Using Data About Real World Pointing Performance to Improve Computer Access with Automatic Assessment

Ph.D. Dissertation
June 28, 2010
CMU-HCII-10-104

Amy Hurst
Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon

Thesis Committee

Jennifer Mankoff (Co-chair), Carnegie Mellon, Human-Computer Interaction Institute

Scott E. Hudson (Co-chair), Carnegie Mellon, Human-Computer Interaction Institute

Jodi Forlizzi, Carnegie Mellon, Human-Computer Interaction Institute

Elizabeth Mynatt, Georgia Institute of Technology, GVU Center

This work was supported in part by the National Science Foundation under grants EEC-0540865, IIS-0713509, IIS-0325351, IIS-0205644, and Amy Hurst's NSF Graduate Student Research Fellowship. Additional financial support came from IBM Research; and the Pennsylvania Infrastructure Technology Alliance, the Quality of Life Technology Center, and the Richard King Mellon Foundation.

Keywords: Assistive Technology, Computer Accessibility, Motor Impairments, Data Collection in the Wild, Pointing Input, Measurement, Performance, Human Factors, Usability Analysis, Target Identification

Abstract

Accurate pointing is an obstacle to computer access for individuals with motor impairments. One of the main barriers to assisting individuals with pointing problems is a lack of frequent and low-cost assessment of those pointing problems. We are working to build technology to automatically assess pointing problems during every day (or real world) computer use. To this end, we have studied real world pointing use from older adults and individuals with motor impairments and developed novel techniques to analyze their performance. Our investigation contributes to a better understanding of real world pointing performance, and how to assess pointing performance with machine learning.

Table of Contents

1. Introduction

Computer technology has become an integral component in people's lives. Computers are used in many domains, such as employment, recreation, and socializing. However, computers are not universally accessible and there is a growing population of people who are motivated to use computers, but who find it physically difficult to do so. Over the years, many types of accessibility software, including pointer configuration utilities, screen magnifiers, speech recognition systems, and onscreen keyboards have been developed to support computer access problems. Even though these technologies may be pre-installed on a computer or available online, many individuals with computer access problems do not use them.

This technology may go unused for many reasons, including lack of knowledge about what technology would make a computer accessible, or how to configure that technology. Appropriate technology can be selected with the help of a friend, teacher or caregiver, but this is not always the most accurate approach. A better solution to this problem is to have a trained assistive technology clinician assess a user's ability and help to select and configure software for them. Assessments may include computer-pointing tasks and possibly physical measures of dexterity. After the assessment, the clinician writes a prescription for the assistive hardware or software the individual needs to access a computer. Unsurprisingly, professional assessments are expensive and may not be covered by insurance. As a result, it is common for people to end up using either no accessibility tools, or tools that are not suited for their current pointing abilities. Additionally, the few who are assessed rarely get follow-up assessments, which are important as many users' abilities change over time.

We have spent two years collecting data about computer use from individuals with motor and cognitive impairments. *Pointing*, or moving the cursor to indicate a particular position or on-screen *object* (often called a *target*) is one of the major problems these individuals encounter when trying to access a computer (Brownlow, N., *et al*. 1989; Riviere and Thackor 1996; Trewin and Pain 1999). Pointing problems include activating unintended targets (or interactive regions), difficulty pressing the desired button on a mouse or other pointing device, difficulty correctly clicking on the desired target, and lacking the freedom of motion to control the device. One example of a commonly observed pointing problem is difficulty activating a desktop icon – we have observed participants who spent over two minutes trying to double-click a desktop icon before they were successful.

The goal of this thesis is to enable automatic assessment of an individual's pointing performance during everyday computer tasks. Automatic assessment has many potential applications that may dramatically increase computer access for many individuals. Some possible benefits include increased awareness, automatic configuration, and performance tracking. To this end, we have developed software to observe and analyze pointing use and assess it with predictive models created with machine learning. While the end goals of our research have the possibility to affect many people, we are focusing on individuals who have pointing problems due to age or to a motor impairment.

Our assessment technique is divided into several steps. Real world pointing data is collected with logging software that captures application and operating system events. Pointing data is then segmented into usable chunks for assessment. Finally, performance metrics are calculated from the user's recorded actions and these calculations are used to assess performance using predictive models to provide an assessment. This technique can be extended with computer vision techniques to identify the targets clicked on by the user.

1.1. Thesis Contributions

The goal of this thesis is to assist individuals with pointing problems by addressing a major obstacle to computer access: frequent and low-cost pointing performance assessment. Specifically, we are working to build a system to automatically assess pointing performance during real world use.

Our contributions are

(1) A tool for simplifying collection field data and a field data set containing multiple months of data from 4 individuals with motor impairments, 4 without, and 8 older adults, that can be used by ourselves and others to learn about the performance characteristics of individuals with pointing problems and the viability of novel interactive techniques.

(2) An analysis of our substantial real-world data set demonstrating that it is possible to assess computer pointing metrics on real world data. We also found the following three trends: 1) Older adults and individuals with motor impairments have "better" performance during games than other applications; 2) Individuals with motor impairments had more targeting problems than older adults or able bodied participants; and 3) Able bodied participants have faster movement than older adults or individuals with motor impairments.

(3) A tool capable of detecting problematic pointing performance using predictive models constructed using machine learning techniques. We demonstrate which features are important for the prediction of pointing performance and show that we are able to detect problematic pointing performance while surfing the web and using the Microsoft Office suite with high accuracy.

(4) A technique that lays the groundwork to improve analysis of real world data by automatically identifying targets (or interactive regions) collected by our logging tool. Our target identification technique found 15% more targets than the standard accessibility API.

This thesis is written in eight chapters. Chapters 2 and 3 set the scene for the primary research contributions. Chapter 2 discusses related research to study pointing performance and real world use. This chapter also includes a description of the machine learning methodology we use to assess performance. Chapter 3 presents a pilot study we did to automatically assess performance based on pointing behavior. In this pilot study, we distinguish between novice and skilled use without a task model and built a system that used real time assessment to assist the user.

Chapters 4 through 7 describe each research contribution in depth. Chapter 4 describes a study protocol and tools we developed to collect a large dataset of real world pointing behavior from these individuals. This chapter includes a description of our logging software, participants recruited and limitations and lessons learned from this technique. Chapter 5 presents our results analyzing real world data in two evaluations that analyze real world pointing use. In this chapter we describe the data we collected and present how we used performance metrics traditionally used to study pointing performance collected in a laboratory study to better understand user ability and natural tendencies during real world use. In Chapter 6 we discuss how pointing performance can be frequently and unobtrusively assessed with predictive models created with machine learning techniques. Chapter 7 describes new techniques we have that will likely improve automatic assessment through improved target recognition and prediction of whether a particular adaptation will improve performance. Chapter 8 concludes by summarizing the work presented in this thesis.

2. Related Work

We discuss work related to this thesis in four sections. The first section defines laboratory and real world data collection and discusses the tradeoffs of both. The second section two discusses foundational studies of pointing performance that were conducted in the laboratory. Next, we present studies investigating real world computer use and pointing performance and the tools used to collect this data. We discuss the limitations of these tools and how we adapted one of them to create our own logging software. The third section presents some common pointing problems that older adults and individuals with disabilities have exhibited in laboratory studies, and that we wanted to measure in our own analysis of real world use. Following a description of common pointing problems, we discuss software adaptations other researchers have developed to address these problems. One of the main gaps in the literature is matching pointing problems to existing adaptations in real world use. We envision using predictive models to fill this gap, and discuss some of the machine learning methods we draw from in this thesis to detect real world pointing problems.

2.1. Tradeoffs Between Laboratory and Real World Data Collection

As we will discuss in this thesis, studying real world data produces a more realistic understanding than laboratory data, but it is more difficult to collect and understand this data than laboratory data. Collecting large datasets of from individuals with pointing problems can be challenging because these individuals may have limited ability to participate in a research study. Specifically limitations from fatigue may prevent an individual from participating for long uninterrupted sessions. In this section we will describe the tradeoffs between real world and laboratory evaluations and why we have chosen to collect real world pointing data.

Studies conducted in a laboratory setting are normally intended to simulate or mimic key features of real world, or natural, interactions. However, to improve control and increase the quantity of data collected, laboratory tasks frequently have users repetitively perform unrealistic tasks such as moving a target to a particular target dozens or even hundreds of times. In addition to collecting data from unrealistic tasks, laboratory data may not be representative of natural behavior. Issues may arise related to: 1) taking place in a location unfamiliar to the participant (possibly with equipment the participant hasn't used before), 2) performing actions while being observed by the experimenter, 3) completing tasks for compensation – which can affect motivation as some studies reward accuracy while others reward speed, and/or 4) variability in performance as the participant may become fatigued and not take necessary rest.

Since better assessments can be made from more examples of performance, it is important to also take the participant's abilities and needs into consideration to collect more data. Conducting research with individuals with disabilities in a laboratory can be more burdensome and less realistic for the participants than conducting the research in their home or office. Conducting research in environments participant are familiar with can be easier for the participant because they are more likely to have a physical environment that fits their needs (Coyne and Nielsen 2001). Additionally, putting study logistics (such as travel) on the experimenter, rather than the participant can reduce barriers to participation.

As predicted by the Hawthorne Effect, a user's performance may change simply because they are being observed (Mayo 1933). Additionally, laboratory studies can be negatively impacted by the demand effect, wherein a participant assumes the role of trying to please the experimenter (Orne 1962). In a performance study with individuals with limited mobility, this could manifest itself as the participant ignoring fatigue, pushing through the study without breaks or stopping. These influences as well as issues of motivation, task comprehension, and familiarity with an input device have the potential to negatively impact performance. Finally, laboratory studies have a resource limitation since most require the experimenter to observe the participant complete the tasks. This limits the amount of data that can be collected.

Studying real world pointing performance can help an experimenter overcome many of the limitations of laboratory studies. However, real world evaluations have their own limitations. One such limitation is not knowing a user's intent. In a laboratory study, the participant is given a very specific task, such as “move the cursor to the box and click” and is scored on how long it took to do that task and how accurate they performed. However, real world computer use can be much more ambiguous as there is seldom only one possible correct action. One example is when the cognitive components of interaction, which are usually controlled for in a laboratory study, further confuse the data (such as deciding at the last minute that a button being moved towards should *not* be pressed). With real world data the experimenter must divine what action was completed and whether that was the user’s intent.

We chose to study pointing performance during real world use instead of through laboratory studies so we could collect a large and realistic dataset from individuals with limited abilities. Our data collection approach reduces study logistics for our participants, allows them to participate with minimal changes to their pre-existing routines, and maximizes the amount of data we can get per person. Part of the contribution of this thesis is a set of techniques which (partially) overcome the limitations and difficulties of real world data collection while retaining the benefits. The next section describes related studies of computer use, and how this data was analyzed in both laboratory and real world evaluations.

2.2. Related Real World and Laboratory Studies of Pointing Performance and Computer Use

This subsection describes previous studies of both real world and laboratory-based pointing performance and computer use related to the work presented here. We will briefly describe some of the many laboratory studies of pointing use, then summarize a few related studies of real world computer use and one study of real world pointing performance. Next, we will focus on tools and techniques that have been developed to evaluate pointing performance (in both laboratory and real world studies): such as tools used to collect user events, detect targets selected, and segment event streams.

2.2.1. Laboratory Studies of Computer Use

There is a long history in the field of Human Computer Interaction (HCI) of studying pointing performance in a laboratory setting. Performance data is frequently collected in the laboratory by instructing participants to perform repetitive pointing tasks and measuring the movement time and accuracy of each trial. In these studies, it is not unusual for the experimenters to use custom software developed to execute and collect data for pointing performance studies. Examples of two common tasks users perform in these studies are illustrated in Figure 1. Figure 1A illustrates a task where the user is asked to move between a series of circular targets arranged in a circle in a predefined sequence. Figure 1B illustrates a task where the user moves between square targets of varying size and distance from each other. During this task each new target appears after

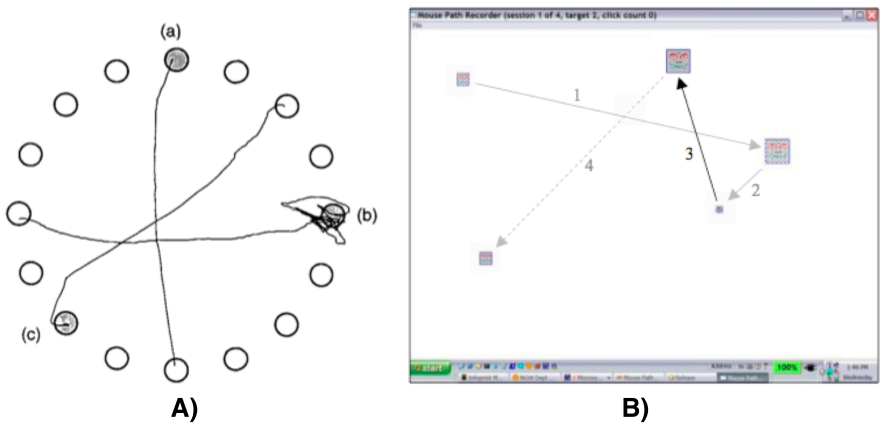

(a)
(b)
(c)
Mouse Path Recorder (session 1 of 4, target 2, click count 0)
1
2
3
4
A)
B)

Figure 1 Examples of software typically used to measure pointing performance. **A)** Software used by Keates to develop cursor measures for individuals with motor impairments. Participants moved cursor between circles in a pre-defined sequence (Keates *et al.* 2002). **B)** Software used by Keates to measure effect of age and Parkinson's disease while pointing (Keates and Trewin 2005). Participants moved cursor to targets of varying size as they appeared in a pre-defined sequence and pressed the mouse button when they reached the target.

the user clicks on the current target.

Pointing studies conducted in the laboratory have resulted in models of human performance that have been proven to be robust across many conditions. Some of the many findings include equations that predict the movement time between two targets in one dimension (Fitts 1954; Douglas *et al.* 1999), two dimensions (Mackenzie and Buxton 1992), in human movement within constrained paths (Accot and Zhai 1997), and while scrolling (Hinckley *et al*. 2002). These models of pointing performance have been used to investigate how tradeoffs in pointing device technology design affect performance, and how Graphical User Interface (GUI) design and interaction can improve or hinder performance.

2.2.2. Studies of Real World Computer Computer Use

Studying real world computer use can help to evaluate new or existing software for usability problems, learn what tasks the software is being used to accomplish, or identify patterns and trends in current application use. As mentioned in Section 1.2, the findings of real world studies often go beyond what can be evaluated in the laboratory as conditions and constraints are less artificial than evaluations in the laboratory.

Studies of real world use range from studying general computer use, to analyzing patterns of use with text editors and document readers, to studying pointing behavior. Beauvisage has studied computer use in daily life in a long (19 months) field study (Beauvisage 2009). In this work he collected a huge dataset of computer use from over 600 homes that included over 1400 users. Log files of application and operating system events were analyzed to uncover five profiles of software preferences and usage frequency. His study illustrates some of the routine behaviors in computer usage at home.

Bingham studied the web browsing behaviors of sighted and blind users during one week of real world use (Bingham *et al.* 2007). They used this data to understand the differences in websites visited by these populations, how blind individuals coped with content that was inaccessible through their screen readers. They also found that blind participants surfed the web much more slowly than sighted participants.

Studies looking at single application use can reveal surprising patterns of use that can be applied to the design of future systems. Alexander and Cockburn have studied document navigation in Microsoft Word and Adobe Reader and uncovered surprising trends in use (Alexander and Cockburn 2008). They found that half of documents opened had already been opened by the user, and that users rarely used document lists (i.e. menus listing most recently used documents) to reopen a document. Whiteside studied how people used early text editors (Whiteside *et al.* 1982), examining the use of text editors for knowledge workers and secretaries. They found that text entry only accounted for half of all keystrokes in those early systems, another quarter was accounted for by cursor movements, an eighth by deletion and the rest by miscellaneous functions. Based on these findings, the authors made suggestions for editor and keyboard design.

Chapuis has conducted one of the few investigations of real world pointing performance (Chapuis *et al.* 2007). They collected several months of pointing data from individuals as they performed their own tasks. The logging system the authors used took advantage of an accessibility *Application Programming Interface (API)* (facility built by the major software vendors to report low-level application information) to get the size and location of many of the targets in their dataset. This dataset was used to analyze the movement time between targets to evaluate how well Fitts' Law held up during real world computer use in XWindows and OSX. The authors found Fitts' Law performance studied in the lab can apply to movement in the field.

These areas of real world computer use illustrate some of the benefits of studying computer use outside the lab: learning more about what features in an application are used, identifying patterns of use, and validating a performance model developed from laboratory use transfers in the real world. However, to our knowledge, this research was not conducted with individuals with pointing problems due to motor or cognitive impairments. One of our goals with this work is to better understand the pointing abilities of individuals with pointing problems, and how performance fluctuates during real world use. Gaining a better understanding of their abilities will help design systems that can accommodate their pointing problems.

2.2.3. Tools Developed to Study Real World Computer Use

In order to study real world use, software that captures and records user actions is needed. One of the main obstacles to studying real world computer use is finding *logging tools*, or software that records information about user actions, that capture enough information about interactions for a given study. For example data collection may only be needed for one application, or it may need to work across applications or operating systems. The level of detail needed about user actions may also vary according to the study goals. This may range from merely needing general, or high-level logs of the applications used, to more detailed, or low-level logs that include the size and location of all buttons the user interacts with. This section presents software for three main problems encountered when trying to collect and analyze real world data: logging user events, identifying target sizes, and segmenting real world data.

2.2.3.1. Tools to Capture User Events

Many tools have been developed to capture event streams from input devices such as mice and keyboards, as well as windowing system events and operating system events. Typically these logging solutions are tied to a specific operating system or windowing system. One example of this kind of system is DART which runs as a background process, and logs the name, size, and location of all windows on a computing system, noting the opening and closing of windows (Iqbal and Horvitz 2007). Another similar system is RUI, a keystroke and mouse event logger that has been written for both the Windows and OSX operating system (Kukreja *et al.* 2006).

In addition to collecting low level event streams from user interactions, there has also been research to collect higher-level information about real world use. Hilbert surveys some of the technology as of 2000 to extract usability information from user interface events (Hilbert and Redmiles 2000). This includes analyzing logs of interactions for sequence detection, sequence comparison, counts and summary statistics, search techniques and synchronization with other data sources such as video, images, or audio). Kim developed the TRUE software to track real time user experience (Kim *et al.* 2008). This software was developed to collect real time information in video games, such as Halo 2. TRUE logs sequences of events, collections of events that are combined into "sets", attitudinal data from participants collected via popup surveys, and collects video that is synced to log files. In order to investigate the web browsing behaviors of sighted and blind users, Bingham extended UsaProxy (a tracking proxy). This proxy recorded the content the user saw, and statistics about interactions by that records key presses, mouse events, and focus events. (Bingham *et al.* 2007)

These data collection tools were developed to study large datasets of both pointing behavior and more general computer use. Of these tools, we found DART to be the most application independent and unobtrusive logging system. As a result, we chose to extend DART to create CRUMBS (described in Section 4.2), which we used to collect our dataset.

2.2.3.2. Tools Developed to Automatically Identify Targets

The software described in the previous section is useful to learn low-level information about mouse, keyboard and operating system events, all of which are relatively easy to analyze and collect. However these tools are not sufficient to use to automatically detect pointing performance because they are either application specific, or they do not provide enough information about pointing actions. One solution is to also collect higher-level information about the user's actions, such as application independent information about the *targets* the user interacted with.

Targets are the interactive regions in a graphical user interface that a user can click or hover over to fire an action. An example of a frequently used target is the Start Button in the Windows operating system. Having access to the size and location of interactive targets (or the specific targets a user interacted with) is necessary to study pointing, as performance depends on target size.

There are several pre-existing techniques to find targets in a GUI, but these are usually application or toolkit specific. One technique is to enable target identification in the software development phase through API support. The Microsoft Active Accessibility (*MSAA*) is one of several APIs that could be used to find targets (Microsoft Active Accessibility, www.msdn.microsoft.com), including the JAVA (Java Accessibility, www.java.sun.com) and Flash Accessibility APIs (Adobe Accessibility, www.adobe.com). Unfortunately, using these APIs is not always possible because they require the developer to compile the program with accessibility included. Additionally, once compiled, these often have runtime limitations (the Flash API only works in Microsoft Explorer and requires ActiveX). Another major limitation to these APIs is that they may not support all targets in a given application or toolkit. Frequently, these APIs are released with tools that make it easy to determine what targets are supported. For example, Microsoft has released the inspect32.exe program (inspect32.exe 2004) for the MSAA API. This is an application that can display all API information available about a target currently under the cursor.

Given the limitations of using accessibility APIs to identify targets, a less restricted solution is desirable. Instead of using programmatic controls or hooks to find the targets, another approach is to analyze the visual components of the screen. One set of solutions that does this requires the user to manually identify targets. For example, the Eggplant Functional Tester allows users to manually select targets for a scripting application to interact with (Redstone Automated Software Application Testing Tools, http://www.redstonesoftware.com/). Users define targets by dragging the cursor diagonally from the upper left corner of a target to the lower left. After a target has been defined, Eggplant automatically scans the GUI for any regions of the screen that exactly match this selection. Any targets that are similar but not identical (for example buttons of the same size but different text) to a selection are not matched by Eggplant. Another tool that which uses the same strategy to let users manually define targets is Sikuli, which allows users to graphically write GUI automation scripts (Yeh *et al.* 2009).

Additionally, there has been success in using computer vision techniques to analyze onscreen content for potential targets. However, the accuracy of these techniques is very tied to *a-priori* information about the types of *interactors* (or interactive regions) the user will encounter and may not be able to easily handle new examples easily. While successful solutions for scripting applications, these target recognizers are limited in that they can only identify targets that a user pre-defined. Instead, a more generalizable approach to visual target identification is desirable. Seminal work in the area of visual analysis for identifying user interface targets has been done by Amant to support cognitive modeling and programming by example tools (Zettlemoyer and Amant 1999, Amant *et al.* 2000A, Amant *et al.* 2000B). More recently Fogarty and Dixon developed PreFab (Fogarty and Dixon 2010) to recognize interactors in interfaces to apply pointing assistance. Both of these techniques recognize targets by searching the screen for targets based on a set of generic targets that are in a database (rather than searching for exact matches). Unfortunately, it is difficult to understand how well these techniques would work with real world datasets because no accuracy numbers are available, and these researchers tested limited applications.

