinct action, would most people be able to recall these mappings? How quickly would the average person be able to "learn" gestures as a method of input? Could gestures become an effective input method for casual users of touchscreens? There are a large number of these questions that need to be answered in order to better understand the possible role of gestures on touchscreens.

These techniques represent a step towards the separation of GUI events and registered callbacks, and the inclusion of a separate level of computation that aims to improve selection interaction. Our research has shown that there are benefits to breaking this direct mapping, particularly within the realm of touchscreen interaction, where imprecision and miscalibration are serious issues. While predictive pointing and gestural drawing have both proved to be feasible interaction strategies that take advantage of this separation, there are many other strategies that may benefit from this paradigm as well. It would be beneficial to examine this concept with traditional mouse-based interfaces as well, to determine if these ideas still hold with more precise pointing devices.

The work in this thesis represents an examination of methods that attempt to reduce touchscreen limitations, by responding to problems of finger inaccuracy and touchscreen miscalibration. While the results of the user study provide evidence that these methods may be viable alternatives and enhancements to kiosk interaction, further work must be done to give a more concrete evaluation of these techniques.

References

[1] Biometric exit procedure expanded, says homeland security. http://www.how2immigrate.net/usanews/exit-procedure.html.

[2] Diebold touch-screen voting terminals perform well in primary elections. http://www.kiosks.org/articles/pr100102-diebold.html.

[3] Health kiosk. http://www.intellvisions.com/pdf/health-tsk.pdf.

[4] History of the first public access kiosk. http://www.kiomag.com/historyjf04.

[5] Introduction to windows forms - replacing, cloning and merging. http://www.developerfusion.com/show/1770/9/.

[6] Kiosk market growth booms - study. http://www.kiosks.org/newsbits/1997/nb06141.html.

[7] Kiosks. http://www.kioskmarketplace.com/research_story.htm?article_id=12222&pavilion=36.

[8] Mcdonald's to expand dvd rentals. http://money.cnn.com/2004/05/24/news/fortune500/mcdonalds/.

[9] Microsoft powertoys for windows xp. www.microsoft.com/windowsxp/downloads/powertoys/xppowertoys.mspx.

[10] M&m color survey. www.candywarehouse.com/color.html.

[11] Quick-serve restaurants likely to install kiosks. http://www.gokis.net/self-service/archives/cat_qsrs.html.

[12] Transparent desktop opens doors. http://wired-vig.wired.com/news/images/0,2334,64129-13325,00.html.

[13] Mouse gestures in opera. http://www.opera.com/features/mouse/, 2005.

[14] Mozdev.org optimoz gestures. http://optimoz.mozdev.org/gestures/index.html, 2005.

[15] Johnny Accot and Shumin Zhai. Beyond fitts' law: models for trajectory-based hci tasks. In *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 295–302, New York, NY, USA, 1997. ACM Press.

[16] Johnny Accot and Shumin Zhai. More than dotting the i's — foundations for crossing-based interfaces. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 73–80, New York, NY, USA, 2002. ACM Press.

[17] B. Ahlstrom and S. Lenman. Touch screen, cursor keys, and mouse interaction. Work with Display Units.

[18] Steven Feiner Alex Olwal. Rubbing the fisheye:l precise touch-screen interaction with gestures and fisheye views. pages 83–84, November 2003.

[19] James A. Landay Lawrence A. Rowe Allan Christian Long, Jr. Implications for a gesture design tool. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 40–47, Pittsburgh, Pennsylvania, USA, 1999. ACM SIGCHI.

[20] Ben Shneiderman Andrew Sears. High precision touchscreens: Design strategies and comparisons with a mouse. January 1989.

[21] Ben Shneiderman Andrew Sears, Yoram Kochavy. Touchscreen field specification for public access database queries: Let your fingers do the walking. In *Proceedings of the ACM Annual Computer Science Conference on Cooperation*, pages 1–7, Washington, D.C., USA, 1990. ACM.