These pre-existing techniques illustrate that while Accessibility APIs can provide much target information, they are not a complete solution, as they do not support all interactors. Additionally, there has been success in using computer vision techniques to analyze onscreen content for potential targets. However, the accuracy of these techniques is very tied to a-priori information about the types of interactors the user will encounter and may not be able to easily handle new examples. In Section 5.2 we present a technique we developed to automatically detect targets that combines an Accessibility API with visual GUI analysis to correctly recognize 15% more targets than the accessibility API did alone (an improvement from 74% to 89% accuracy).

2.2.3.3. Segmenting the Event Stream to Reveal Recognizable Interactions

Data must be segmented into *recognizable interactions,* or user actions (or portions of user actions), which can be readily isolated. These are indicative of the phenomena needed to study, model or predict, and can be easily and accurately labeled. Many common interactions can be modeled as one of a small class of well-understood movement types. Examples include moving the thumb of a scrollbar, steering through a menu, and acquiring a target such as a button or checkbox with the mouse. Each of these activities is well defined, the first is a crossing task, (Apitz and Guimbretiere 2004), the next is a steering task (Accot and Zhai 1997), and the third is a pointing task (Mackenzie and Buxton 1992). Abstract and/or “realistic” versions of these interactions are often used in lab studies. Because these are well defined, with known targets or well-described tasks, they can easily be used to measure user performance.

Segmenting real world data into potentially meaning subunits is less straightforward than segmenting data collected in the laboratory. This is due to the semi-ambiguous nature of real world data (i.e. the experimenter does not know the task the user intended to perform). In order to compensate for not knowing user intent, real world data should be segmented according to meaningful subunits in the event stream (such as pointing, window, or keyboard events and pauses). Given our interest in studying pointing performance, we are interested in segmentation strategies that are based on pointing use such as button events, and cursor movement. The type of segmentation strategy used depends on the kind of analysis being performed.

Segmentation by Button Event: Segmentation by pointer button events is a very simple segmentation algorithm to execute as it simply cuts the data at button up and button down events. This simple strategy can be very useful for automatic target identification since pointer button events are usually executed over a target. Segmentation by button events is also useful for metrics of performance that are focused on target selection. These methods of performance include measuring how long it takes a user to move between targets, the time to press a target, or if they slipped off the target while trying to activate it (Keates and Trewin 2005, Hurst *et al.* 2008B).

Segmentation by Movement: Not all interactions in real world pointing use involve clicking on a target, nor do they all start or end with the user pressing or releasing a button on the pointer, so it is important to also consider segmenting the event stream with other events. For example, there are several intentional interactions that do not require the user to press the mouse button, such as hovering or pausing over a target (perhaps to activate a tooltip) or changing speed to move to a target (perhaps in a video game). Additionally, paying extra attention to these actions may help with assessment because these actions maybe indicative that the user is having pointing problems, such as drastic changes in cursor direction frequently occur when a user overshoots a target (Hwang 2003, Phillips and Triggs 2001). Finally, segmentation by button events alone does not help to identify the start of a targeting motion, so it is important to also consider segmenting the event stream by cursor pauses.

Segmenting cursor movement by pauses or movement changes can be very informative. The duration of a pause (or period without cursor movement) is one piece of information that may be able to distinguish between able bodied and motor impaired use. In a laboratory pointing evaluation, Keates found that able bodied users have an average pause duration during pointing tasks is around 70 milliseconds, while the average pause time for motor impaired individuals pause is longer (110 milliseconds) (Keates *et al.* 2002B). Furthermore pauses longer than 250 milliseconds likely represent approximate reaction time to a stimulus (Keates and Trewin 2005) and may be a good pause threshold to segment actions. Cursor movements have been defined to begin with at least 100 milliseconds of pointer motion with a speed greater than zero (Hwang 2003; Walker *et al.* 1993).

Due to the freeform nature of real world data, segmentation may need to be different from what is typically done in the lab. Specifically, real world data is much more likely to have more pauses during movement data, because the user isn't forced to complete a sequence of movements. Common potential causes of additional pauses include when the user is reading the screen, wiggles the mouse to "wake up" the screen or simply walks away from the computer. When segmenting real world pointing data, Chapuis looked at both button events and movement metrics to segment their data. Specifically, they segmented data into samples of movement that ended with a button down event. They further segmented their data so it only included movements that were at least 10 pixels long, and did not have any pauses longer than 300 milliseconds.

2.2.4. Summary of Related Real World and Laboratory Studies of Pointing Performance and Computer Use

In this subsection we have described related studies of pointing performance and computer use in both laboratory and real world settings. While we have only found one related example of studying real world pointing across multiple applications, these related studies are useful resources to learn how to collect and analyze computer use. In fact, much of our study design and analysis tools draw on this body of related work for inspiration. The following section describes some of the pointing metrics that were used in laboratory evaluations of pointing performance that could also be used to assess pointing performance.

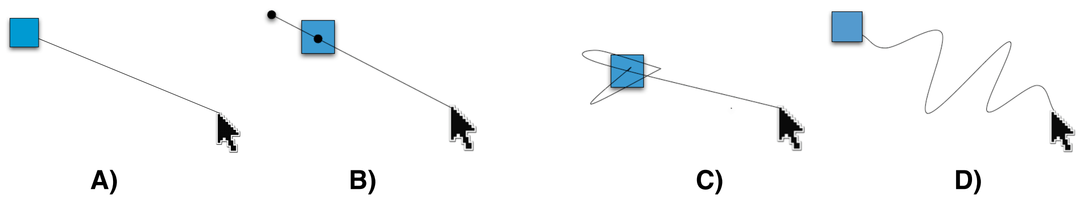


Figure 2 Examples of ideal pointing A and pointing problems (B-D). **A)** An example of error-free pointing where the user moves the cursor to the blue square (in a straight line) and clicks on the square with the left pointer button. **B)** Clicking Errors occur when the user fails to press a single button on a pointing device without moving the pointer. **C)** Targeting problem occurs when the user has difficulty pressing the mouse button while the cursor is on a target. **D)** Movement problems occur when the trajectory between the start and end of a movement are far from optimal.

2.3. Common Pointing Problems and Existing Adaptations

Physical difficulty accessing a computer can be caused by many factors including a physical impairment or age-related changes in motor coordination. Pointing errors often occur when the user knows what he or she wants to do but has difficulty completing the physical action. Figure 2 illustrates some of the differences between an ideal pointing motion (Figure 2A) and some common errors that may happen during that action (Figure 2 B-D). Below is a list that describes many common pointing problems reported in the literature (adapted from Paradise *et al.* 2005) and some of the software solutions that have been designed to minimize those problems. This list is divided into three categories of pointing subtasks: clicking, targeting, and moving. Our list focuses on errors that could be easily addressed with software. The problem is listed in italics followed by possible software solutions to that problem.

2.3.1. Clicking Errors

A user "clicks" when they press and release the button on a pointer device to interact with a target. There are several possible problems an individual can encounter while interacting with the buttons on a pointing device, such as a mouse.

Pressing the incorrect physical button can happen for many reasons including not knowing which button to press or not noticing that one's finger is on the wrong button. In addition to pressing the wrong button it is not uncommon for an individual with pointing problems to unintentionally press multiple buttons at once. For example, in a laboratory study of the effect of an adaptation to assist in clicking on performance, Trewin used clicking errors (when the wrong mouse button is pressed) as a metric of performance. In her control condition, with no adaptation present she observed 27 clicking errors, and in 13 of those errors both buttons were pressed (Trewin *et al.* 2006). Solutions to help with this problem include remapping button functions or replacing clicking with another style of activation such as crossing, where the user does not need to press any pointer button (Accot and Zhai 2002).

Accidental clicks occur when a user presses a pointer button when he or she did not intend to. For example, a user may click partway through a mouse motion to a target, or may click the mouse button without moving the pointer (Trewin and Pain 1999). One possible solution to help an individual who has problems accidentally clicking is to ignore pointer button events when the cursor is moving.

Repeated clicks are additional clicks made during a single click. One strategy to assist users with this problem is to ignore multiple clicks on the same target, or set an upper bound for the clicking rate.

Double click timing errors can be frequent as double clicking is a common task in modern GUIs, especially when interacting with items on a desktop. Double clicking is more difficult than single clicking for individuals with pointing problems. For example, in a laboratory study of multiple clicking (Trewin and Pain 1999), Trewin asked motor impaired users to perform six double clicks and two triple clicks. Participants had an error rate of 39.5%. Trewin also found a median error rate of 28.3% for single clicks. Able-bodied users had a much lower error rate on multiple clicks (9%), and a median error rate of 6.3% for single clicks. Across participants, these errors were the result of positioning errors, moving during a click, or long delays between clicks. In this study, the delay between clicks in the motor impaired group ranged from 100 milliseconds to 333 milliseconds. Click times for the able bodied group were between 50 milliseconds and 100 milliseconds. Fortunately, adjusting the time between clicks required to make a double click is a common accessibility option in modern operating systems.

2.3.2. Targeting Errors

Difficulty with target acquisition is a common problem as most of the interaction in a Modern GUI involves targets, or regions of the interface for the user to interact with. The following subsection describes some of the many target acquisition techniques that have been developed. All of these techniques require complete knowledge of available targets, something that is easy to achieve in a laboratory study, but much harder to do in the wild. According to Balakrishnan, Object Pointing (Guiard *et al.* 2004) has the best performance gains when trying to radically help with target acquisition, or beat Fitts' Law (Balakrishnan 2004). This technique removes the time a user spends moving to a target, by allowing a user to "jump" to any onscreen target. Sutherland's thesis includes the first published example of a technique, now termed *snapping*, for making it easier to acquire targets (Sutherland 1964). He computes a "pseudo pen location" by moving the pen (a pointing device) to an object of interest if it is within a certain radius of that object. Since then, a range of related techniques that manipulate the control-display gain (ratio between the physical distance moved by the pointer and the effect on the pointer) have been developed and tested (Balakrishnan 2004). For example, Snap-and-go supports snapping to a target by increasing the virtual size of the target by slowing down the cursor when it is over the target, thus creating the illusion that the pointer stops on the target before moving forward again (Baudisch *et al.* 2005).

Pseudo-haptic and haptic approaches support target acquisition using a force model (i.e. changing how the cursor behaves with software or a physical device). Oakley laid out empirically based guidelines for the design of haptic user interfaces, and what the user should be able to configure (Oakley *et al.* 2002). Most work in this area has focused on the design of haptic interactors, and thus assumes that the location, use, and shape of those interactors are known.

Predicting the target of cursor motion (based on a known set of targets) also relies on knowing the location of targets). While an exciting area of research, target prediction (Lank *et al. 2007)* can be error prone, and not always useful as it may not be able to correctly detect the desired target until the user is more than halfway towards it. Despite these drawbacks, target prediction research has matured sufficiently to be deployed successfully in carefully designed adaptations supporting target acquisition (*e.g.,* (Lane *et al.* 2005; Rogers *et al.* 2005). These applications were designed around constraints of their recognition system (accuracy, and time required to make target prediction) and evaluated in laboratory studies.

Difficulty staying on target can occur when the user moves the mouse during the button press and falls off the target. Trewin has developed an adaptation that freezes the cursor while the button is pressed to help with this type of error (Trewin *et al.* 2006). Trewin's "Steady Click" adaptation was developed to assist users with problems slipping off a target by disabling dragging during a click (i.e. the user is not able to move the pointer, or slip, while one of the buttons is pressed). It was found that this adaptation significantly reduced slipping errors for 8 of the 11 participants, and lead to significantly improved target acquisition times for 5 of the 8 participants. For some participants, suppression of slipping errors did not significantly improve performance because other targeting errors remained. For others, target acquisition times were reduced, but the improvement was not statistically significant perhaps due to not collecting enough examples of slips.

2.3.3. Movement Problems

Some individuals may experience difficulty keeping mouse motion steady, which may be the result of tremor or spasticity. Not being able to keep steady motion can make target acquisition difficult, particularly if the desired target is small. Path navigation, or movement towards a target, has received far less attention than target acquisition in the literature. However, Jul's work on lodestones and leylines demonstrates that the combination of attractive targets and constrained motion to those specific targets significantly decreases time to acquire a target (Jul 2003). Additionally, work has shown the value of force fields (directing or pulling the cursor in a given direction) for supporting menu navigation (Ahlström 2005). Koester has explored the effects of tailoring the cursor's gain (a measure of how far the cursor moves relative to the amount of physical movement sensed by the input device) to a user's pointing performance based on movement during target acquisition tasks. They saw that cursor gain had a positive or negative effect on the following features (based on the individual's ability): percent of error free trials, cursor entries and overshoot (Koester *et al.* 2005). Finally, movement problems could be addressed by filtering spurious pointer movements and smoothing all pointer motion (Levine and Schappert 2002).

2.3.4. Changing the Interface to Meet User Needs

Another alternative to the assistive software described above (none of which alter the appearance of the GUI) is to change the interface to suit a user's ability. For example, all the interactors of a software application could be replaced with ones tailored for the user's input device. Carter demonstrated this technique in a reconfigurable system that could automatically replace the interactors in any JAVA application with interactors appropriate for users with a variety of input constraints. They demonstrated this tool by modifying GUIs to meet the needs of the following four configurations: 1) keyboard and mouse, 2) only a keyboard, 3) only a mouse, 4) or a single switch button (Carter *et al.* 2006). An alternative approach to tailoring interactors to a specific device is to tailor them to a user's ability. Gajos has developed a technique that chooses an interactor, its size, and the distance between interactors based on the outcomes of a pointing performance test (Gajos *et al*. 2008).

2.3.5. Summary of Common Pointing Problems and Existing Adaptations

In this subsection we presented some of the commonly studied pointing errors individuals with pointing problems encounter and several of the existing techniques to support individuals with these problems. While many of these techniques have the potential to dramatically improve computer access for individuals with pointing problems, one of the main obstacles to adoption is assessment, or knowing when to apply any given technique and how to configure it for current performance. In this thesis, we aim to use predictive models created with machine learning techniques on real world pointing data to identify problematic pointing behavior. The next section gives a brief summary of the machine learning techniques we use to assess pointing.

2.3.6. Machine Learning Methodology Used In This Thesis

In this thesis we use machine learning techniques to automatically classify user performance data into specific categories. We use these techniques because they are often able to produce classifiers of higher quality than those created using heuristics or simple decision procedures. The following section describes some of the machine learning methods we use to assess pointing performance.

Assessment of pointing performance can be described as identifying particular patterns of interaction as belonging to specific categories. In fact, this is similar to many other problems that can be represented as deciding whether some input data belongs in a specific category. A category, or *class*, is a collection of instances that share some common characteristics that identify them as members of the class. Instances that lack these characteristics belong to another class. A *classifier* is an algorithm that is able to identify whether a given instance belongs to a given class. Machine learning techniques are used to train classifiers to make these categorizations from example data.

For any given dataset we segment it into *training instances* (or simply *instances* for short). For example, if we were going to classify levels of pointing performance we would define an instance as movement followed by a click on the target. Each such instance is *labeled* with an indication of the properties of that motion. One example of a label could be an indicator of whether the user who performed it had known pointing problems or not. It is these labels that our statistical models, or classifiers, attempt to predict. The best way to build models that will generalize to a range of situations is to use large datasets that represent a wide range of data behaviors.

Information associated with the details of each movement instance is summarized into a set of *features* that quantify the performance of each interaction. Examples of potential features to describe a pointing task include the amount of time it took to perform the instance, the number of pauses during movement, or a count of the number of times the mouse button was pressed.

A statistical model works by finding what amount to correlations between the occurrence of certain features and the occurrence of the property it tries to predict (the label for each instance). Once a statistical model has been created (*learned*), it can be used to classify new data. For example, we could use a statistical model with features associated with a new motion and make a prediction about the properties of the user who made that motion and if they would benefit from a particular adaptation (which we do in Section 7.2).

Since not all features necessarily contribute to creating a good classifier, we employ *feature selection algorithms* to provide an indication of how useful each feature may be for constructing a classifier. To select features that are finely tuned to a particular learning algorithm, while taking into account any information overlap, we frequently employ a wrapper-based feature selection approach (Kohavi and John 1997). This approach performs a combinatoric optimization to choose the subset of features that produces the best accuracy for a given type of classifier. This technique is called wrapper-based because it can be "wrapped around" any existing learning algorithm. Although this technique is typically computationally expensive – it creates and evaluates a very large number of different classifiers – it tends to do a very good job of feature selection. The additional computational expense is irrelevant during our model building because we build these models offline and the reuse the resulting model many times at run-time.

Many of our classifiers are built with the C4.5 Decision Tree learning algorithm (Quinlan 1993) as implemented in the WEKA machine learning environment (Witten and Frank 2005).

These classifiers are evaluated with their classification accuracy, or the number of classifications that were correct (or we expect to get correct). In addition to classification accuracy, we use the Kappa Statistic to measure the agreement between predicted and observed classifications of the dataset (Witten and Frank 2005). This measure corrects for agreement that occurs by chance, and is reported as a number between 0 and 1, where 1 is 100% agreement, and a value of 0.7 is often considered acceptable agreement (Krippendorf 1980). The *prior probability*, or simply *prior*, is also important to understanding the accuracy of a prediction. The prior is the accuracy one would obtain using a trivial classifier that always selected the most frequent class as its prediction.

We can estimate a classifier's *generalizability*, or how it will perform on new data, using *cross validation*, a technique that simulates the evaluation of a trained model's performance with previously unseen data. During cross validation, a *test set* of data is withheld from the total collection of example data. The remainder, or *training set*, is used to build a model. This model is then used to predict labels for each sample in the test set, producing an accuracy measurement for the trained model. This process is repeated several times with different testing and training splits, and an average of the performance values is taken as an estimate of a trained model's performance in the real world.

We used two different methods for generating test and training sets, *random* and *per-person* hold out. The random holdout method creates a test set by randomly selecting samples from the entire dataset. Random holdout ensures that samples from all participants are represented in both training and test sets, but it may produce artificially high accuracies as samples from a single session may end up in both the test and the training sets. The per-person holdout method creates a test set from all samples from one or more participants. This holdout method simulates how the classifier will perform on data from a new person, eliminating the bias from splitting a single person's data across training and testing sets. However, per-person holdout may leave certain populations underrepresented in training data, thus reducing classifier performance for members of that population.

2.4. Summary of Related Work

This related work section described a wide range of related work that we will draw on throughout this thesis. We began with a discussion of the tradeoffs between laboratory and real world data collection and described why we chose to focus on studying real world data. Next we presented related studies of both laboratory and real world computer use and pointing performance, which we used as inspiration when designing our study protocol to gather real world data (Chapter 4). This section also included a discussion of the tools used to collect and analyze pointing data. This thesis will use much of the work described in the second subsection to develop our own data logging tool (Section 4.3), segment our real world data (Section 5.1.1.2 and 5.2.1.3), and understand what target a user selected (Chapter 7).

In the third subsection we described pointing performance metrics that will be used to detect specific pointing problems, and assess pointing performance (Chapter 5). The final subsection discussed the machine learning techniques we will use to assess real world pointing data. These techniques will be used throughout this thesis in Section 3.2, Chapter 6 and chapter 7.1.3.

3. Automatically Assessing Pointing Behavior: Pilot Software to Distinguish Between Novice and Expert Use

This section discusses work to automatically assess pointing behavior in near-real world use, without using a task model (Hurst *et al.* 2007A). Our results show that it is possible to automatically distinguish between novice and skilled use by observing user actions. We present this work to illustrate our process to collect, analyze, and assess performance data in a real world application. This work also shows how performance assessments can be made in real time and used to assist the user. This work is limited as it only looks at one model of performance and uses limited collection software that is restricted to one software toolkit.

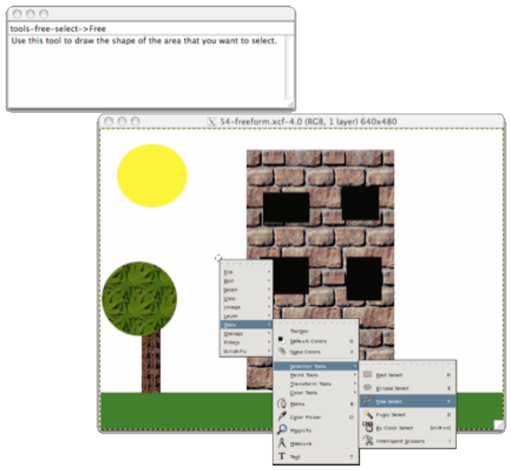

tools-free-select->Free
Use this tool to draw the shape of the area that you want to select.

Figure 3 Prototype system that tailors a help information tool that is tailored to the user based on whether they were detected an expert or a novice.

In this work we built a learned statistical model to distinguish between novice and expert use with 91% accuracy. This model was used in an adaptive application that tailored help messages to current expertise. This section describes how data was collected to build these models, how the models were built, and how an adaptation was deployed and integrated into a pre-existing application. Finally, this section discusses the limitations of studying menu use in an image manipulation program and describes how these limitations suggested the need to study more general real world use.

3.1. Data Collection

To produce a predictive model that will classify real world use, it is important to collect data that is representative of real world behavior. Additionally, in order to build robust learned statistical models, we need to collect a large number of labeled examples (to serve as ground truth) of the behavior we want to classify (expertise). Users may not be able to label their actions as novice or skilled, and would likely find it burdensome to report this frequently (*e.g.* after every action). Our solution for this proof of concept pilot work was to have users perform realistic activities in a real world application, but in a laboratory setting where we could ensure that users passed through a learning curve to provide both novice and skilled data for similar actions (at different points in time). In this study, we had users interact with the GNU Image Manipulation Program (GIMP 2.2.8, www.gimp.org), a widely used open source image manipulation and editing application.

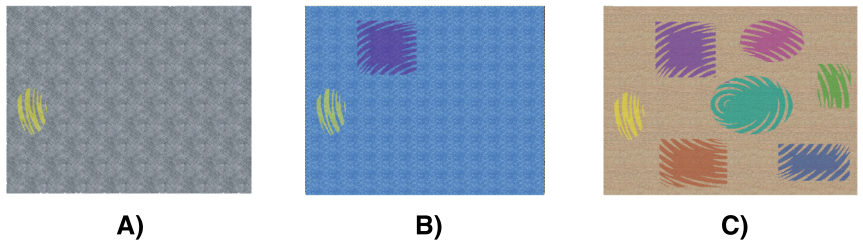


Figure 4 Examples of one of the tasks users completed in the study where they drew transparent shapes and changed the background texture, seven times. These images show their progress after completing the task **A)** once (one shape) **B)** twice (two shapes) and **C)** Seven times (seven shapes).

Data was collected from 44 participants (19 female) whose average age was 25.3 years (SD = 8.37, Min = 19, Max = 59). All but two used Windows as their primary operating system. The majority of these participants reported that they were novice users of image editing and drawing manipulation programs. For example, the average participant used Microsoft Paint between two and three times per month. None had used GIMP before.

In this study, participants completed two multi-step image-manipulation tasks seven times. Participants were given sequential instructions for each task on paper, which they could write on to keep track of their progress. These instructions were formatted so the goal of each step would be clear. In the first task participants drew a new transparent shape and changed the background pattern on the canvas for each trial. In the second task they drew a new letter or shape and colored it with a solid color or gradient (Figure 4).

3.1.1. Collection Software

The collection software used in this study was designed to only use information that is easy to gather in an application independent way (such as the specific width and height of a menu item). Data was collected with a slightly modified version of the GNU Image Manipulation Program, which is built with the Gnome Tool Kit (GTK+ 2.6.10, www.gtk.org). Menu bars and toolboxes were removed so participants could only use the GIMP's popup menus, accessible with a right click. This ensured that users interacted only with menu selections so that data collection would proceed quickly. The GTK was modified to log the appearance and disappearance of all menus and submenus, all menu selections and deselections (when the mouse entered or left a menu item) and all mouse button interactions with the menus.

3.1.2. Data Segmentation

Data was segmented into *recognizable interactions,* or user actions (or portions of user actions) which can be readily isolated, are indicative of the phenomena we wish to study, model or predict, and can be easily and accurately labeled. Recognizable interactions for this study consisted of menu operations (starting from the right click to open a pop-up menu and ending with the left click to select a menu item or dismiss the menu without a selection). While other potential recognizable interactions were considered, menu use was selected because it proved very predictive and it is one of the most ubiquitous interactions in current interfaces.

3.2. Automatically Distinguishing Between Novice and Skilled Use

We labeled the first trial of each task as novice use, and the last was labeled as skilled; the other trials were not used in model building. To test the effectiveness of our classifier we employed a C4.5 decision tree and validated it with a 10 fold cross-validation test with random holdout using Weka (http://www.cs.waikato.ac.nz/~ml/weka/) data mining software. Using this validation measure our classifier achieved an accuracy of 91%, using the following five features selected from the wrapper-based feature selection: average cursor acceleration in the Y axis, total number of menu item visited, depth of the menu item that was selected, ratio between the time taken to select the menu item and the depth of that item, and a count of the unique menu items visited. Of these five features, the ratio of the time taken to make a menu selection and the depth of that menu selection was the most predictive feature, (it had the most impact on classification).

3.3. Closing the Loop: Deploying an Assistive Adaptation

To begin to explore the use of our classifier in real applications we developed a simple adaptive tool that can function across any GTK application. Our tool shows an on-screen window (top of Figure 3) with an expertise-tailored description (obtained from live classification of menu and pointer movement) of the currently highlighted menu item. This description is tailored to the classification provided by our trained statistical model.

Descriptions for novices include examples of what would happen if the menu item were selected and attempts to minimize the use of domain specific knowledge, i.e. information that has been shown to improve novice performance (Dumas and Landauer 1983). Descriptions for users displaying skilled behavior include alternative names for different functions, and also included the keyboard shortcuts for different menu items. For example, a description of the freehand selection tool for a novice said “Use this tool to draw the shape of the area that you want to select” rather than a more complicated description “Lasso: use this tool to make a free-hand selection. Either close a selection by ending in the point you started from, or let the tool automatically close an open shape.” The set of descriptions could be configured by providing a data file with alternate text for each classification associated with each menu item.

We implemented a real-time “live” classifier in Java to parse data about mouse and menu interactions and predict if an action was skilled or novice. Real time data was gathered using the same tools developed for our study. To reduce "jitter", we aggregated the classifier’s predictions using an exponentially decaying average (with a decay factor of 0.5). The aggregated prediction was reported to the adaptive tool, which monitored the currently highlighted menu item and selected the appropriate description of that menu item to display.

A preliminary evaluation of the “live” classifier and its use for adaptation was conducted with 4 participants (2 female, ages = 21, 24, 32, 34) who did not participate in the first round of data collection. To familiarize themselves with the GIMP, participants first completed the same scripted task of drawing shapes and changing the background (the same task used in the data collection study) seven times. Next they worked on a free-form task where they were told to draw a scene. Half of their time on the freeform task was with the live adaptation. During the scripted task, the classifier correctly identified that all participants started as novices and exhibited skilled behavior by the end of the trials. However, the four participants had different learning curves and chose different strategies for the free-form task. In a post study interview, all participants said that they found the example descriptions in the adaptive help system extremely useful. They all responded positively towards the application's awareness of their activities and needs, and had no reservations about automatic evaluation of their performance. However, they expressed concern about having the ability to control the adaptation if it was based on an incorrect classification or if it hindered their activity.

3.4. Lessons Learned

This work was a proof of concept illustrating that it is possible to make accurate assessments with learned statistical models from simple user input trace data, and respond to those assessments in real time. While this was an important step towards the goal of automatically assessing pointing performance during real world use, it did have several limitations that to generalizability.

One of the major limitations of this work was all of our participants were novices. Although this made it easy to label them (based on their learning curve of our task), it did not give us a picture of true expert behavior in GIMP. The GIMP is not a commonly used program (it requires XWindows, which is not commonly used with the most widely deployed operating systems), and this made it difficult to recruit anyone but novices. Another limitation was that collecting data from a scripted task limited the types of errors we saw. Collecting real world data would broaden the pointing actions individuals would do, and would also give a much more realistic dataset.

In the following sections we present our progress towards building software to automatically adapt to pointing problems during real world use. Specifically, our techniques to collect data, segment and analyze it, build and choose features that indicate the presence of pointing problems.

4. Collecting Real World Data

To fully understand and detect the wide range of abilities individuals have and problems they encounter during real world computer use, we need a wide variety of data. Specifically, we need to collect examples of pointing use from individuals with a range of abilities doing tasks that are representative of common real world interactions. One of the limitations from our work distinguishing between novice and skilled actions (Chapter 3) was the lack of variety in the data collected. Specifically, all data was collected from one application from novice users during one session.

Real world data can provide a more accurate picture of an individual's everyday use than performance data collected from a laboratory study (see Section 1.2). Specifically, studying real world use can uncover variances in performance that can be caused by a number of external factors such as fatigue, mood changes, a degenerative disease or a major change in medication.

Figure 5 Real world pointing data was collected from individuals with motor impairments at Pittsburgh's United Cerebral Palsy Center.

Figure 6 Real world pointing data collected from older adults in collaboration with Blueroof Technology. These photos taken during a data collection session at Blueroof Technology.

An effective strategy to collect both short and long-term performance changes is through a long-term deployment to collect pointing performance that captures variance in performance.

In addition to understanding the variance of one person during real world use, it is important to understand the variance across users. The best way to understand variance across groups is to collect examples from many different groups. To accomplish this, we collect pointing data from individuals with and without pointing problems and we collect data from individuals with different kinds of pointing problems. Examples of some of the different pointing problems that we would want to include are limited motion, spastic motion, and tremor.

In this chapter we describe two real world data collection deployments we have performed: one with desktop machines and one with laptops. We present details about our study protocols including recruitment, user tasks, and equipment. We also discuss lessons we learned while running these multi-month studies.

4.1. Data Collection Deployments

We began collecting real world pointing data in early 2008. To date, we have collected hundreds of hours of real world data from 28 people with and without pointing problems (Table 1). Examples of activities they engaged in during collection include web surfing, socializing (email, instant messaging, social networking websites), homework, and video games. This data was collected through desktop and laptop deployments that logged pointing related input events.

Desktop Chapter 5.1	Individuals with Motor Impairments 12 (4 female), 40.5 mean age		
Laptop Chapter 5.2	Individuals with Motor Impairments 4 (2 female) 43.3 mean age 2 were also in desktop study	Older Adults 8 female 67.6 mean age	Able Bodied Adults 4 (1 female) 29.5 mean age

Table 1 Summary of participants in all real world data collection deployments. More details about laptop participants can be found in Appendix 10.3.

4.2. Study Protocols for Desktop and Laptop Deployments

4.2.1. Participant Recruitment

When gathering real world data, it is important to gather a wide range of data from individuals with varying abilities. Two dimensions of diversity we are interested in to assess pointing performance are motor abilities and age range. In our studies we recruit participants with and without motor impairments, age 22 to 80. In order to recruit this range of participants we worked with several different organizations, and used a different recruitment technique for each.

We recruited three groups of participants for our studies including: 1) 14 individuals with motor and cognitive impairments from United Cerebral Palsy Pittsburgh (UCP Pittsburgh, http://www.ucppittsburgh.org), 2) 8 older adults who meet monthly in a group to talk about technology at Blueroof Technology (http://www.bluerooftechnologies.com), and 3) 4 students and staff members from Carnegie Mellon. None of the participants from groups 2 or 3 had any visible cognitive or motor impairments that would affect pointing performance, nor did they indicate any in our pre-test questionnaire. The exclusion criteria for our studies were minimal: participants must be at least 18 years old, and able or willing to use a mouse. Since we had limited equipment, and wanted to collect as much data as possible, we sought participants who we thought would use this computer as their primary machine.

4.2.1.1. Recruiting Individuals with Motor Impairments

Finding individuals with motor impairments who are interested and able to participate in research can be difficult. Since we were interested in maximizing participation from this population, we put a lot of effort into building a relationship with UCP Pittsburgh, a local center which teaches individuals with motor and cognitive impairments a wide range of life skills including computer use. In order to meet these individuals and their instructors, the author spent two half years volunteering weekly in UCP's computer classes to both meet potential participants and to observe this population's existing computer practices.

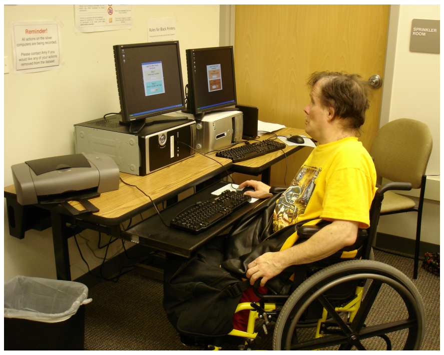


Figure 7 Our desktop deployment was conducted at United Cerebral Palsy (UCP) Pittsburgh. In this deployment we installed two desktop computers in a pre-existing computer lab. Participants were free to use these computers during their classes or free time.

We recruited participants at UCP for both our laptop and desktop deployments. For both studies we met with the director of center services and the curriculum specialist to describe all of the details of our study, and get recommendations for potential participants. We began our first desktop deployment after volunteering at the center for a few months. In addition to helping with recruitment, regularly volunteering at the center where we were running our study gave us an opportunity to observe how the computers were being used and made it easy to quickly answer questions participants had. We installed these desktops in one of the computer rooms at UCP (Figure 7). Participants were allowed to use these machines during classes (with instructor's permission) and during their free time.

Our laptop study began approximately one year after the desktop study. In many ways recruiting for this study was easier because we had a well-established rapport with the instructors and participants. Two of the participants from our earlier desktop study wanted to participate in this study. The rest of the participants were people whom the instructors at UCP thought would be interested in the study or who would benefit from having access to a laptop. One unexpected problem with recruitment for this study was concern instructors had about participants taking equipment out of the center. The instructors wanted to minimize the chance that the participant would be put at a safety risk for carrying around new laptop as they might get stolen. To solve this problem, instructors negotiated with the participants' families and/or care givers if the participant would take the laptops home with them, or only use them at the center.

4.2.1.2. Recruiting Older Adults

We recruited older adults for our laptop deployment through Blueroof Technologies, a partner of the Quality of Life Technology center (qolt.org), a National Science Foundation Engineering Research Center whose mission is to enable older adults and individuals with disabilities to live independently. Similar to our approach at UCP, we did not seek out individuals with specific diagnosis or problems using a pointing device. Instead, we were looking for individuals who could physically use a mouse and we thought would actually use the computer.

All of our participants in the older adult group are termed "Research Associates" at Blueroof Technology, and have known each other for years and all live in the same area of Pittsburgh. The research associates are a group of older adults (mostly women, but some men) who meet in Blueroof's smart cottage once a month for several hours to discuss new technology that has been designed or is currently under development for older adults. In addition to learning about new technology, this group also works together (under guidance of several staff at Blueroof) to learn how to use computers. This group had existed for at least a year before we recruited them as participants for the study.

We attended one of these meetings and discussed the study protocol, and asked anyone interested to fill out a questionnaire about their computer experience (a combination of questions from Appendix 11.1 and 11.2). Fortunately, everyone at that meeting was interested in participating, so we admitted the whole group into our study. While several of our older adult participants already had home computers, none of them had laptops, and they were all excited about having access to a faster computer with Internet.

4.2.1.3. Recruiting Able-Bodied Indivdiuals

Recruitment of able-bodied individuals was the simplest. We did this by posting an advertisement on our Carnegie Mellon's message board that participants could participate in a study and borrow a new laptop for the summer. Our advertisement asked potential participants to answer the same questionnaire (Appendix 11.1) about computer use and experience we asked our older adults for. We received many replies and chose four participants whose answers to our questions convinced us that they would use these laptops as their primary machines.

4.2.2. Deployment Equipment

Desktop Deployment: This deployment was installed in one of the computer labs at UCP Pittsburgh (Figure 7). For this deployment, we installed two DELL desktop computers with the default DELL mice and keyboard, 15" LCD flat panel monitors and color printers. At the time of our deployment, having access to a color printer was attractive to many of our participants. Due to external constraints, our computers differed from those already in the laboratory in that they were not connected to the Internet due to a networking limitation. Our computers had the same productivity tools as the other computers in the laboratory, but different games. Participants could use the study machines as they pleased, however, we did require them to use the USB mouse and keyboard provided by DELL with the machines. These computers were restricted to participants in our study.

Laptop Deployment: We purchased 16 DELL Inspiron 15" widescreen laptops for this deployment. We sent 15 into the field and kept one as a test machine. Participants were allowed to install any software on these computers, but we gave them to them with Windows XP, Microsoft Office, Firefox, and VLC media player pre-installed. Since we were studying mouse use (and the laptops had touch pads on them) we disabled the touch pads on these laptops and gave our participants identical USB optical mice.

4.2.3. Initial Meeting

Since we were studying real world pointing data, we had our participants complete some traditional laboratory tasks during our first meeting. We had them perform these standard tasks so we could have some baseline data, and also to verify that they were able to use a computer and manipulate the mouse. This session took participants about an hour, and they were compensated $25 for their time. During this session participants filled out a questionnaire about demographics and answered questions about their computer experience (Appendix 11.2).

In both deployments we collected laboratory, or baseline performance data using the IDA software suite (http://www.kpronline.com/), developed to support clinical evaluation of problems with keyboard and pointer use (Koester *et al.* 2005). This software is designed for use by clinicians as part of their assessment process for determining assistive technology needs. We had participants perform 30 trials of the AIM task, a standard task included with IDA. In the AIM task, participants were given two minutes to move the cursor to a blue square and click the left button for each trial. The software let them work on the task until they reached the two-minute time limit, and did not alert them if they made errors. IDA automatically varied the size and location of the targets throughout all of the tasks. The mouse gain during the baseline trials was set to the default value (10) and Enhanced Pointer Precision was turned on.

After initial tests, participants in the study were allowed to use the computers (either desktop or laptops) at their own pace and on their own schedule. Participants could use the computers for any task they wanted, and all of the computers had access to the Internet. Participants were required to use the input devices we gave them: USB mouse and keyboard provided by DELL with the desktops, and an external USB mouse for the laptops. Participants were permitted to change the pointing settings (such as mouse acceleration) on the computers; however none chose to do so.

To motivate participants to use our computers, a raffle was held every month for participants who logged into the machines at least once during that time period. Raffle winners received $25, and we did this for both the laptop and desktop deployments.

4.2.4. Data Collection and Storage

There are many possible data storage solutions when collecting large amounts of data. An ideal solution is easy to populate, robust against theft, and failure-safe. Since our dataset was very likely to include sensitive and private information about our participants, we prioritized having a solution that would be robust to intruders or hackers. To this end, all of our data was stored on redundant external hard drives that were encrypted and stored in a secure location. Choosing to store data on these external drives meant that all data collection had to happen in person. While a remote solution that would upload data to a remote server (such as using Dropbox https://www.dropbox.com/) we did not want to risk storing data on another company's server, or worry that uploads could be intercepted.

We met with our participants once a month to collect data from their machines and answer any questions participants had about the computers (and perform system updates if needed). The majority of participants appreciated these frequent meetings to get help installing new software or answer questions about how to use certain programs or websites. During these meetings we would compress and move all logging data off each computer's hard drive and put it on an external drive.

4.3. CRUMBS Collection Software

The CRUMBS (Capture Real-world User Mouse BehaviorS) software was developed to capture real world user actions. CRUMBS is application independent software that logs information about user events in real time. This software extends DART (Disruption and Recovery Tracker) a suite of system monitoring components that run in the background to log pointing, keyboard, and window events (Iqbal and Horvitz 2007). We extended dart to probe the MSAA API for the Role (or type), location and size of the interactor that received a button event. The API returns the Role of interactors it supports and returns "client" for ones that are unsupported.

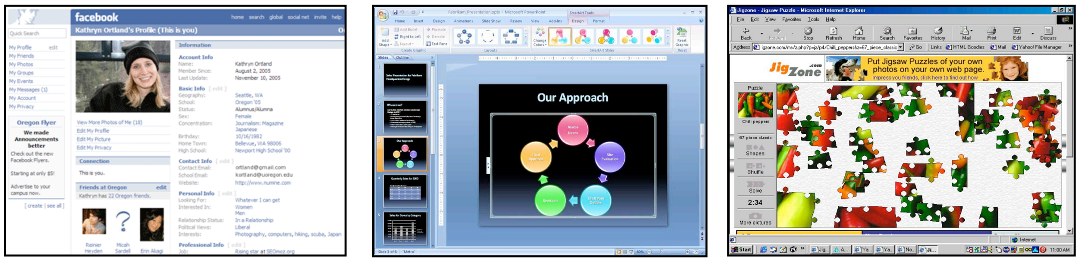
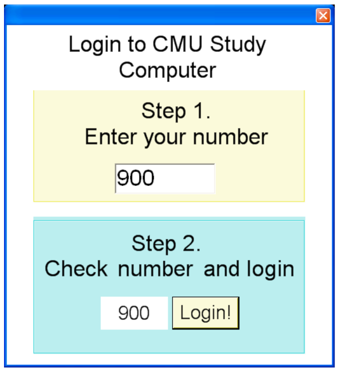

facebook
Kathryn Ortland's Profile (This is you)
Our Approach
Jigzone
Put Jigsaw Puzzles of your own photos on your own web page.

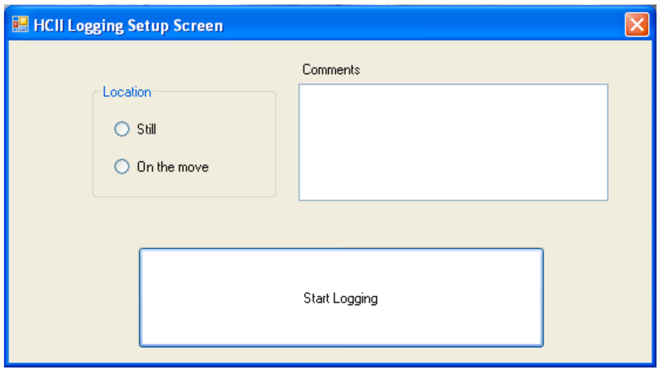
Figure 8 Examples of screenshots captured by our logging software, CRUMBS. CRUMBS is able to capture mouse, windows and keyboard events from any application in Windows XP. It also captures information about targets from the MSAA API (when available) and takes full screen screenshots immediately before all button events.

We also extended DART to capture full screen images immediately after both button press and release events but before the windowing system makes a visual change. These screen captures were called through a hook to the API that receives the button press and release events, and these captures were taken before visual changes are fired due to the button event. This code is written in C++.

We used two slightly different versions of CRUMBS in the desktop and laptop deployments. An early version of CRUMBS was used in the desktop study that only captured 300x300 pixel images (centered on the cursor), and did not record target size and location information. After collecting this data, we added fullscreen image capture and target information from the MSAA API to CRUMBS, and used this improved version in the laptop evaluation.



A)



B)

Figure 9 CRUMBS login screens for desktop (A) and laptop (B) deployments. Participants logged in with a unique number in the desktop deployment because multiple people used the same computer, but in the laptop deployment participants didn't share equipment with anyone else. In laptop deployment participants were asked to provide details about where they were with the laptops and anything unusual (such as not using a mouse pad).

Since CRUMBS logged use in any application, it was very likely to capture sensitive data. To help participants remember that CRUMBS was running and collecting data, a small window appeared at the top of the screen reminding participants CRUMBS was recording with a button that lets them log out. Additionally in the desktop deployment we put up signs near the computers that all data on these computers was being recorded, and participants could ask the experimenter if they would like any of their sessions to be removed from the dataset.

Participants activated CRUMBS by logging into the software. We used slightly different versions of a login screen for our two deployments (Figure 9). In the desktop deployment, participants were assigned unique numbers that were used to identify data and for login purposes. When a user wanted to log in, they would first enter their number, then CRUMBS would verify that it was a valid number (if it were, it would appear in the textbox in the blue region), and then the participant could log in.

For the laptop study, participants were differentiated by the names of the laptops (named after famous composers). Instead of entering a number in the laptop deployment, participants were asked to give additional information about their context such as their location (if they were in a still or moving location such as a train or airplane), and to provide a description of their environment (such as whether or not they were using a mouse pad).

4.4. Lessons Learned by Studying Real World Data with This Protocol

Our deployments were overall successful in achieving our primary goal (gathering a large and realistic data set of real world pointing behavior). Below are a few of the lessons we learned from our experiences.

4.4.1. Tradeoffs Between Using Desktops Vs. Laptops

Our shared-computer desktop protocol worked well for collecting data at UCP, because the computers were placed in a computer lab that was already busy, and gave our participants access to new technology (printers). However, because we did not have a private location for deployment, and because many of the people we wanted to recruit already had a computer, we decided that we could collect more data and recruit from a wider population by studying laptop use instead of desktops in a public lab.