[22] Patrick Baudisch, Edward Cutrell, Ken Hinckley, and Robert Gruen. Mouse ether: accelerating the acquisition of targets across multi-monitor displays. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 1379–1382, New York, NY, USA, 2004. ACM Press.

[23] Michel Beaudouin-Lafon. Designing interaction, not interfaces. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 15–22, New York, NY, USA, 2004. ACM Press.

[24] Benjamin B. Bederson. Fisheye menus. In *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 217–225, New York, NY, USA, 2000. ACM Press.

[25] Nicholas Burtnyk, Azam Khan, George Fitzmaurice, Ravin Balakrishnan, and Gordon Kurtenbach. Stylecam: interactive stylized 3d navigation using integrated spatial & temporal controls. In *UIST '02: Proceedings of the 15th annual ACM symposium on User interface software and technology*, pages 101–110, New York, NY, USA, 2002. ACM Press.

[26] Sheelagh Carpendale, John Ligh, and Eric Pattison. Achieving higher magnification in context. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 71–80, New York, NY, USA, 2004. ACM Press.

[27] Ben Shneiderman Christopher Ahlberg. The alpha-slider: A compact and rapid selector. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 365–371, Boston, Massachusetts, USA, 1994. ACM SIGCHI.

[28] Andy Cockburn and Andrew Firth. Improving the acquisition of small targets. 2003.

[29] G. W. Furnas. Generalized fisheye views. In *CHI '86: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 16–23, New York, NY, USA, 1986. ACM Press.

[30] George W. Furnas and Benjamin B. Bederson. Space-scale diagrams: understanding multiscale interfaces. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 234–241, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[31] Douglas J. Gillan, Kritina Holden, Susan Adam, Marianne Rudisill, and Laura Magee. How does fitts' law fit pointing and dragging? In *CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 227–234, New York, NY, USA, 1990. ACM Press.

[32] Yves Guiard, Michel Beaudouin-Lafon, Julien Bastin, Dennis Pasveer, and Shumin Zhai. View size and pointing difficulty in multi-scale navigation. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 117–124, New York, NY, USA, 2004. ACM Press.

[33] Yves Guiard, Renaud Blanch, and Michel Beaudouin-Lafon. Object pointing: a complement to bitmap pointing in guis. In *GI '04: Proceedings of the 2004 conference on Graphics interface*, pages 9–16, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

[34] Carl Gutwin. Improving focus targeting in interactive fisheye views. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 267–274, New York, NY, USA, 2002. ACM Press.

[35] Stanley Holmes. Starbucks tunes in to digital music. `http://www.businessweek.com/bwdaily/dnflash/mar2004/nf20040311_7142_db016.htm`.

[36] Faustina Hwang, Simeon Keates, Patrick Langdon, and P. John Clarkson. Multiple haptic targets for motion-impaired computer users. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 41–48, New York, NY, USA, 2003. ACM Press.

[37] William Buxton I. Scott MacKenzie. Extending fitts' law to two-dimensional tasks. pages 219–226, 1992.

[38] Douglas W. Jones. Observations and recommendations on pre-election testing in miami-dade county. `http://www.cs.uiowa.edu/~jones/voting/miamitest.pdf`.

[39] James A. Landay and Brad A. Myers. Extending an existing user interface toolkit to support gesture recognition. In *CHI '93: INTERACT '93 and CHI '93 conference companion on Human factors in computing systems*, pages 91–92. ACM Press, 1993.

[40] Neil Willis Lorna Uden. Designing user interfaces using activity theory. In *Proceedings of International Conference on System Sciences*, Hawaii, USA, 2001. IEEE.

[41] Jim Ludwick. Some early voters say machines mark incorrect choices. `http://abqjournal.com/elex/246845elex10-22-04.htm`.

[42] I. Scott MacKenzie. Movement time prediction in human-computer interfaces. In *Proceedings of the conference on Graphics interface '92*, pages 140–150, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[43] I. Scott MacKenzie and Aleks Oniszczak. A comparison of three selection techniques for touchpads. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 336–343, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[44] Martin C. Maguire. A review of user-interface design guidelines for public information kiosk systems.