One of our goals with our laptop study was for participants to use these computers as their primary machines, so we could maximize the data we collected. We were optimistic that laptops would also be attractive to participants since they could take them home, to work, or class. During our laptop deployment we saw overwhelming evidence that participants did use these computers as their primary machines and thought of them as their own. The most dramatic response to the laptops was from the older adults. We heard many stories of how these machines changed our participants' lives as they had something exciting to do other than watch television. Additionally, we observed that most of these participants purchased fashionable bags to carry these computers in, and some of them even personalized the outside of these computers. One of our most extreme examples was from one participant who liked her computer so much that she would write little notes to it in the text box on the CRUMBS login screen. This participant would greet the laptop differently most days and write a cheerful sentence about how her day had gone so far. For example: “this is a good day, [the laptop] is here”.

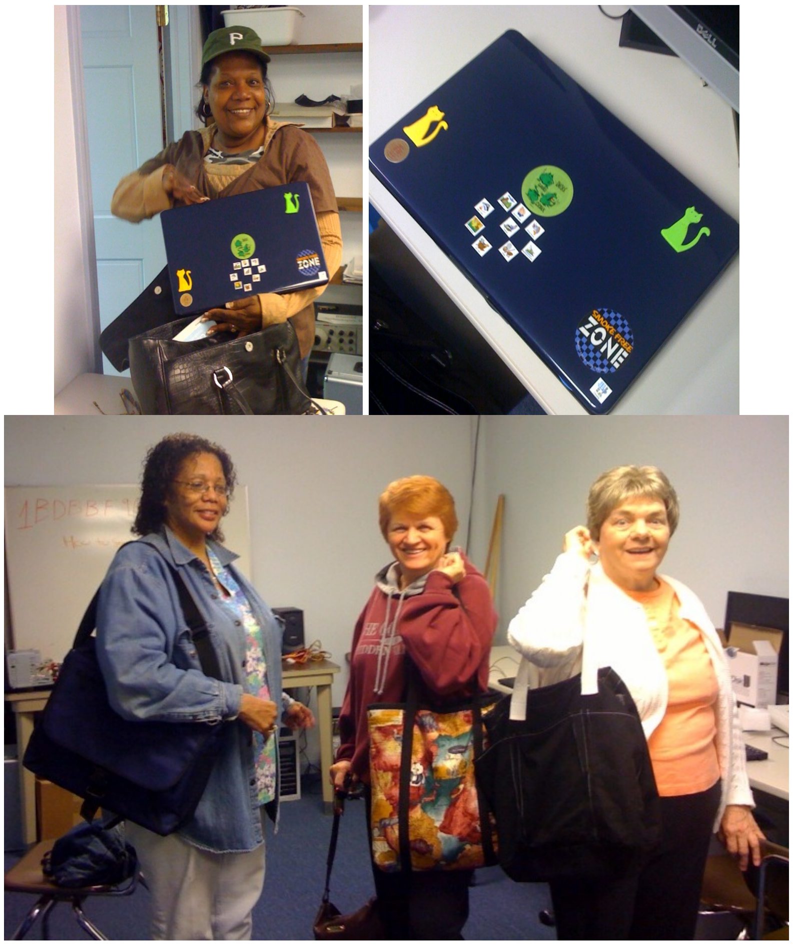


Figure 10 Many of the older adults embraced these laptops as if they were their own, and personalized them with stickers, or comics, and bought fashionable bags for them.

4.4.2. Additional Data Collection Support

We found our desktop data collection model worked better in environments where people were motivated to use computers and there was pre-existing support for them to learn how to use them. At UCP, we had an amazing amount of support from instructors who would encourage participants to complete in-class assignments on our computers, or answer their technical questions.

Unfortunately, not all of our deployments were successful. We deployed a second desktop deployment in the common area in apartment building for older adults. However, we only two participants signed up for the study, and therefore we did not gather as much data from this deployment. We hypothesize that this deployment was not as successful because the participants were computer novices and did not have as much external support as the UCP participants for computer training.

4.4.3. Data Management and CRUMBS problems

One of the difficulties of collecting large amounts of data is the storage and management of terabytes of data. While the log files generated by CRUMBS are relatively small, the full sized screenshots taken during each button press can take a lot of space. Fortunately, storage is currently very affordable, as multiple terabytes of data storage can be easily purchased for a few hundred dollars. However, we did spend more time than we initially expected moving and compressing data from the laptops onto our external data storage drives.

While we designed CRUMBS to easily give participants full control over recording, some participants would not log out of CRUMBS when they finished using the computer. This was problematic for purposes of analysis because these files would be extremely long, and contain performance data from multiple sessions of use. Most of the participants from the UCP group always turned off the computer when they were finished (thus logging themselves out). However, several participants in both the older adult and able groups closed the laptop instead of logging themselves out. In the future this can be fixed by automatically logging users out after being away from the computer for a significant time or if a long time has elapsed since the last event we recorded.

5. Real World Pointing Performance

Little is known about pointing performance in the real world. Performance metrics traditionally used to study pointing performance collected in a laboratory can also help us to better understand user ability and natural tendencies during real world use. This chapter discusses performance across the pointing data we gathered from our laptop and desktop deployments with a high level description of the data we collected, segmentation technique we used, and some key targeting and pointing performance metrics. Together, the analyses presented in this chapter illustrate the real world properties of movement and targeting performance during real world tasks.

Our desktop deployment was a pilot for the laptop deployment. For the desktop deployment, we set up two computers at United Cerebral Palsy in Pittsburgh and gathered data from 16 motor impaired individuals over nine months. In this pilot, showed that we could collect and analyze real world pointing data from individuals with motor impairments. We developed an improved collection method and analysis based on it (summarized in Section 4.2). The analysis of data from our desktop deployment highlights the dramatic range of real world pointing performance for individuals with pointing problems.

In our laptop deployment we recruited a wider range of participants (older adults, individuals with motor impairments, and able bodied adults) and loaned them laptops for multiple months. We also used an improved version of CRUMBS that logged target size and location when clicked on, and a segmentation technique that used pause information. We found that our laptop protocol enabled us to collect a large amount of data, from a wide variety of applications. The analysis of the data from our laptop deployment illustrates the differences in targeting and movement performance across our three groups of participants.

5.1. Targeting in Desktop Deployment: Pointing Measures and Performance from Indviduals with Pointing Problems

In this section we present clicking and pointer motion problems that we assessed from our desktop deployment collecting real world use from individuals with motor impairments. These problems are drawn from, the list of errors described in Section 2.3. To understand actual pointing problems we looked at performance from individuals with pointing problems who used the computers multiple times during a four month period. We collected this data using the study protocol presented in Section 4.1, from participants using shared computers at United Cerebral Palsy Pittsburgh. Data was collected with our CRUMBS software. This data includes use interacting with desktop icons, playing one of two clicking games, and using Microsoft Word (Hurst *et al.* 2008B*)*.

5.1.1. Distribution of Data

Using our desktop protocol, we collected real world data from 11 participants starting in January 2008, and data collection is still ongoing (as of publication date). In the following subsections, we present a subset of our total dataset from 6 participants (4 female) who used the computers at least twice in a four-month period from 1/08 to 5/08. The mean age of our participants was 40.5 years (SD = 7.77) and 2 of them used a mouse with their left hand. Activity and frequency of use varied widely across participants, but ranged from 2 sessions to 120 sessions of length 2 minutes to 74 minutes. For participants who logged many sessions, we randomly selected 9 sessions (and relabeled them 1-9) for this analysis.

5.1.1.1. Applications

Our participants used these computers for a variety of tasks including playing clicking games and card games, solving jigsaw puzzles, making greeting cards and typing documents. Overall, the majority of the data we collected was from clicking games, specifically a matching game called the “Same Game”, and a memory game called “Memory Blocks”. While Microsoft Word was the third most popular application, we collected eight times more data from the two clicking games than Microsoft Word. Participants P2 and P6 did not use the computers as frequently as the other participants and did not play either popular clicking game; however they did interact with the desktop and Microsoft Word.

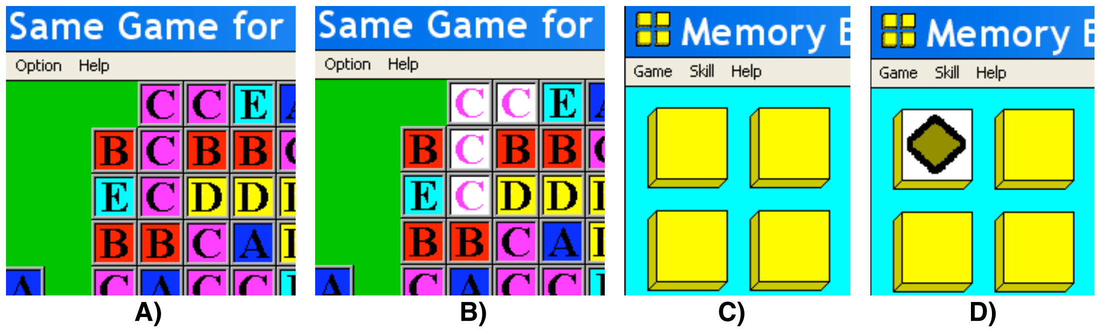


Figure 11 Screenshots from the two most popular games A & B) Same Game, where participants need to click on a connected group of blocks with matching colors and letters. B shows the currently selection of connected blocks. If user clicks on selected area, all highlighted blocks will disappear. C & D) Screenshots of the Memory Blocks Game where participants click on blocks to flip them over and match blocks with identical icons. This is a memory game because tiles are flipped back over after two have been viewed, so participants must remember the location of the icons.

5.1.1.2. Segmentation of Pointing Data

One of the disadvantages to studying real world data is it is less straightforward to analyze than data collected in the laboratory. Before we can detect problems in performance, we must put the data into a form that we can easily analyze. The biggest obstacle to doing this is making sense of enormous streams of real world computer use collected by CRUMBS. Specifically, tools are needed to segment long event streams into useful samples that represent single actions (such as selecting a target). Since we collected data during real world computer use, we did not have a task model for the user's actions, nor did we know what tasks the user intend to perform.

CRUMBS data is organized into "sessions" which start when the participant logs in and end when they log out. For this analysis, we divided data into samples by button events. Specifically, these samples describe everything that happens between two button up events. We used this segmentation technique because it was straightforward, has been used in laboratory evaluations to segment data, and we thought it would best match the data. This dataset contained 11,811 samples.

5.1.2. Targeting Performance Errors

In this dataset we looked at two targeting, or clicking metrics that are target agnostic, or do not require the size and width of the target for calculation. All of the performance metrics presented here have to do with types of errors that may occur during targeting.

5.1.2.1. Pressed Too Many Buttons

Some participants have difficulty pressing only one button on a mouse, and instead press both the left and right mouse buttons. Pressing both buttons is an error because none of the applications that our participants chose to use handle this event. This is an interesting feature to analyze because it is easy to detect through log file analysis and may be an easy way to detect pointing problems.

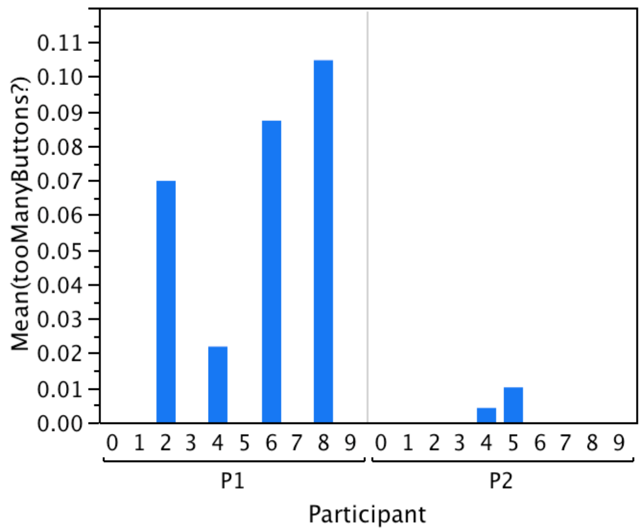


Figure 12 Plot of number of clicks with too many buttons across all logins and sessions. Login 0 is from the baseline task.

Of our six participants, only P1 and P2 had overlapping button presses from pressing multiple buttons during a click. Figure 12 illustrates the frequency of this performance problem across 9 login sessions for all applications. In this figure, P1 only exhibited this problem during sessions 2,4,6 and 8. Participant P2 had three overlapping button presses, which happened while using Microsoft Word during sessions 4 and 5.

5.1.2.2. Slipping During a Button Press

We define a *slip* as any distance a pointing device moves while a pointer button is pressed. When analyzing slipping performance in real world data, we have to differentiate between drags (intentionally holding down the mouse button and moving the mouse to manipulate an object) and slips (unintentionally moving the mouse while the mouse button is pressed). One simple technique to do this is to declare a cutoff distance between slips and drags. This technique is used by Trewin in Steady Clicks (Trewin *et al.* 2006), where they use a cutoff of 100 pixels to distinguish between drags and slips.

While this technique may work in many cases, it misses all short drags and classifies them as slips. Another approach is to look at the type of the interactor clicked on and decide if it is draggable or not. Of course, this technique only works on targets that are supported by Accessibility APIs. For this data, we only analyze applications that do not include drags. To this end, we also hand coded and removed all drags due to menu use. For each click, we calculate the "slip" as the Euclidean distance between button up and down events.

The mean slip distance ranged from 0 pixels to over 20 pixels across all users, the mean slip distance for all participants across all samples was 5.9 pixels (SD = 19.45). Participants P4 and P5 had the most problem with slips {P4 mean = 21.34 pixels, SD =43.96; P5 mean = 10.25 pixels, SD = 24.49}. Large slips can cause a click to fail because the pointer moves off the selected interactor before it is released.

5.1.3. Movement Performance Metrics

In addition to features that describe clicking performance, we evaluate pointer motion. In this section we describe two target agnostic performance metrics we use to evaluate pointer motion.

5.1.3.1. Changes in Direction

Mackenzie defines Movement Direction Change (MDC) as the number of times the pointer path changes direction relative to the task axis and Orthogonal Direction Change (ODC) as the number of times the change in pointer path is orthogonal to the task axis (Mackenzie *et al.* 2001). For our analysis, rather than calculating the task axis and the amount of movement direction change/orthogonal direction change, we focused on the simpler measures of number of X direction changes and Y direction changes from one click to the next.

Figure 13 illustrates the average number of direction changes per motion along the X and Y-axis. The 0th session shown for each participant is the baseline session, and the following are from real world use. In this figure, there is data from 10 sessions from P1, 1 from P2, and so on. Participants P1, P4 and P5 experienced high numbers of direction changes most frequently, with means ranging from more than 30 to 0 across different sessions. Total scores for these individuals across all sessions for X direction changes were {P1 Mean = 4.8, SD = 8.62; P4 Mean = 2.8, SD = 7.45; P5 Mean = 3.17, SD = 5.14} and Y direction change were {P1 Mean = 2.14, SD = 3.13; P4 Mean = 3.11, SD = 7.51; P5 Mean = 3.63, SD = 7.47}. This analysis highlights that P1 had more direction changes in the X direction (with a large SD) than in the Y, and there was not much difference between X and Y direction changes for participants P4 and P5. P1's X and Y frequency direction changes may be due to this participant's intermittent tremor.

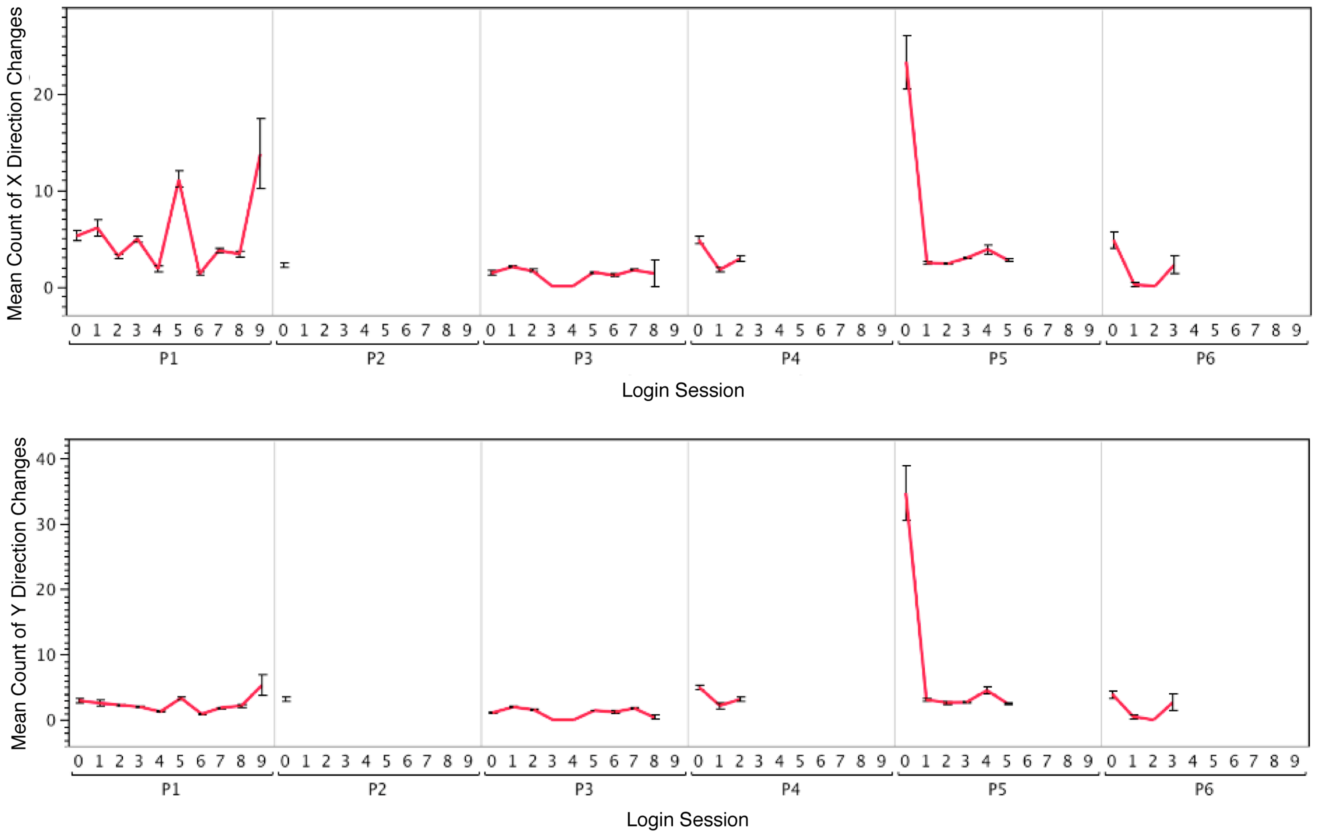


Figure 13 Comparison of the difference between the mean number of direction changes in the X (top) and Y (bottom direction by each user across login sessions. Login 0 was from a baseline clicking task, all other data is from double clicking a desktop icon or one of the clicking games.

5.1.3.2. Excess Distance Traveled

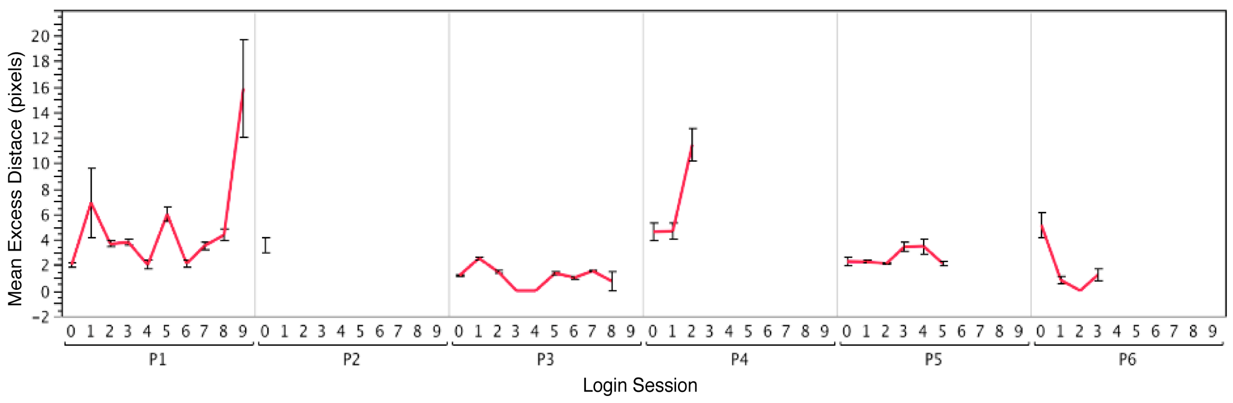


Figure 14 Comparison of the mean excess distance traveled by each user across login sessions. Login 0 was from a baseline clicking task, all other data is from double clicking a desktop icon or a clicking game.

Keates defines the excess distance traveled as the ratio of the actual distance traveled relative to the minimum straight line distance to a target (Keates *et al.* 2002A). The actual distance traveled becomes higher when the cursor wanders. We calculate the ratio of the total distance traveled and the minimal Euclidean distance between each pair of clicks. This ratio represents the efficiency of the trajectory; it is 1 when the cursor moved in a straight line from the source point to the target, and larger otherwise. One limitation to analyzing cursor trajectories in real world data is that we cannot determine if the path data is a long or curvy movement made because the user was having a pointing problem or rather they changed their mind during the movement and selected a different target than their initial intent.

Figure 14 shows how the average excess distance traveled varied unpredictably between different sessions for some participants in our study. When this value is close to 1, the user moved in a straight line from the source to the target. P1's mean excess distance traveled ratio is low (below 2.02-2.11) in the baseline session and real world sessions 4 and 6. However P1's performance on this metric rises as high as 6.9 in session 1 and 15.86 in session 9.

5.1.4. Summary: Desktop Performance

This desktop deployment proved that we could calculate useful pointing performance measures from individuals with motor impairments while they performed real world tasks. However, open questions remain about how real world pointing performance differs across a wider variety of individuals, and with a wider variety of applications (including Internet use). Additionally, our desktop data lacks information about the size of the target users clicked on, making it impossible to analyze target size affected performance in real world use. The laptop deployment addresses these issues.

5.2. Targeting in Laptop Deployment: Pointing Measures and Performance from Indviduals with Pointing Problems, Older Adults, and Able-bodied users

Using our laptop protocol we collected over 360,000 samples of real world targeting motions from 16 (11 female) participants over a 9-month time period. As in section 5.1, these samples contain keyboard, mouse and window events that happen between press events. Demographic information about our participants is summarized in our appendix (Section 11.3). Performance from our participants is organized according to three groups: able bodied (4 participants), older adults (8 participants), and individuals with motor impairments (4 participants). As mentioned in our study protocol, participant actions were recorded at the participant's discretion and they were free to use the laptops when and how they chose. We collected roughly 83,000 samples from able-bodied individuals, 251,000 from older adults, and 29,000 from individuals with motor impairments. Note that a refined segmentation approach for samples is introduced and used below, however, this does not dramatically change the number or distribution of samples.

The large volume of data we collected allowed us to look at subsets of data, from which we could more unambiguously infer intent. We did this by subdividing data by the application the participant was using, and then selecting specific applications that were used by all or most of our participants.

We determined what application the user was interacting with by analyzing the titles of the windows they were interacting with. This process helped us identify the applications used in 90% of the samples. The remaining 10% of samples were not assigned to any application. The most frequent applications used in our data across all participants were Firefox (23% of all data), Spider Solitaire (20%), and Mahjongg (19%) and FreeCell

(14%). In the able bodied group Firefox was by far the most common application (accounting for 75% of their data) other popular applications were File Explorer (1.7%) and Microsoft Word (1.7%). Games were extremely popular with our older adults (at least 82% of their data) specifically Mahjongg (27%), Spider Solitaire (25%), and FreeCell (18%). Internet use accounted for only 8% of their data. Spider Solitaire was the most popular application with the participants with motor impairments 38%, Solitaire and FreeCell were the next most popular (both at 18%), and Internet accounted for only 12%. In all, games accounted for 64% of all the data we collected in this deployment.

Underrepresented applications in our dataset (which we expected to be more generally popular, based on our own computer use) include Instant Messaging (0.3%), Microsoft Outlook Email Client (< .01%) and Microsoft Outlook Calendar (< .01%). The lack of use of these applications could have been caused in part by our study design. Participant may have chosen not to set up and use these applications because they were borrowing the laptops and their information would be recorded locally. While we did not see much Instant Messaging or Calendar use, we did see a lot of webmail usage in all three groups.

Because many of the aspects of pointing motion that we analyzed depend on target size, we also looked at what target each sample ended on. We used the MSAA API to obtain the width and height of the interactors users clicked on. As mentioned earlier, the MSAA API is far from perfect at recognizing all targets, but it is a fast way to get this information. The MSAA API was able to recognize about 25% of all targets clicked on in our dataset (the MSAA API returns nothing, "(null)", or "client" for targets it does not recognize). Approximately 60% of the unsupported targets in this dataset came from games that older adults played: such as Spider Solitaire, Solitaire, FreeCell and Mahjong.

For our analysis, we created three subsets of data from Microsoft Office (MS Office), Internet, and game use. All of our analyses use one or more of those subsets. The rest of this paragraph describes the distribution of applications across these three subsets. The MS Office subset included use from Microsoft Word (.6%), Excel (.09%) and PowerPoint (.01%). The Internet subset was data from both Mozilla Firefox (26.34%)

and Internet Explorer (.02%). We obtained data from all three groups during MS Office and Internet use. However, we only observed instances of game use from older adults and individuals with motor impairments. The games they played included Spider Solitaire (23.1%), Mahjong (21.3%), FreeCell (15.8%), Scrabble (8.6%), Solitaire (3.2%), Minesweeper (0.4%), Wheel of Fortune (.01%), and Pinball (.01%). We include pointing performance from games for features that do not require target size information because they represent a large portion of the data, and we see performance differences between games and other applications for older adults and motor impaired individuals.

5.2.1.1. Segmentation of Pointing Data

We used a more sophisticated segmentation technique for our real world laptop data than we used to analyze our desktop data (Section 5.1). In addition to segmenting by button events, we divide each sample into segments at each pause of greater than 200 ms. The last segment (with at least 10 pixels of movement) before the end of the sample is kept for analysis and the others are discarded. If there are no pauses, or less than 10 pixels of movement, the entire sample is used. The earlier segments are presumed to be non-targeting motions since they do not end in a click. For example, the user may move the mouse along words while reading, pause, then move it over a link and click.

This technique is based on prior work by Chapuis on real world pointing (Chapuis *et al. 2007*), as well as our own investigation of segmentation using 200ms, 300ms and 500ms cutoffs. Confirming Chapuis' findings, we found that longer cutoffs for able bodied users incorrectly included more samples that were non-targeting motions such as movements to a target and back (possibly to check a value) and moving the cursor to refresh the screen. The best single segmentation threshold for our older adult and motor impaired participants was less clear. The 200ms cutoff worked for most of them, however a few participants in these groups may have needed a longer cutoff. In order to determine the optimal segmentation cutoffs, (and determine if these should be individual or group based) we feel that much more investigation is required. Since the 200ms cutoff was no worse than the other options across all users, we kept it.

5.2.1.2. Outliers

We performed an outlier analysis of our data that looked at several timing features, window sizes and API target sizes. This outlier analysis was based on entire samples (click to click), instead of the last segment of a sample. Samples with more than 10 minutes of pauses were considered outliers and were removed. This removed 485 samples of our data. These long timing events were created when participants were not actively using the mouse, which happened most frequently while participants were watching videos or while on the web. Window dimension and target size features were capped at 2,000 pixels (which is larger than the maximum window resolution of these laptops). Our outlier analysis removed less than 1% of the data.

5.2.2. Distribution of Data Across Independent and Dependent Variables

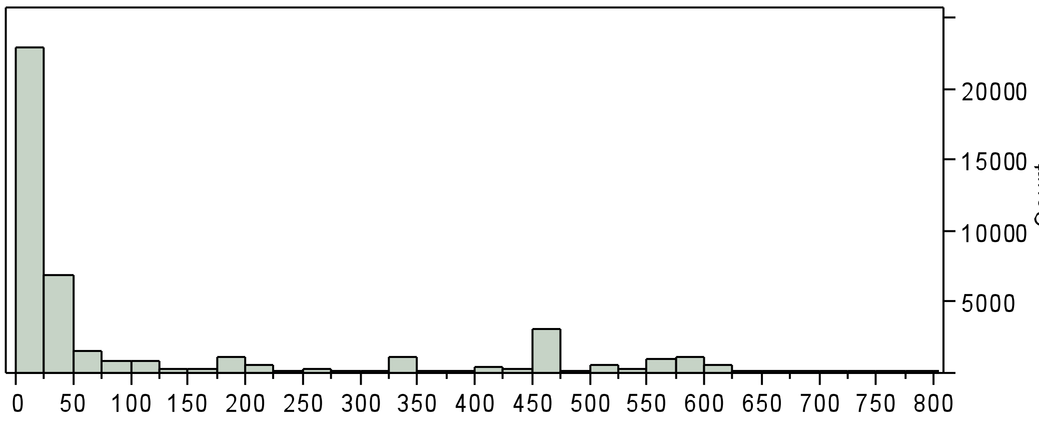
Once we had selected which subsets of the data we would use for analysis, we examined several independent variables to better understand the distribution of the data. These include the target sizes that participants interacted with, and the index of difficult (target size combined with distance to target). We also looked at a set of dependent variables, including motion metrics and targeting metrics. Below we briefly summarize both sets of variables before jumping into a detailed analysis of the dependent variables.

5.2.2.1. Distribution of Target Sizes

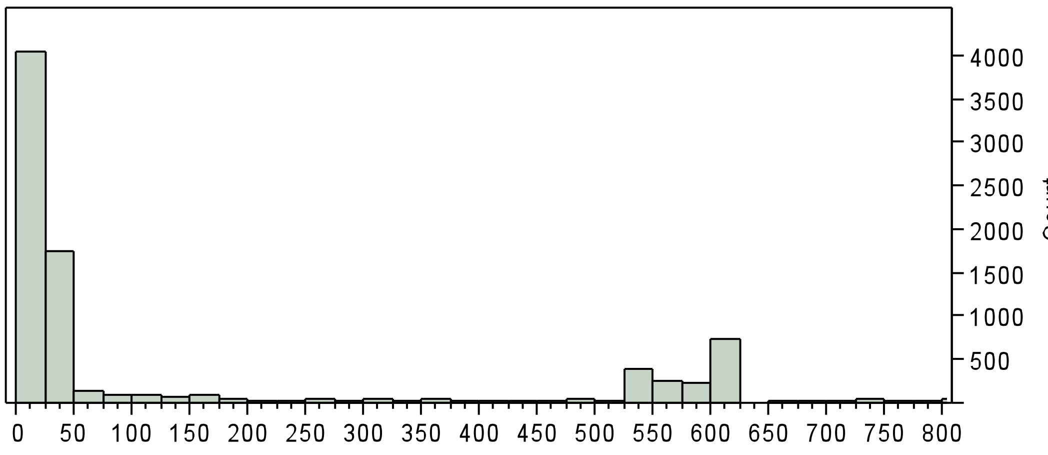
We analyze target sizes by using the smaller of the width or height of the selected target, a frequently used technique in Fitts' Law analysis (MacKenzie and Buxton 1992). Table 2 and Figure 15 (below) show the distribution of targets selected across our three groups both across Internet and MS Office use. We include all of the interactor types returned by the MSAA API in our analysis except for "client" and "(null)" types (values this API returns when it does not know the target size).

	Mean	Std Dev	Median
Able	114.17	177.71	24
Older Adult	144.53	225.54	24
Motor Impaired	111.00	188.21	26

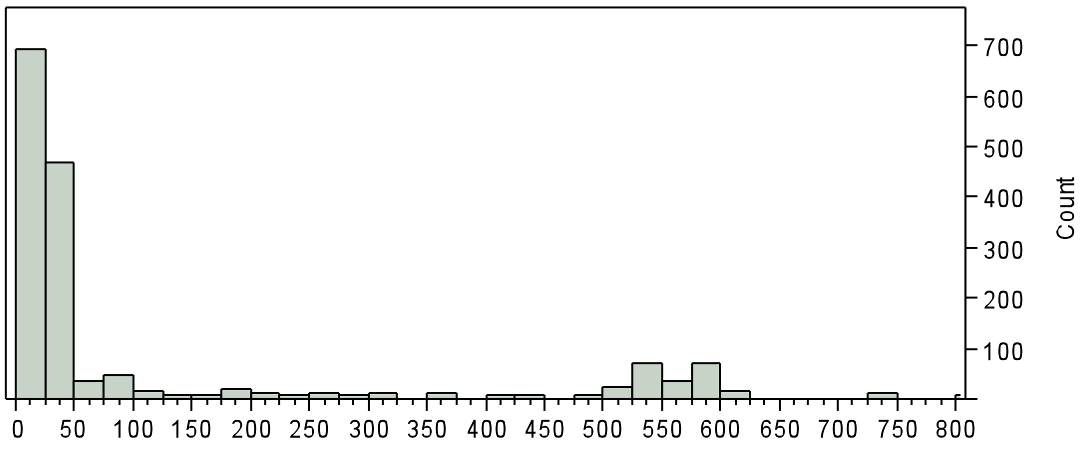
Table 2 Distribution of target sizes across all three groups for Internet and MS Office use. This data excludes game use since targets in games were rarely detected by MSAA API.



A) Target sizes selected by able bodied participants



B) Target sizes selected by older adults



C) Target sizes selected by participants with motor impairments

Figure 15 Selected target sizes (in pixels) reported by the MSAA API for our three groups for Internet and MS Office Use A) able bodied B) older adults, C) individuals with motor impairments. This reported target size is the smaller of either the height or width of the target. Note that the Y-axis is different for each group because the sample sizes are different for each group.

Note that older adults and motor impaired users preferentially used applications for which target sizes were not available in the MSA API (games, in particular). The extremely large target sizes in Figure 15 were from fullscreen images on a webpage, clicking on whitespace on a webpage, or the “list” widget (a widget that contains a scrollable list of options, such as a font list). For the “list” interactor type, the MSAA API returns the size of the container (bounding box of list), not the smallest targets (list items). Overall for some of these target types it could be argued that the actual object being selected by the user is actually a smaller subcomponent of the object being reported by the API (e.g., as in the case of lists). This represents a source of noise in our data.

5.2.2.2. Index of Difficulty

We can use the Shannon formulation of Index of Difficulty (ID) as a measure of how difficult a movement to a target was (MacKenzie and Buxton 1992). ID is calculated using the Euclidean Distance to a target and the smaller dimension of the target (either width or height) (Equation 1). Smaller IDs represent easier movements either because the target is large or nearby. Likewise, larger IDs represent targets that are harder to reach because they are either far away or small. Figure 16 shows the histograms of Index of Difficulty across our three groups (for both Internet and MS Office use) and across Internet and MS Office use. We see a trend across all participants that the most common ID was 1 (after binning to the nearest integer). However, the frequency of the other Index of Difficulties varies across groups.

$$ID = \mathrm{Log}_2\left[\frac{EuclideanDistanceToTarget}{Width} + 1\right]$$

Equation 1 Index of Difficulty is a measure of the distance moved and the target size

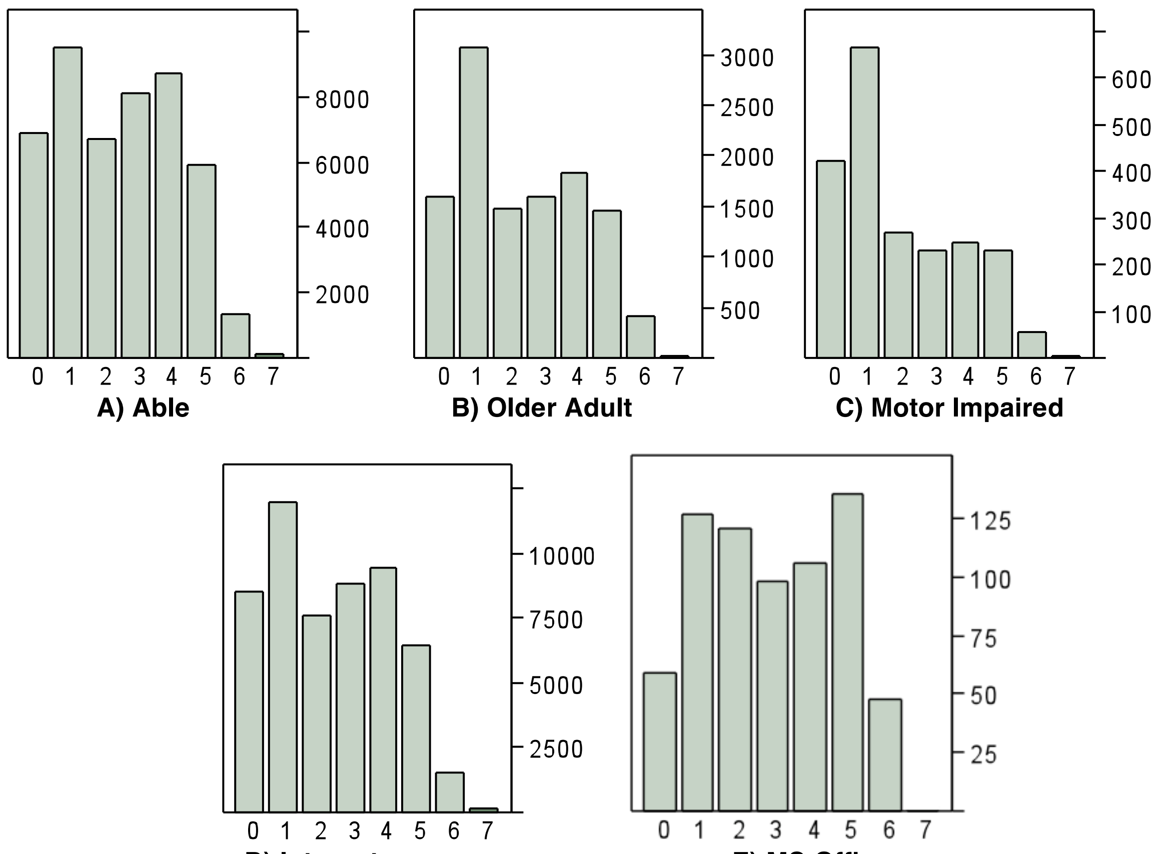


Figure 16 Distribution of Index of Difficulty across group (A-C) and application (D and E). We do not report the Index of difficulty for game data because so few of the targets were supported. Note that the Y-axis is different for each group because the sample sizes are different for each group. Also, there were zero targets at ID = 7 in MS Office.

	Index of Difficulty			
Variable	**Estimate**	**Std Error**	**Prob>ltl**	
Intercept	2.2888	0.1153	<.0001	***
Group: Older Adult	0.0105	0.1503	0.9454	
Group: Motor Impaired	-0.4169	0.1713	0.0315	*
	R^2 = .04			

Table 3: Model of Index of Difficulty by group: older adult, motor impaired, able bodied (omitted). *p < .05, **p < .01, ***p < .001.

We investigated whether ID varied across group by building a hierarchical linear model, treating participant as a random effect (this analysis method is explained in Section 5.2.2.4). Table 3 presents the results of this model, which illustrates that targets selected by able bodied participants were significantly more difficult (had higher IDs) than those selected by motor impaired participants (p < .05). There was no significant difference in the difficulty of targets selected by older adults and able bodied participants (p = .945).

Computer expertise is likely one cause (more experienced users may know about the function of smaller, and hence more obscure targets) for this difference. It is also possible that able-bodied participants were more confident in their ability to select these targets, and were more willing to select them while the other participants found another way to perform a task. If so, application developers may want to design interfaces with larger targets for older adults and individuals with motor impairments.

5.2.2.3. Relationships Between Metrics

We calculated the following raw performance metrics from the segmented samples in our Internet, Microsoft Office (MS Office), and games data sets:

1. Too many buttons (were two buttons pressed simultaneously during the button down to button up period?)
2. Click duration (time from button down to button up in milliseconds)
3. Distance slipped (distance traveled between button down and up event in pixels)
4. Movement time (from start of segment to click)
5. Total distance (distance traveled by the cursor in pixels)
6. Euclidean distance (distance between the start position of the cursor and the location of the button down event in pixels)
7. Time to peak velocity (time passed before peak velocity is reached in milliseconds)
8. Direction changes in X and Y direction (count of direction changes in movement in both X and Y direction)

To understand the relationship between these eight performance variables, we conducted a multivariate principal component analysis. Three groups of correlated features arose from this analysis: clicking measures (metrics 1 and 2), slipping (metric 3), and moving (metrics 4-8). We will describe our real world pointing performance using these metrics groupings. Section 5.2.3 focuses on “targeting metrics” which includes these click metrics and the distance slipped. Section 5.2.4 presents “movement metrics” that describe how the cursor moved between targets. This section uses combined metrics such as velocity, throughput, and efficiency instead of raw metrics such as time and distance.

5.2.2.4. Data Analysis Method

In the following subsections we discuss three targeting and five movement features and how their performance varied across application and group, and difficulty of the targeting task (measured by Index of Difficulty). We investigate targeting performance by analyzing the frequency of pressing too many buttons, duration of clicks, and distance slipped during a button press. We investigate movement by looking at the number of direction changes (in both the X and Y directions), efficiency of movement to a target, velocity, and peak velocity.

We analyzed our data using multiple linear regression, a method which lets us investigate the effect of multiple independent variables (participant, group, application type, and difficulty of movement) on a given targeting or movement metric. We include participant in these models as a random variable to account for non-independence of participant observations (we have much more data from some participants than others).

We present the results for each metric with one or two models, depending on the complexity of interactions we are investigating. In each, we include the coefficient of determination, R^2, which is the amount of variability explained by the model, and a measure of model fit. We include both *main effects* (attributable to one dependent variable, holding all others constant) and *interactions* (between two dependent variables). In this document, we do not present the results of all of the steps in the regressions we did, but instead report only the significant ones. In general, our simple model reports the main effects of all dependant variables, and an interaction between group (able bodied, older adult, or motor impaired) and application type (Internet, MS Office, or game). In our more complex models, we include an interaction between difficulty, (measured as either ID or distance moved and size of target), with group.

Categorical variables (such as group and application type) have been transformed into standard dummy variables, such that there are N-1 binary variables representing the N categories. For example, the variable Group has three categories: Able-bodied, Older Adult, and Motor Impaired. It is represented as two binary variables, Older Adult and

Motor-Impaired, and the third, “omitted” category, Able-bodied, is represented by setting the two dummy variables to zero.

In the models below, the intercept represents the estimated value of the model when all continuous independent variables have a value of zero, and all categorical values are set to the omitted category (e.g., the intercept in Table 4 represents the estimated click duration of an able bodied adult using MS Office). For most of our models these are the omitted values, but we specify the omitted values for each regression.

The columns in these tables include the *estimate*, standard error, and probability that this was a significant finding. The estimate for an independent variable is the amount added or subtracted from the intercept every time the independent variable is increased by one unit (for continuous variables) or is equal to 1 (for dummy variables).

We take the logarithm (base 10, after adding a start value of 0.001) to features that roughly follow a power law distribution to control for skew. We indicate which metrics were normalized this way in both the regression table and in our description of our results. Note that in order to calculate the estimated values for these normalized variables, one must use the inverse log.

5.2.3. Targeting Performance Metrics

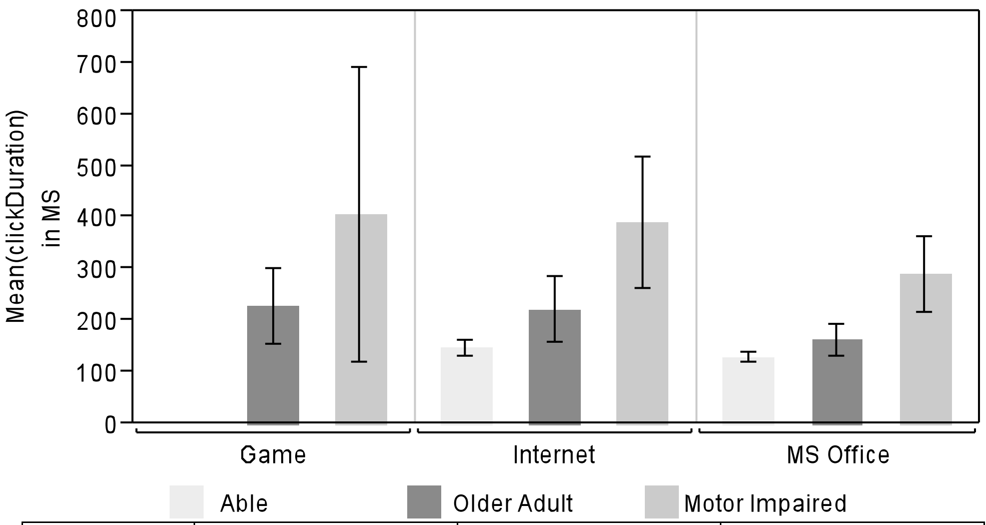
In this section we discuss targeting performance metrics. These metrics evaluate how long users took to click on targets, how frequently they pressed too many buttons on the mouse, and slipped during a click. Our principal component analysis (Section 5.2.2.3) found click duration and pressing too many buttons to be in the same factor, and the distance slipped to be in its own. The following section will show how click durations varied across groups, and how three groups experienced errors slipping off targets

5.2.3.1. Click Duration

Metric description and calculation: We calculate click duration as the amount of time (in milliseconds) the mouse button was held down during a click.