[45] Jennifer Mankoff. Providing integrated toolkit-level support for ambiguity in recognition-based interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 368–375, The Hague, Netherlands, 2000. ACM SIGCHI.

[46] David Margulius. Going to the atm for more than a fistful of twenties.

[47] Janet Martinson. Issues related to designing touchscreen interfaces.

[48] Michael McGuffin and Ravin Balakrishnan. Acquisition of expanding targets. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 57–64, New York, NY, USA, 2002. ACM Press.

[49] George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63:81–97, 1956.

[50] D.A. Muratore. Human performance aspects of cursor control devices. 1987.

[51] R.A. Murphy. Evaluation of methods of touch screen implementation for interactive computer displays. 1986.

[52] John Smelcer Neff Walker. A comparison of selection times from walking and pull-down menus. In *Proceedings of Computer-Human Interface Conference*, Seattle, Washington, USA, April 1990. ACM SIGCHI, ACM Press.

[53] Shumin Zhai Par-Anders Albinsson. High precision touch screen interaction. April 2003.

[54] John Clarkson Patrick Langdon, Simeon Keates and Peter Robinson. Using haptic feedback to enhance computer interaction for motion-impaired users, 2000.

[55] Shumin Zhai Per-Ola Kristensson. Shark: A large vocabulary shorthand writing system for pen-based computers. pages 43–52, 2004.

[56] J.A. Pickering. Touch-sensitive screens: the technologies and their application. *International Journal of Man-Machine Studies*, 25(3):249–269, 1986.

[57] James A. Pittman. Recognizing handwritten text. In *Proceedings of SIGCHI Conference of Human Factors in Computing Systems*, pages 271–275, New Orleans, Louisiana, USA, 1991. ACM SIGCHI.

[58] Michel Beaudouin-Lafon Renaud Blamnch, Yves Guiard. Semantic pointing: Improving target acquisition with control-display ratio adaptation. In *Proceedings of Computer-Human Interations Conference*, Vienna, Austria, April 2004. ACM SIGCHI, ACM Press.

[59] Jo W. Tombaugh R.F. Dillon, Jeff D. Edey. Measuring the true cost of command selection: Techniques and results. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*, pages 19–26, Seattle, Washington, USA, 1990. ACM SIGCHI.

[60] Ben Shneiderman Richard L. Potter, Linda J. Weldon. Improving the accuracy of touch screens: An experimental evaluation of three strategies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 27–32, Washington, D.C., USA, 1988. ACM SIGCHI, ACM Press.

[61] Judy M. Ireland Robert J. Seiler, Antonette M. Seiler. Enhancing internet access for people with disabilites. `http://www.elr.com.au/eiad/spathrep.htm`.

[62] Dean Rubine. Specifying gestures by example. *Computer Graphics*, 25(4):329–337, July 1991.

[63] Kishore Swaminathan and Steve Sato. Interaction design for large displays. *interactions*, 4(1):15–24, 1997.

[64] Greg Swistak. Photo market not the only one developing bright for kiosks. `http://www.kioskmarketplace.com/news_story.htm?i=21822`.

[65] Gary L. Newell Tyson R. Henry, Scott E. Hudson. Integrating gesture and snapping into a user interface toolkit, 1990.

[66] Max van Kleek. Intelligent environments for informal public spaces: the ki/o kiosk platform, 2003.

[67] Aileen Worden, Neff Walker, Krishna Bharat, and Scott Hudson. Making computers easier for older adults to use: area cursors and sticky icons. In *CHI '97:*

Proceedings of the SIGCHI conference on Human factors in computing systems, pages 266–271, New York, NY, USA, 1997. ACM Press.

[68] Shumin Zhai, Stephane Conversy, Michel Beaudouin-Lafon, and Yves Guiard. Human on-line response to target expansion. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 177–184, New York, NY, USA, 2003. ACM Press.