Divisions of data: In our base statistics we compared click duration across Internet, MS Office and game use. In our more detailed regression analysis, due to the lack of game data from able bodied participants we only looked at this metric for Internet and MS Office use across our three groups. We also performed a second regression analysis to see how game data from older adults and individuals with motor impairments was different from MS Office and Internet.

Since click duration is naturally longer during drags, we only looked at click duration where the mouse did not move during the button press.



	Game		Internet		MS Office	
	Mean	**Std Dev**	**Mean**	**Std Dev**	**Mean**	**Std Dev**
Able			147.236	15.912	128.526	10.1733
Older Adult	226.845	72.801	221.076	63.239	163.282	30.4858
Motor Impaired	406.756	285.245	391.059	128.410	290.252	74.4451

Figure 17 Mean and standard deviation of click duration in milliseconds organized by group and application. Able bodied participants had the shortest click duration, and individuals with motor impairments had the longest.

	Click Duration			
Variable	**Estimate**	**Std Error**	**Prob>ltl**	
Intercept	2.2786	0.0321	<.0001	***
Group: Older Adult	-0.0117	0.0417	0.7845	
Group: Motor Impaired	0.1822	0.0473	0.0023	**
Application: Internet	0.0050	0.0043	0.2468	
Older Adult X Internet	0.0116	0.0064	0.0689	
Motor impaired X Internet	-0.0260	0.0070	0.0003	***
	R^2 = .21			

Table 4: This model presents estimated click duration (log base 10 of milliseconds) by group: older adult, motor impaired, able bodied (omitted), and application type: Internet or MS Office (omitted). *p < .05, **p < .01, ***p < .001.

Regression results for click duration during Internet and MS Office use: We created a hierarchical linear model treating participant as a random effect. Table 4 presents the results of a model of click duration. This model presents the main effects and interactions between user group (able bodied adults, motor impaired adults, and older adults) and application type (MS Office, Internet) (R^2 = .21).

We can use this regression table to calculate the estimated click duration for any group of participants using either application type. We can calculate the estimated click duration for motor impaired participants using MS Office (the omitted, or default application category) as the intercept (2.2786) plus the estimated value for motor impaired participants (0.1822) which equals 2.4608. We use the logarithm (base 10) in this metric, so estimated click durations are calculated with the inverse log (10^2.4608) = 288.94ms (.289 seconds). Likewise, the estimated click duration for motor impaired participants during Internet use is the intercept (2.2786) plus the estimated value for motor impaired participants (0.1822), plus the estimated value for internet use (0.005), which equals 2.4658 or 10^2.4658 = 292.28ms (.292 seconds). The estimated click duration for able bodied participants using MS office is the intercept 2.2786 or 10^2.2786 = 189.93ms (.190 seconds), and the estimated click duration for this group using the Internet is the intercept (2.2786) plus the estimated value for internet use (0.005) which equals 2.2836 or 10^2.2836 = 192.13ms (.192 seconds).

This model illustrates the following trends:

- Across all applications, motor impaired participants had significantly longer click durations than able bodied users ($p < .01$) across all application types, but there was no difference between older adults and able bodied users ($p = .785$).
- There was no main effect for application type on click duration ($p = .247$).
    - There was an interaction effect between able bodied and motor impaired participants during Internet use ($p < .001$). Able bodied participants had significantly shorter estimated click durations (10^(2.2786+0.005-0.026) = 180.97 ms (.181 seconds) than participants with motor impairments (10^(2.2786+0.005+.1822-0.026) = 275.30 ms (.275 seconds).

Regression results for click duration during Internet, MS Office, and game use: To understand how click duration varies during Internet, MS Office, and game use we created a second hierarchical linear model (that also treats participant as a random effect). Table 5 presents the results of the model of click duration. This model presents the main effects between user group (older adults and motor impaired adults (omitted category)) and application type (MS Office, Internet, and games (omitted). ($R^2 = .54$).

	Click Duration			
Variable	**Estimate**	**Std Error**	**Prob>ltl**	
Intercept	2.3885	0.0520	<.0001	***
Group: Older Adult	-0.0756	0.0520	0.1771	
Application MS Office	0.0241	0.0075	0.0014	**
Application: Internet	0.0108	0.0040	0.0072	**
Older Adult X MS Office	-0.0253	0.0075	0.0007	***
Older Adult X Internet	0.0199	0.0040	<.0001	***
	R^2 = .54			

Table 5: This model presents estimated click duration (log base 10 of milliseconds) by group: older adult, motor impaired (omitted), and application type: Internet, MS Office, and games (omitted). *p < .05, **p < .01, ***p < .001.

This model illustrates the following trends:

- There was no significant difference in click duration between older adults and participants with motor impairments ($p = .177$).

- Click durations were significantly shorter in games than they were in MS Office (p < .01) or during Internet use (p < .01).
    - Click durations for older adults were significantly shorter during games than MS Office (p < .001) and during Internet use (p < .001).

Summary of regression results: We found that able bodied participants had the shortest click durations of our three groups. Our regression analysis on Internet and MS Office use across our three groups showed that motor impaired participants had statistically longer click durations than able bodied participants across both applications. A regression analysis on Internet, MS Office and game data from our older adult and motor impaired participants showed that click durations during games were significantly shorter than during Internet or MS Office for both groups.

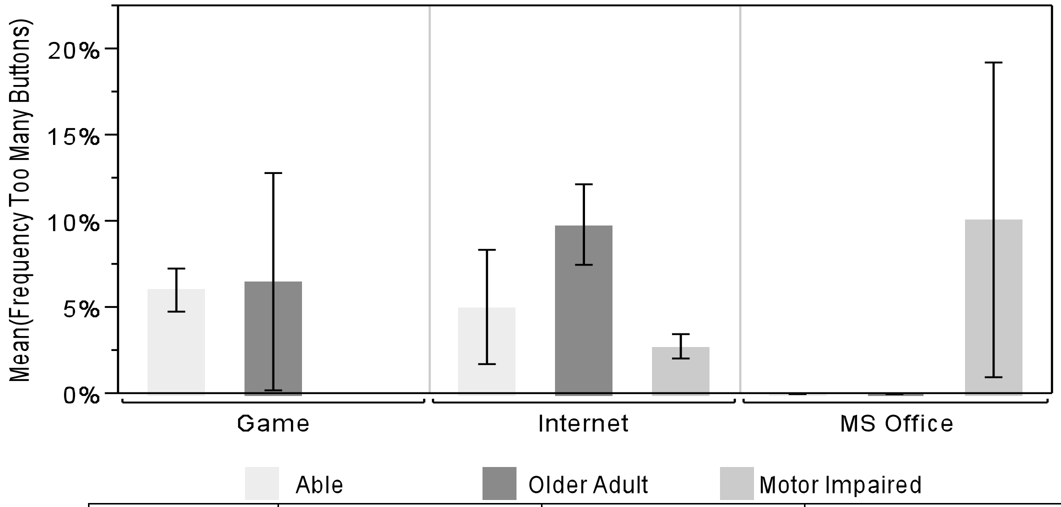
Comparison of results to related work: Our findings are consistent with Keates' investigation of the click durations across four diverse user groups (Keates *et al.* 2005). In their analysis they found the following mean click duration times across all trials in a laboratory tests: Young Adults = 123ms (.123 seconds), Adults = 102ms (.102 seconds), Older Adults = 271ms (.271 seconds), and individuals with Parkinson's disease = 316ms (.316 seconds).

Design implication for assistive adaptations: Individuals who have difficulty quickly pressing and releasing a button may benefit from changing the default click duration settings. However, given that we observed significant differences in this metric across applications, these individuals may benefit most if this setting is based on trends in their behavior, or application specific instead of across all applications.

5.2.3.2. Too Many Buttons

Metric description and calculation: This metric indicates if the participant pressed both buttons on the mouse during a click. This is easily calculated by analyzing the button event log stream for occurrences where a button (either the left or right) gets pressed before the other has been released. We consider pressing too many buttons to be an error, because this is not a recognized action in the Internet and MS Office data sets, or in any of the games we tested (FreeCell, Spider Solitaire, Solitaire, Pinball, and Minesweeper).

Divisions of data: In our base statistics we compared the frequency of pressing too many buttons across Internet, MS Office and games. In our more detailed regression analysis, due to the lack of game data from able bodied participants we only looked at this metric for Internet and MS Office use across our three groups. Note older adults and able bodied participants never pressed both buttons during MS office use.



	Games		Internet		MS Office	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
Able			2.75%	0.0147	10.14%	0.1828
Older Adult	6.06%	0.0360	5.04%	0.0877	0.00%	0.0000
Motor Impaired	6.50%	0.1264	9.80%	0.0405	0.00%	0.0000

Figure 18 Mean and standard deviation of frequency of pressing too many buttons organized by group and application. Note that this error never happened for older adults or able bodied participants during MS Office use.

	Frequency of Pressing Too Many Buttons			
Variable	**Estimate**	**Std Error**	**Prob>ltl**	
Intercept	0.0465	0.0185	0.0330	*
Group: Older Adult	-0.0204	0.0250	0.4341	
Group: Motor Impaired	0.0025	0.0274	0.9301	
Application: Internet	0.0121	0.0173	0.5028	
Older Adult X Internet	0.0122	0.0235	0.6167	
Motor impaired X Internet	0.0369	0.0257	0.1856	
	R^2 = .32			

Table 6: This model presents estimated number of times both mouse buttons were pressed by group: older adult, motor impaired, able bodied (omitted) and application type: Internet or MS Office (omitted). *p < .05, **p < .01, ***p < .001.

Regression results for pressing too many buttons in Internet and MS Office use: We created a hierarchical linear model treating participant as a random effect. Since this is a binary variable, we aggregated the frequency each participant made this error across all sessions by application, thus providing a frequency of this error for each participant during all Internet and all MS Office use. Table 6 presents a model of the frequency both buttons were pressed aggregated by all data from each participant. This model presents the main effects between user group (able bodied adults, motor impaired adults, and older adults) and Application (Internet or MS Office) ($R^2 = .32$).

This model did not find a significant difference in number of times both buttons were pressed between older adults and able bodied users ($p = .434$) or between motor impaired users and able bodied users ($p = .9301$) across both application types. There was no significant difference across applications, and no interaction between Application type and Group.

Summary of regression results: Participants from all three groups pressed both buttons on the mouse occasionally. All but one of our participants experienced this problem in one of these three applications. Although not statistically significant, we saw a trend in Figure 18 that older adults and individuals with motor impairments experienced this problem more frequently than able bodied participants. Though not represented in the model above, in our laptop deployment we saw more cases where participants clicked both mouse buttons than we did in our desktop deployment.

Performance across sessions: Figure 19 plots the frequency of pressing too many buttons across all of our able bodied and motor impaired participants (and 4 of our older adults) from 10 randomly selected sessions of Internet and MS Office use. Appendix 10.4.1 shows data from all participants. These charts illustrate that some participants never experienced this problem, while some individuals experienced it more frequently. It also illustrates how this was an intermittent problem, and that some participants (especially from the motor impaired group) experienced it very frequently in a session.

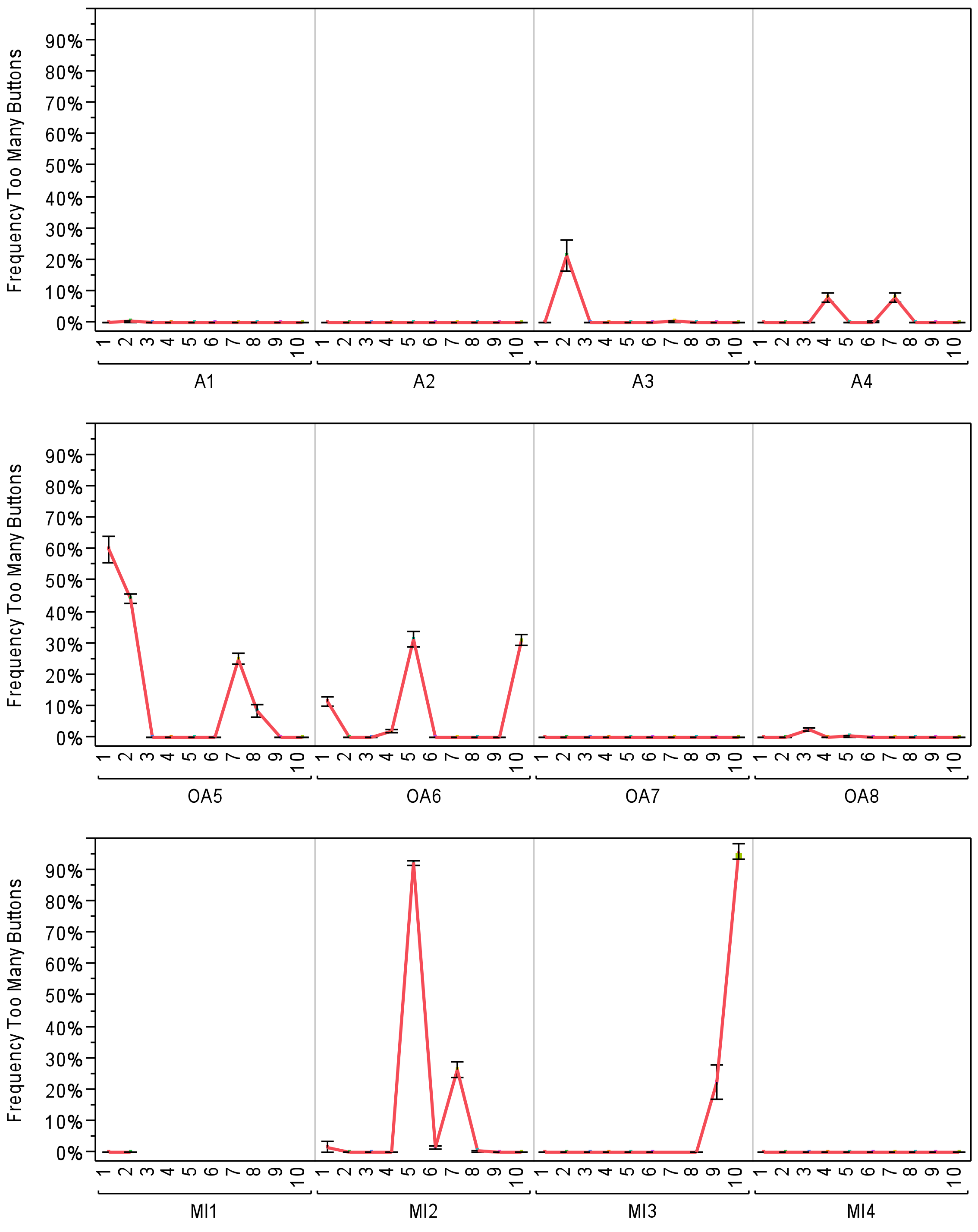


Figure 19 Frequency of pressing too many buttons in ten randomly selected sessions. These charts are organized by group (A = able bodied, OA = older adult, MI = motor impaired) and within each group, participant and session number.

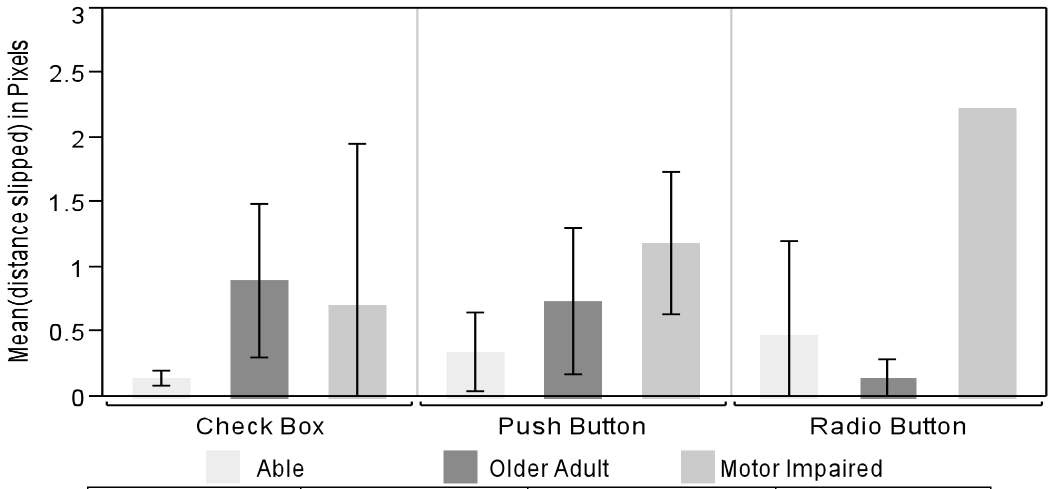
Comparison of results to related work: Our analysis is consistent with findings from our desktop deployment and research by Trewin, which showed that motor impaired individuals have difficulty pressing too many buttons (Trewin *et al.* 2006). Additionally, our laptop results show that able bodied individuals and older adults also occasionally have this problem (which was not investigated in our desktop deployment or by Trewin).

Design implication for assistive adaptations: While we did not observe that this problem happened frequently, we still believe that it is still a useful metric for assessment since it is an indication of a clicking problem. As we mentioned in Section 2.3.1, individuals who frequently experience this problem could be assisted by remapping button functions, or replacing clicking with another interaction style, such as crossing (Accot and Zhai 2002).

5.2.3.3. Slipping During a Button Press

Metric description and calculation: Slipping occurs when a user unintentionally moves the cursor during a click (accidentally executing a drag). This metric is calculated by measuring the Euclidean distance traveled while a mouse button is pressed. Because past work shows that movements during a click under 100 pixels tend to be slips and not drags, we labeled movements as drags using this threshold. (Trewin *et al.* 2006).

Divisions of data: To help distinguish between accidental slips and intentional drags, we limit our slip analysis by interactor type and distance moved. In our slip analysis, we only looked at the distance slipped and frequency of slips on interactors that are not draggable in our MS Office and Internet data sets. These interactors were check boxes, push buttons and radio buttons. These interactors came in a wide range of sizes. Check boxes and radio buttons were the smallest of these interactors when we took the mean of the smaller dimension of each target in pixels (check box mean = 13.28, SD = 1.42, radio button mean = 13.00, SD = 0), and push buttons were the largest (mean = 26.3, SD = 7.3). This analysis was performed on Internet and MS Office data since these interactors were rarely found in our game data.



	Check Box		Push Button		Radio Button	
	Mean	**Std Dev**	**Mean**	**Std Dev**	**Mean**	**Std Dev**
Able	0.1473	0.06092	0.3509	0.303198	0.4751	0.726288
Older Adult	0.9005	0.59192	0.7335	0.565243	0.1420	0.14226
Motor Impaired	0.7148	1.23803	1.1878	0.54802	2.2361	.

Figure 20 Mean and standard deviation of distance slipped in pixels organized by group and application. Note that there was only one interaction with a radio button from the motor impaired group, and it was 2 pixels of slip.

	Distance Slipped			
Variable	**Estimate**	**Std Error**	**Prob>ltl**	
Intercept	-1.8077	0.2036	<.0001	***
Group: Older Adult	-0.1241	0.2460	0.6187	
Group: Motor Impaired	1.0130	0.3455	0.0049	**
Interactor: Push Button	-0.3375	0.1325	0.0109	*
Interactor: Radio Button	0.3994	0.2564	0.1193	
Push Button X Older Adult	0.2196	0.1390	0.1141	
Push Button X Motor Impaired	-0.5830	0.2613	0.0257	*
Radio Button X Older Adult	-0.9087	0.2687	0.0007	***
Radio Button X Motor Impaired				
	R^2 = .13			

Table 7: This model presents estimated distance slipped (as a log base 10 of pixels) group: older adult, motor impaired, able bodied (omitted), and interactor type: push button or radio button or check box (omitted). *p < .05, **p < .01, ***p < .001.

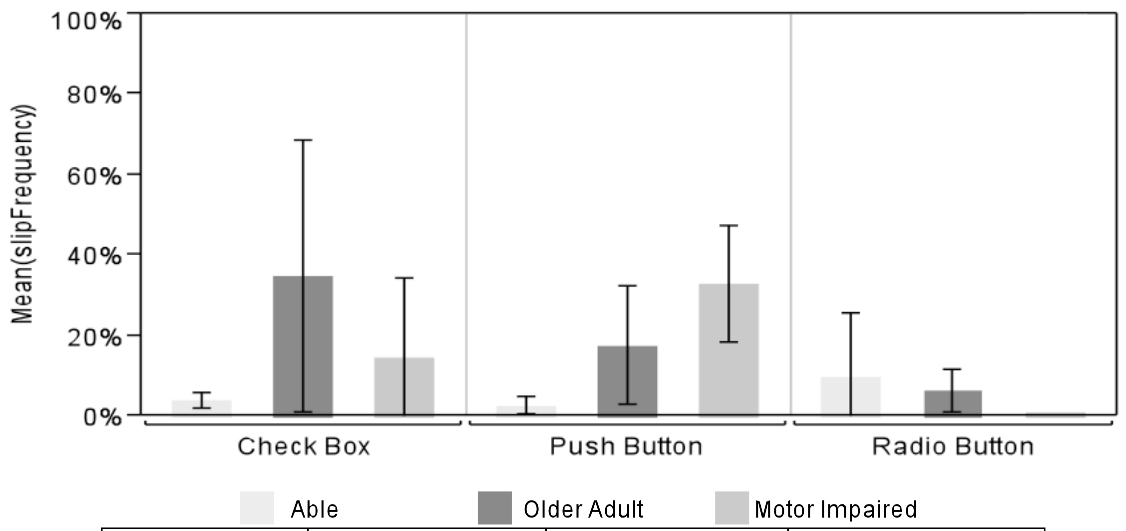
Regression results for distance slipped across interactors for Internet and MS Office use: We created a hierarchical linear model treating participant as a random effect. Table 7 presents a model of distance slipped. This model presents the main effects and interactions between user group (able bodied adults, motor impaired adults, and older adults) and interactor type (check box, push button and radio button) (R^2 = .13). Since the distance slipped is a continuous variable that roughly followed a Power Law distribution, we use the log of it in this analysis.

We have omitted the interaction of radio buttons for motor impaired participants because we only had one sample with a participant slipping on a radio button.

This model illustrates the following trends:

- There was a significant difference in distance slipped between participants with motor impairments and able bodied users (p < .01) across all interactor types, but not between able bodied users and older adults (p = .619).
- Across all groups, slips were longer on check boxes than push buttons (p < .05), but not significantly different between check boxes and radio buttons (p = .119).
- There was a main effect for interactor type on distance slipped across all groups. For all of the interactor types listed below (with a significant difference), able bodied participants had shorter slips than either older adults or motor impaired participants.
    - For push buttons, there was a significant difference in distance slipped between able bodied and motor impaired participants (p < .05), but not between able bodied and older adults (p = .114).
    - For radio buttons there was a significant difference in the distance slipped between able bodied participants and older adults (p < .001). We do not include the results for motor impaired participants since there was only one interaction with a radio button.

Regression results for frequency of slips across interactors for Internet and MS Office use: In addition to analyzing the distance slipped, we also investigated the frequency of slips (any slip with a distance greater than 0) across the same interactors. Figure 21 shows the mean frequency of slips by each interactor type and across groups.



	Check Box		Push Button		Radio Button	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
Able	3.92%	0.017585	2.53%	0.022553	9.71%	0.157522
Older Adult	35.53%	0.37758	13.85%	0.147341	6.11%	0.053576
Motor Impaired	12.82%	0.222058	27.16%	0.133552	.	.

Figure 21 Mean and standard deviation of frequency of slips (under 100 pixels) organized by group and application. Note that there was only one sample with a slip from the motor impaired group on a radio button.

	Frequency of Slips			
Variable	**Estimate**	**Std Error**	**Prob>ltl**	
Intercept	0.2477	0.0430	<.0001	***
Group: Older Adult	-0.0462	0.0516	0.3777	
Group: Motor Impaired	0.2637	0.0740	0.0007	***
Interactor: Push Button	-0.1008	0.0294	0.0006	***
Interactor: Radio Button	0.1576	0.0570	0.0057	**
Push Button X Older Adult	0.0651	0.0309	0.035	*
Push Button X Motor Impaired	-0.1688	0.0581	0.0037	**
Radio Button X Older Adult	-0.2514	0.0597	<.0001	***
Radio Button X Motor Impaired				
	R^2 = .80			

Table 8: This model presents estimated frequency of slips by group: older adult, motor impaired, able bodied (omitted), and interactor type: push button or radio button, and check box (omitted). $^*p < .05$, $^{**}p < .01$, $^{***}p < .001$.

We created a hierarchical linear model treating participant as a random effect. Table 8 presents the model of frequency of slips. This model presents the main effects and interactions between user group (able bodied adults, motor impaired adults, and older adults) and interactor type (check box, push button and radio button) (R^2 = .80). We have omitted the intersection of radio buttons for motor impaired participants because we only had one sample with a participant slipping on a radio button.

This model illustrates the following trends:

- Motor impaired participants were significantly more likely to slip during a button press than able bodied adults (p < .001); however, older adults were not (p = .377).
- Across all groups, participants were more likely to slip on radio buttons than check boxes (p < .01). Likewise, they were more likely to slip on check boxes than push buttons (p < .001). This is likely because radio buttons are smaller than checkboxes, which are smaller than push buttons.
- There was a main effect for interactor type on frequency of slips across all groups.
    - For push buttons, motor impaired and older adults were more likely to slip than able bodied participants (older adults p < .05; motor impaired p < .01).
    - For radio buttons, older adults were more likely to slip than able bodied participants (p < .001).

Summary of regression results: Our regression analysis found that motor impaired participants slipped farther than able bodied users. We also found that there were longer slips on checkboxes than on push buttons across all participants. Finally, we found that able bodied participants tended to have shorter slips than older adults or participants with motor impairments.

Our regression on frequency of slips showed that motor impaired participants experienced more slips than our able bodied group. Across all participants, slips were more likely to occur on checkboxes than on push buttons. This was most likely because check boxes were so much smaller than these other interactors. When we looked at the interaction between group and interactor type, we saw that older adults were more likely to slip on radio buttons than able bodied participants. Motor impaired participants were more likely the slip on push buttons than able bodied participants.

Comparison of results to related work: Trewin and Keates have studied slipping problems in laboratory tasks (Keates *et al.* 2005, Trewin *et al.* 2006). In these studies they found slipping errors to be one of the problems encountered by individuals with motor impairments. In an exploratory study of pointing behavior from individuals with motor impairments, they found slipping account for 15% of all clicking errors (Keates et al. 2005). In a study of individuals who had slipping problems, 55% of all clicking errors (Trewin *et al.* 2006) were slips. In both of these studies, participants interacted with push buttons. In our data, we found that 27.17% of clicks on push buttons by individuals with motor impairments were slips. Unfortunately, we cannot directly compare our mean slip distance results to theirs because they do not report the mean distance slipped.

Design implication for assistive adaptations: As we have mentioned in Section 2.2.3, the Steady Click adaptation (Trewin *et al.* 2006) can help individuals who have difficulty slipping by temporarily disabling dragging. Given the frequency of this problem for motor impaired participants, we believe this adaptation could greatly benefit individuals who slip off a target, causing a failed click attempt.

5.2.4. Movement Performance Metrics

In this section we discuss movement performance metrics. These metrics evaluate how the cursor moved between targets, and measure efficiency and speed. Given that many of these metrics use movement time and distance traveled (both total distance and Euclidean distance to a target) our principal component analysis (Section 5.2.2.3) found them to be in the same factor. The following section will show that some of these metrics varied dramatically across groups and applications, and some did not.

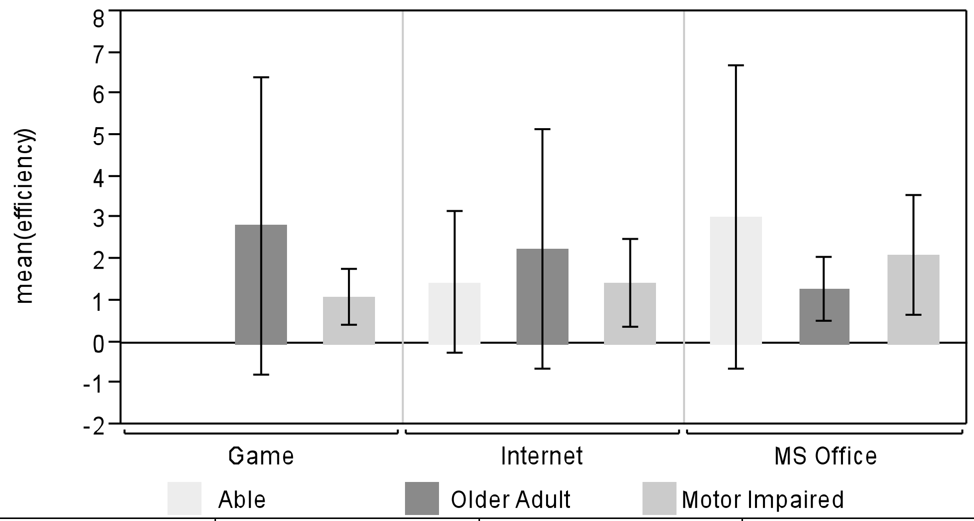
5.2.4.1. Efficiency, or Excess Distance Traveled

Metric description and calculation: The ratio between the distance traveled and the Euclidean distance between targets is a measure that has been used to quantify motor impaired use (Keates *et al.* 2002A, Keates *et al.* 2005). We use a normalized version of this ratio in our analysis to account for overall movement length. When looking at this ratio, values closer to 0 indicate perfect (or efficient movement) because there is little difference between the total distance and Euclidean distance moved. Values larger than one represent a less direct path.

$$Efficiency = \frac{DistanceToTarget - EuclideanDistanceToTarget}{EuclideanDistanceToTarget}$$

Equation 2 Normalized efficiency or excess distance traveled to target (values closer to 0 indicate the most efficient movement)

Divisions of data: In our base statistics we compared efficiency of movement across Internet, MS Office and games. In our more detailed regression analysis, due to the lack of game data from able bodied participants we only looked at this metric for Internet and MS Office use across our three groups.



	Games		Internet		MS Office	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
Able			1.4494	1.7060	3.0319	3.6843
Older Adult	2.8176	3.5989	2.2446	2.8889	1.2919	0.7801
Motor Impaired	1.0853	0.6883	1.4406	1.0596	2.1192	1.4497

Figure 22. Mean and standard deviation of movement efficiency (total distance / Euclidean distance traveled) organized by group and application

	Efficiency of Movement (simple)				Efficiency of Movement (complex)			
Variable	Estimate	Std Error	Prob>ltl		Estimate	Std Error	Prob>ltl	
Intercept	4.2355	0.5254	<.0001	***	4.2629751	0.554074	<.0001	***
Group: Older Adult	0.0906	0.8649	0.9166		0.022951	0.866704	0.9789	
Group: Motor Impaired	-0.4547	0.7095	0.5217		-0.303703	0.718586	0.6726	
Application: Internet	0.0679	0.5030	0.8927		0.0313966	0.50307	0.9502	
Older Adult X Internet	0.6436	0.8513	0.4496		0.571869	0.851655	0.5019	
Motor impaired X Internet	-0.0764	0.6943	0.9124		-0.052887	0.694319	0.9393	
Index of Difficulty	-0.8721	0.0457	<.0001	***	-0.857124	0.104975	<.0001	***
Older Adult X Index of Difficulty					-0.357855	0.128008	0.0052	**
Motor impaired X Index of Difficulty					0.3305785	0.194645	0.0895	
	$R^2 = .00$				$R^2 = .00$			

Table 9: Our simple model presents estimated normalized efficiency of movement (total distance traveled divided minus Euclidean distance traveled, divided by Euclidean distance traveled) by group: older adult, motor impaired, able bodied (omitted), and application type: Internet or MS Office (omitted). Our complex model includes interactions between user group and Index of Difficulty. *p < .05, **p < .01, ***p < .001.

Regression results for normalized movement efficiency for Internet and MS Office use: We created a hierarchical linear model treating participant as a random effect. Table 9 presents the results of two models of efficiency of movement. The simple model presents the main effects for user group (able bodied adults, motor impaired adults, and older adults), application (MS Office, Internet), and Index of Difficulty, as well as interactions between user group and application ($R^2 = .00$. Overall this regression analysis showed no significant effects (the R^2 of the model was extremely low). The complex model adds an interaction between user group and ID, with no change ($R^2 = .00$).

Summary of regression results: Overall, there is not a clear trend for which group was the most efficient. We expected that able bodied participants would have been the most efficient, however we did not see this effect.

Comparison of results to related work: This metric can be highly affected by individual difference, and is especially affected by spastic movements (which none of our participants had). In a laboratory study, Keates *et al.* found that older adults had the least efficient movement (1.7874) when compared to young adults (1.3762), adults (1.4111), and individuals with Parkinson's Disease (1.213) (Keates *et al.* 2005). They found the mean of this metric to be significantly different across groups at $p < .05$. Note that Keates used the ratio between distance traveled and Euclidean distance, and did not control for overall movement length as we did.

Performance across sessions: Figure 23 plots the mean normalized efficiency across all of our able bodied and motor impaired participants (and 4 of our older adults) from ten randomly selected sessions of Internet and MS Office use. Data from all participants is in Appendix 10.4.2. These charts show that efficiency varied across sessions and participants, suggesting a possible reason our regression provide inconclusive results. This variance is especially clear for older adults, a group where we see that some individuals (OA3 and OA4) were efficient with little variance across sessions, and others (A2 and MI4) are less efficient, and varied across sessions.

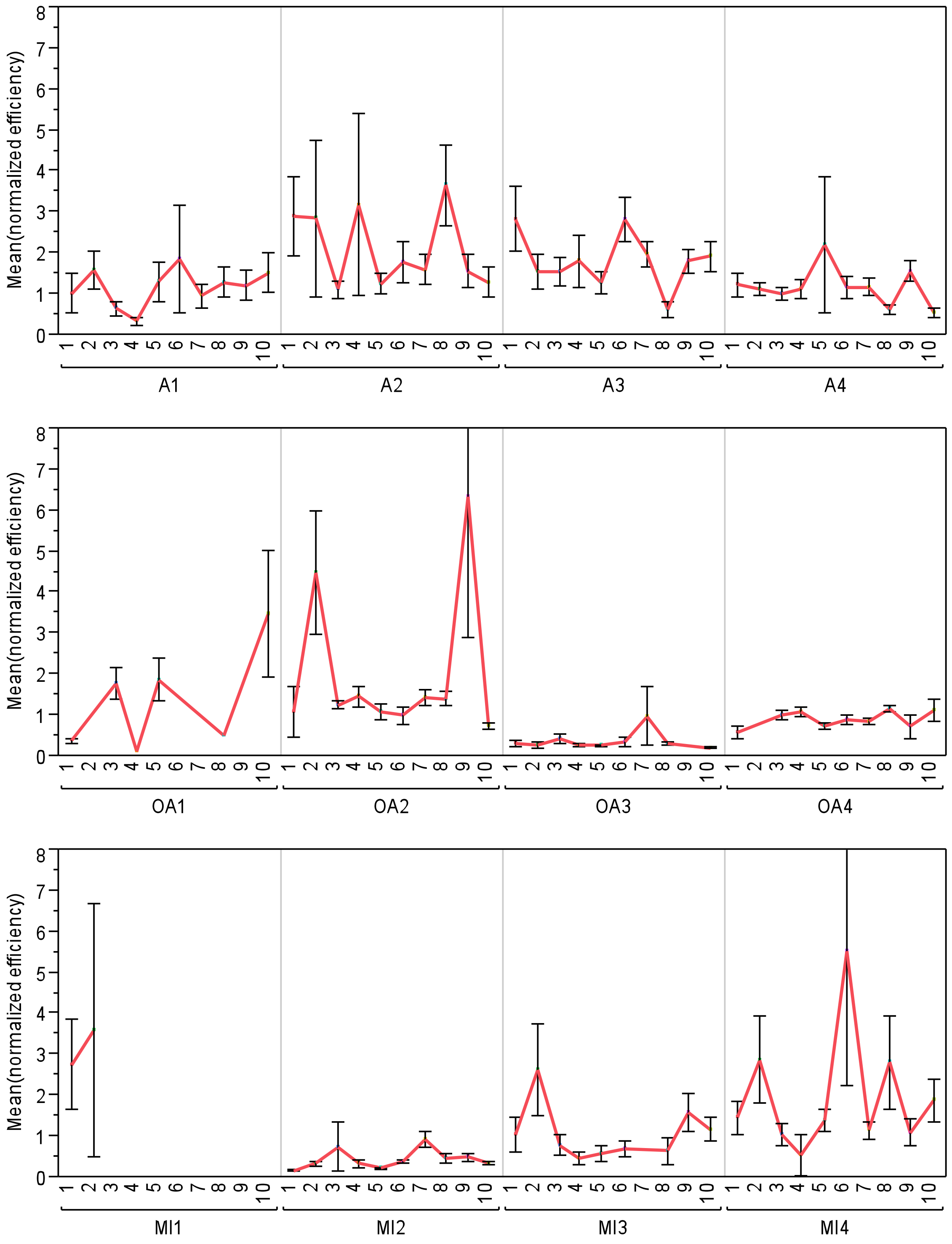


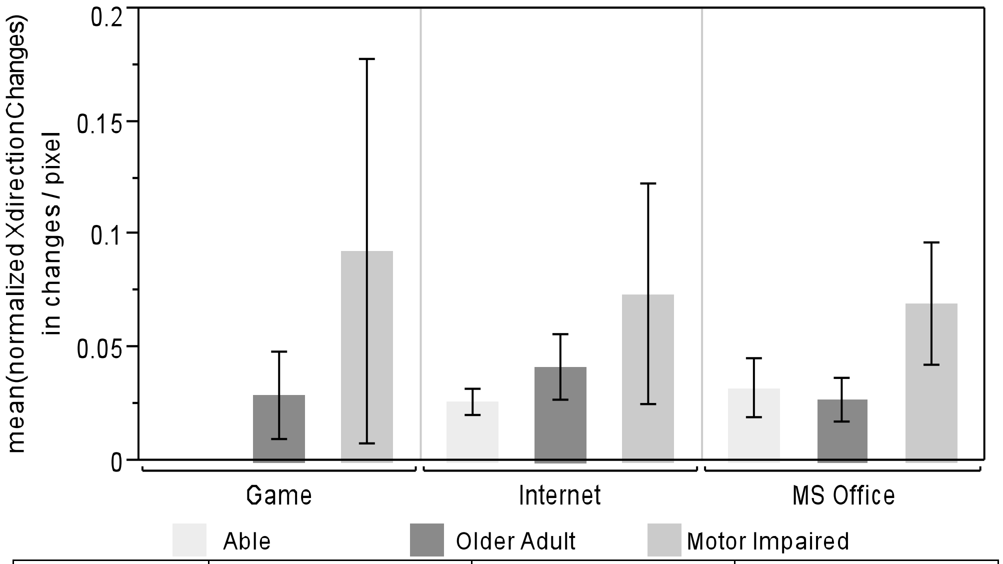
Figure 23 Normalized efficiency across ten randomly selected sessions. These charts are organized by group (A = able bodied, OA = older adult, MI = motor impaired) and within each group, participant and session number.

Design implication for assistive adaptations: Individuals who have extremely inefficient movement may benefit from adaptations that help them acquire targets. Examples of these adaptations include automatically jumping or "snapping" the cursor to a target (Sutherland 1964), or changing how the cursor behaves near a target (Balakrishnan 2004). However, given the high potential cost of errors (placing a user on an unintended target) the effectiveness and configuration of these adaptations in real world interfaces needs to be investigated. We also believe further investigation is needed to distinguish between inefficient movements caused by pointing problems, versus inefficient movements caused by "meandering" behavior.

5.2.4.2. Changes in Direction

Metric description and calculation: The number of direction changes during a sample is another metric to describe movement and give an estimate of efficiency (in addition to excess distance traveled). There are many potential reasons an individual may experience a large number of direction changes, such as a tremor that makes straight movement difficult. Another potential cause for a large number of direction changes is wiggling the cursor to either wake up the screen (if it has fallen asleep), or when the user can't immediately find the cursor. We investigate direction change by normalizing it by the Euclidean distance traveled since it is likely that longer motions will have more direction changes.

Divisions of data: In our base statistics we compared the number of normalized X and Y changes across Internet, MS Office and games. In our more detailed regression analysis, due to the lack of game data from able bodied participants we only looked at this metric for Internet and MS Office use across our three groups. We did not include Index of Difficulty in this model because these metrics are normalized by the Euclidean distance traveled (the log of which is used in the calculation of ID), instead we included log (base 10) of the target size.



	Games		Internet		MS Office	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
Able			0.026371	0.005735	0.032231	0.012941
Older Adult	0.029234	0.019546	0.041493	0.014466	0.026964	0.009523
Motor Impaired	0.092916	0.085009	0.073792	0.048513	0.069344	0.026817

Figure 24 Mean and standard deviation of X direction changes (normalized by Euclidean distance traveled) organized by group and application (direction changes / pixel).

	X Direction Change (simple)				X Direction Change (complex)			
Variable	**Estimate**	**Std Error**	**Prob>ltl**		**Estimate**	**Std Error**	**Prob>ltl**	
Intercept	-1.7778	0.0467	<.0001	***	-1.7860	0.0499	<.0001	***
Group: Older Adult	-0.0220	0.0630	0.7312		-0.0385	0.0635	0.5531	
Group: Motor Impaired	0.2314	0.0674	0.0050	**	0.2439	0.0681	0.0038	**
Application: Internet	-0.0167	0.0189	0.3772		-0.0162	0.0189	0.3910	
Older Adult X Internet	0.0355	0.0298	0.2330		0.0514	0.0298	0.0842	
Motor impaired X Internet	-0.0408	0.0290	0.1590		-0.0551	0.0291	0.0582	
Log10(Target Size)	0.0509	0.0049	<.0001	***	0.0549	0.0120	<.0001	***
Older Adult X Log10(Target Size)					-0.1344	0.0140	<.0001	***
Motor impaired X Log10(Target Size)					0.1176	0.0226	<.0001	***
	R^2 = 0.05				**R^2 = 0.06**			

Table 10: Our simple model presents estimated count X direction changes normalized by Euclidean distance traveled (log base 10) by group: older adult, motor impaired, able bodied (omitted), and application type: Internet or MS Office (omitted) and log10(target size). Our complex model includes interactions between user group and log10(target size).
*p < .05, **p < .01, ***p < .001.

Regression results on normalized X direction changes for Internet and MS Office use: We created a hierarchical linear model treating participant as a random effect. Table 10 presents the results of two models of X direction changes. The simple model presents the main effects and interactions between user group (able bodied adults, motor impaired adults, and older adults) and application type (MS Office, Internet) and a main effect for the target size (log base 10) ($R^2 = .05$). Our regression analysis uses the log (base 10) of the normalized number of X direction changes.

Our simple model illustrates the following trends:

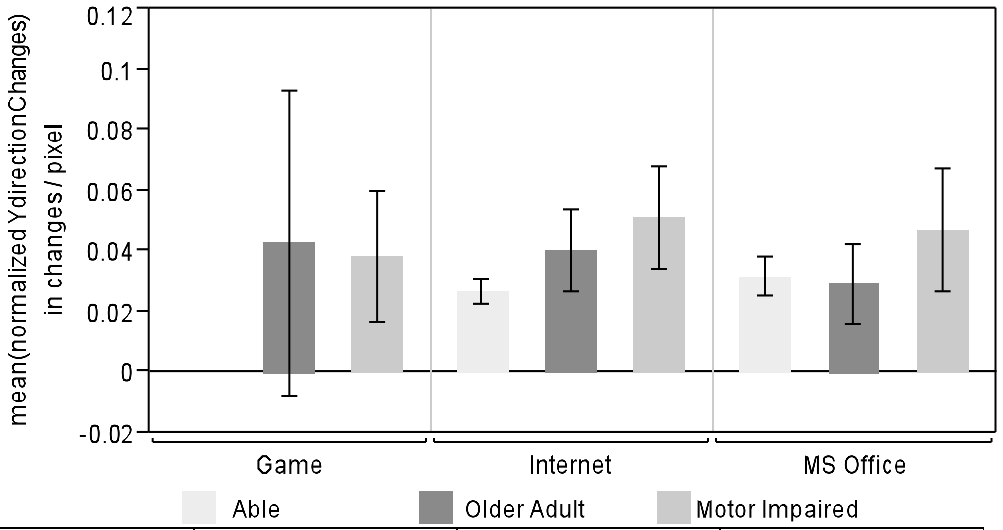
- Individuals with motor impairments had significantly more X direction change per Euclidean distance traveled than able bodied participants ($p < .01$) across application type with constant target size, however there was no significant difference in performance in this metric between older adults and able bodied users ($p = .731$).
- There was no main effect for application type on number of X direction changes ($p = .377$) across all groups controlling for target size. Likewise there were no interactions between application and group.
- Target size has a significant impact on the number of X direction changes ($p < .001$) across all application types and groups. Larger targets are associated with more X direction changes.

To understand the effect of target size on X direction changes across groups, our more complex model adds an interaction between user group and target size ($R^2 = .06$).

- There was a significant interaction between target size and user group. Though all groups had more X direction changes for larger targets, the effect was larger for motor impaired individuals ($p < 0.001$) and not as large for older adults ($p < .001$) when compared to able bodied individuals (illustrated in table 11).
- Table 11 shows the estimated number of X direction changes for a small target (10 pixels in the smallest dimension) and a large target (100 pixels in the smallest dimension) during Internet use. For these targets, motor impaired participants had the most direction changes per distance traveled, and able bodied adults had the fewest.

	Target Size = 10 pixels	Target Size = 100 pixels
Able	.0110 (changes/pixel)	.0130 (changes/pixel)
Older Adults	.0221 (changes/pixel)	.0184 (changes/pixel)
Motor Impaired	.0235 (changes/pixel)	.0350 (changes/pixel)

Table 11 estimated number of normalized X direction changes during Internet use for a target whose size is either 10 or 100 pixels.



	Games		Internet		MS Office	
	Mean	**Std Dev**	**Mean**	**Std Dev**	**Mean**	**Std Dev**
Able			0.0268	0.0041	0.0316	0.0062
Older Adult	0.0427	0.0503	0.0403	0.0134	0.0293	0.0132
Motor Impaired	0.0380	0.0217	0.0511	0.0171	0.0468	0.0203

Figure 25 Mean and standard deviation of Y direction changes (normalized by Euclidean distance traveled) organized by group and application (direction changes / pixel).

	Y Direction Change (simple)				Y Direction Change (complex)			
Variable	**Estimate**	**Std Error**	**Prob>ltl**		**Estimate**	**Std Error**	**Prob>ltl**	
Intercept	-1.7256	0.0442	<.0001	***	-1.7441	0.0475	<.0001	***
Group: Older Adult	-0.0383	0.0603	0.5337		-0.0473	0.0608	0.4474	
Group: Motor Impaired	0.1962	0.0636	0.0089	**	0.2037	0.0642	0.0076	**
Application: Internet	0.0130	0.0196	0.5083		0.0131	0.0196	0.5044	
Older Adult X Internet	0.0572	0.0312	0.0671		0.0654	0.0313	0.0366	*
Motor impaired X Internet	-0.0316	0.0297	0.2882		-0.0391	0.0298	0.1899	
Log10(Target Size)	-0.0077	0.0049	0.1164		0.0037	0.0120	0.7580	
Older Adult X Log10(Target Size)					-0.0766	0.0140	<.0001	***
Motor impaired X Log10(Target Size)					0.0783	0.0226	0.0005	***
	R^2 = 0.05				R^2 = 0.06			

Table 12: Our simple model presents estimated count Y direction changes normalized by Euclidean distance traveled (log base 10) by group: older adult, motor impaired, able bodied (omitted), and application type: Internet or MS Office (omitted) and log10(target size). Our complex model includes interactions between user group and log10(target size).
*p < .05, **p < .01, ***p < .001.

Regression results on normalized Y direction changes for Internet and MS Office use: We performed the same analysis on Y direction changes that we used for X direction changes. We created a hierarchical linear model treating participant as a random effect. Table 12 presents the results of two models of Y direction changes. Our simple model presents the main effects and interactions between user group (able bodied adults, motor impaired adults, and older adults) and application type (MS Office, Internet) and a main effect for the target size (log base 10) ($R^2 = .05$). We also created a more complex model that adds an interaction between user group and target size ($R^2 = .06$). Our regression analysis uses the log (base 10) of the normalized number of Y direction changes.

Results for Y direction changes were qualitatively similar to X direction changes, with the exception of no main effect for target size.

Summary of regression results: Our regression analysis showed that motor impaired participants had significantly more direction changes (both X and Y) per distance traveled than able bodied users. When we looked at the interaction of group on target size, we found that both older adults and motor impaired individuals had more direction changes per distance moved than able bodied adults. The estimates from this model showed that motor impaired individuals had more than twice as many direction changes per pixel traveled than able bodied participants (for both small and large targets).

Performance across sessions: Figure 26 plots the mean number of X and Y direction changes (normalized by Euclidean distance traveled) across all of our able bodied and motor impaired participants (and 4 of our older adults) in ten randomly selected sessions of Internet and MS Office use. Appendix 10.4.3 shows data from all participants. In these charts X direction changes are indicated red, and Y direction changes with a blue. These charts illustrate that the number of direction changes appear to vary more across sessions for older adults and motor impaired participants than able bodied individuals. Additionally, there was variability across participants in the older adult group in this metric (OA4 and OA8) had few direction changes and little variability, while OA1 and OA5 had more direction changes and variability.

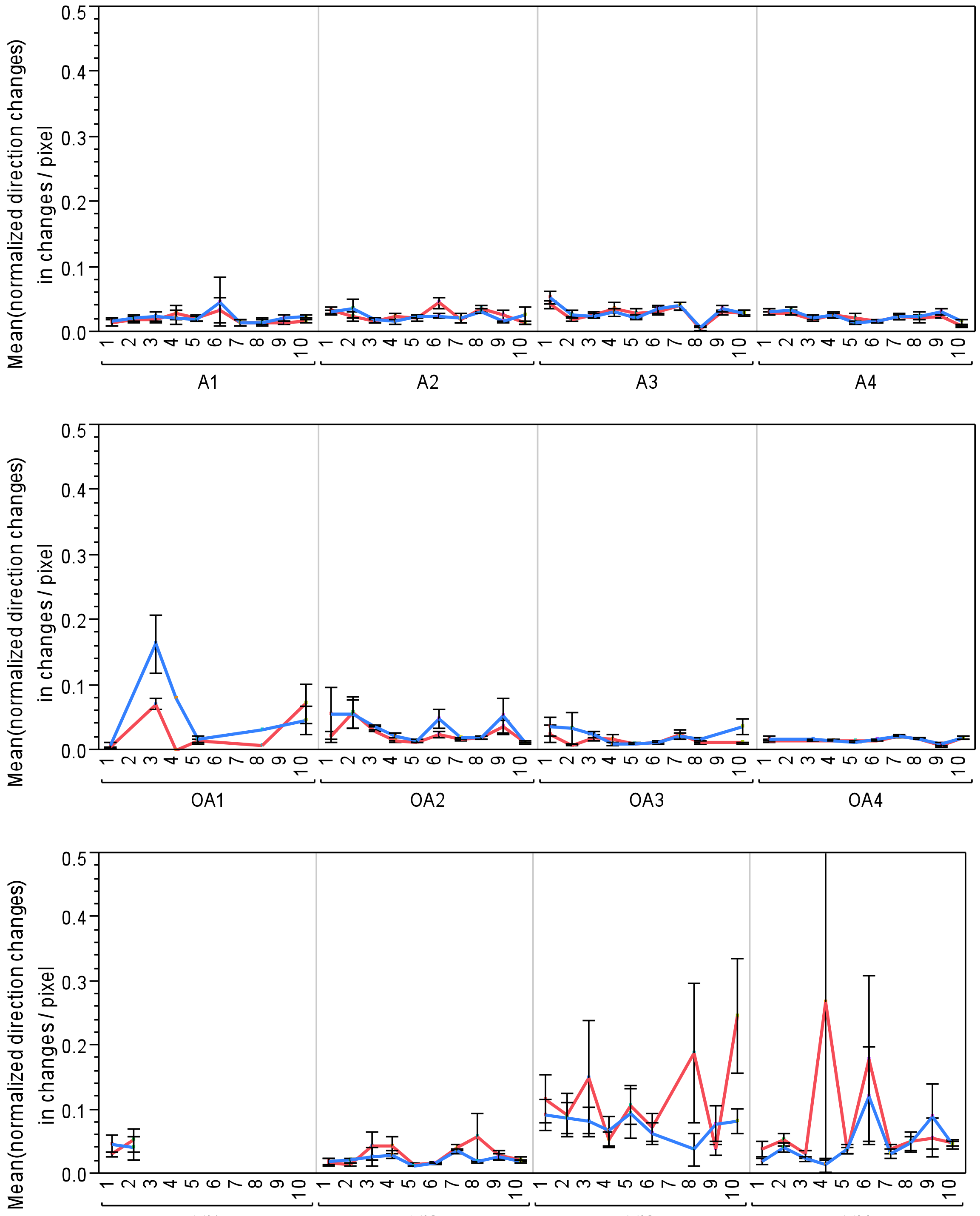


Figure 26 Normalized number of direction changes across ten randomly selected sessions. These charts are organized by group (A = able bodied, OA = older adult, MI = motor impaired) and within each group, participant and session number.

Comparison of results to related work: Related work by Keates has explored the difference in the raw count of direction changes for both individuals with and without pointing problems (Keates *et al.* 2002A, Keates *et al.* 2005). They also observed that older adults and individuals with motor impairments had more direction changes than able bodied individuals (Keates *et al. 2005*). However, since they did not take the Euclidean distance traveled into account, there may be a slight difference in our results.

Design implication for assistive adaptations: Poor performance for this metric (high ratios between number of direction changes and distance traveled) can occur for a variety of reasons. Two likely reasons we observed poor performance include difficulty moving in a straight line (possibly due to an intermittent tremor) or difficulty selecting a target (possibly due to a slipping problem). Appropriate adaptations depend on the cause of this performance problem. Individuals who have difficulty moving in a straight line would most likely benefit from an adaptation that smoothes the cursor movement (to remove a tremor or sporadic movement) (Levine and Schappert 2002). Individuals who have difficulty targeting may benefit more from an adaptation that is designed to help with targeting problems such as steady clicks (Trewin *et al.* 2006) or changing how the cursor interacts with a target (Worden *et al.* 1997, Balakrishnan 2004). Further investigation is required to automatically detect which kind of efficiency problem an individual is experiencing.

5.2.4.3. Throughput

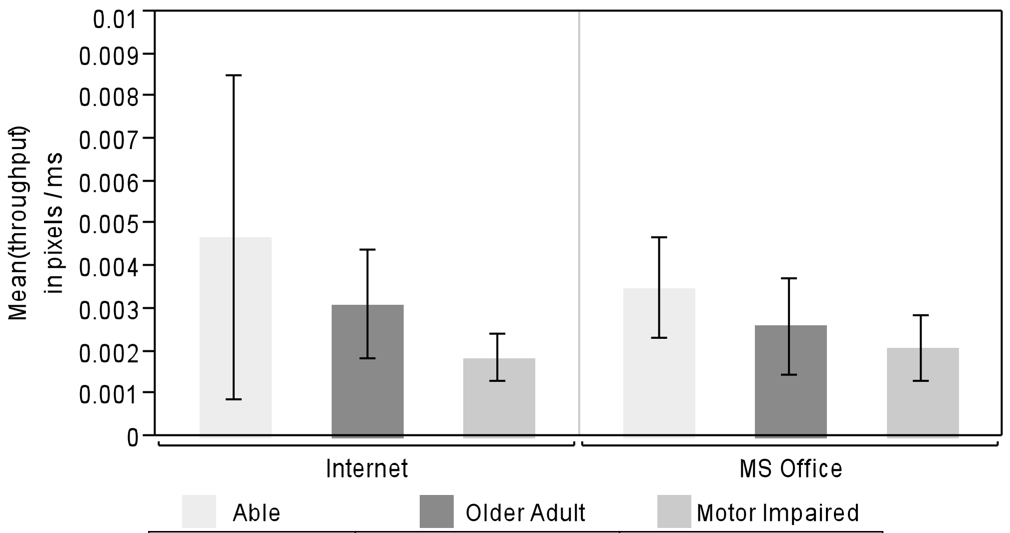
Metric description and calculation: Throughput, or Index of Performance, is a common metric that describes movement speed, independent of target size and distance to the target (ISO 9241-9:2000(E) 2002). Generally speaking, movement time should increase as index of difficulty increases (i.e. it should take individuals longer to reach harder targets than easy ones) (ISO). Throughput is computed by dividing the Index of Difficulty (ID) by movement time to a target (Equation 2). ID (Section 5.2.1.2) is a measure of how difficult a movement is based on the target size and distance to that target.

$$Throughput = \frac{ID}{Time}$$

Equation 3 Throughput (bits per millisecond) is a combined speed/accuracy metric

Divisions of data: In our base statistics we compared throughput across Internet, MS

Office and games. In our more detailed regression analysis, due to the lack of game data from able bodied participants we only looked at this metric for Internet and MS Office use across our three groups. It is important to note that when we look at throughput, we are only able to evaluate performance on targets that are supported by the accessibility API. This includes the Microsoft Word and Internet data but not the games data, and thus excludes a majority of the motions logged for elders and motor impaired users.



	Internet		MS Office	
	Mean	**Std Dev**	**Mean**	**Std Dev**
Able	0.0047	0.0038	0.0035	0.0012
Older Adult	0.0031	0.0013	0.0026	0.0011
Motor Impaired	0.0019	0.0006	0.0021	0.0008

Figure 27 Mean and standard deviation of throughput organized by group and application (throughput is in bits per millisecond).

	Throughput (simple)			
Variable	**Estimate**	**Std Error**	**Prob>ltl**	
Intercept	-1.9029	0.0054	<.0001	***
Group: Older Adult	-0.0026	0.0074	0.723	
Group: Motor Impaired	-0.0248	0.0078	0.006	**
Application: Internet	-0.0060	0.0022	0.007	**
Older Adult X Internet	0.0011	0.0036	0.762	
Motor impaired X Internet	0.0002	0.0033	0.942	
	R^2 = .04			

Table 13: Our simple model presents estimated throughput (log base 10) in milliseconds by group: older adult, motor impaired, able bodied (omitted), and application type: Internet or MS Office (omitted). *p < .05, **p < .01, ***p < .001.

Regression results for throughput for Internet and MS Office use: We created a hierarchical linear model treating participant as a random effect. Table 13 presents the results of a model of throughput. Our model presents the main effects and interactions between user group (able bodied adults, motor impaired adults, and older adults) and application type (MS Office, Internet) ($R^2 = .04$).

Our model illustrates the following trends:

- Motor impaired participants had a lower throughput than able bodied users ($p < .01$) across all application types, and there was no significant difference between older adults and able bodied users ($p = .72$).
- There was a main effect for application on throughput ($p < .01$) across all groups where throughput was higher in MS Office than during Internet use.
    - There was no significant interaction in throughput between application and group for able bodied users compared to older adults ($p = .762$) or motor impaired users ($p = .942$).

Summary of regression results: Able bodied participants had higher throughput than motor impaired participants. This means that able bodied participants were moving the fastest for any given ID. Additionally, our regression analysis showed that there was a main effect for application across group, where participants had higher throughput during MS office use than Internet use.

Comparison of results to related work: Throughput has been well investigated for able bodied individuals. In a laboratory study investigating throughput across pointing devices, MacKenzie found a mean throughput of 4.9 bits per second (MacKenzie *et al.* 2001). This value is very similar to our throughput for able bodied participants during Internet use (4.7 bits per second). In a laboratory investigation, Wobbrock and Gajos found higher throughput for able bodied users (5.79 bits per second) than individuals with motor impairments (3.07 bits per second) (Wobbrock and Gajos 2008).

Performance across sessions: Figure 28 plots the mean throughput across all of our able bodied and motor impaired participants (and 4 of our older adults) in ten randomly selected sessions of Internet and MS Office use. Appendix 10.4.4 shows data from all participants. These charts illustrate that throughput varied more for some individuals (OA6, OA7) than others (A2, A4) across sessions.

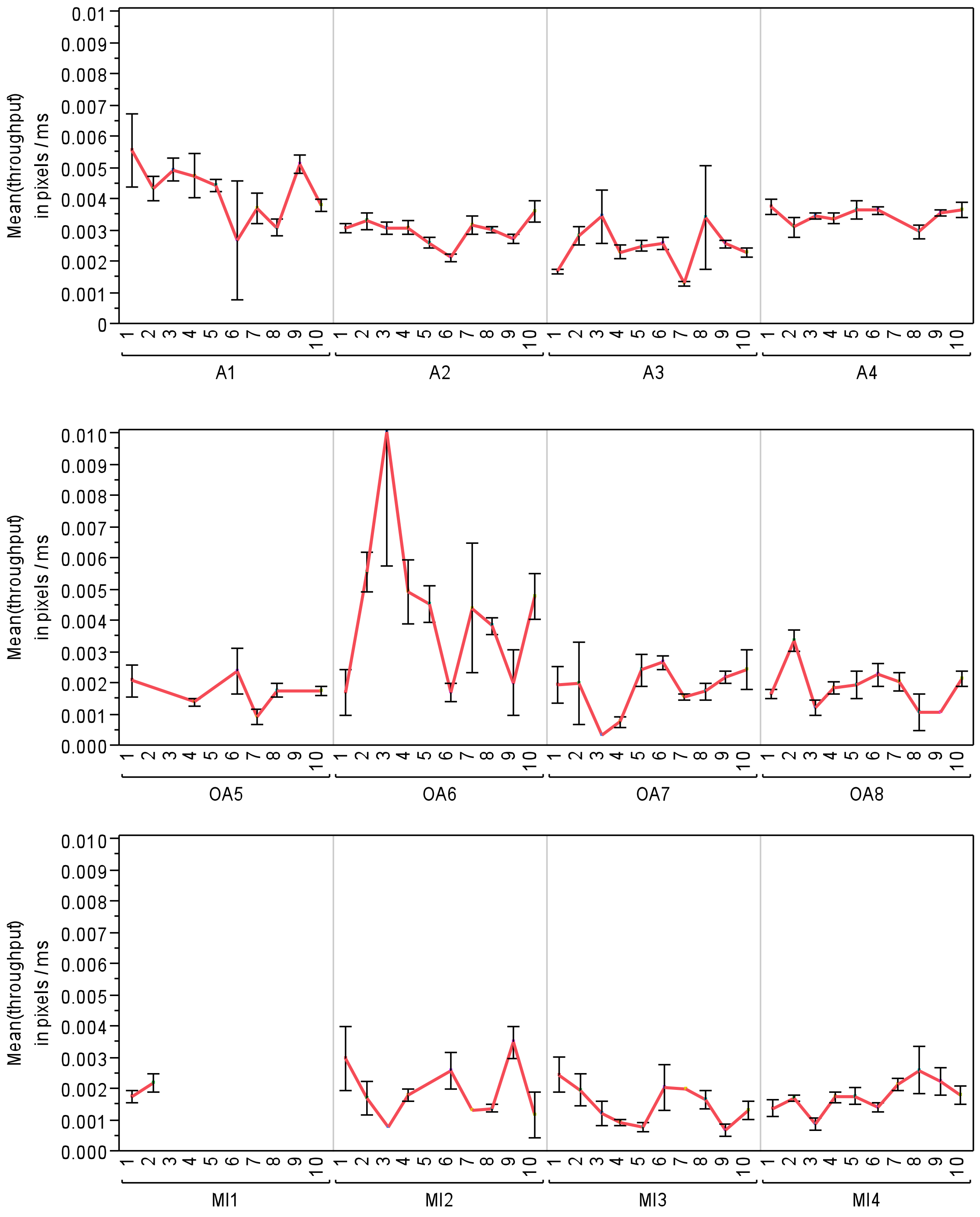


Figure 28 Mean throughput across ten randomly selected sessions. These charts are organized by group (A = able bodied, OA = older adult, MI = motor impaired) and within each group, participant and session number.

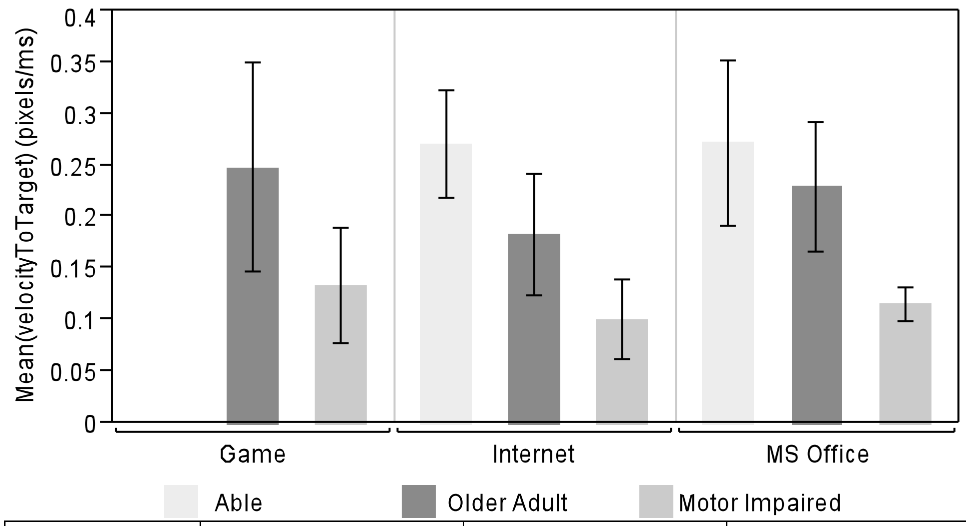
Design implication for assistive adaptations: Individuals who have a low throughput may benefit from adaptations that dynamically increase the control-display gain (ratio between the physical distance moved by the pointer and the effect on the pointer) to make their movements have a bigger impact (Balakrishnan 2004, Koester *et al.* 2005). As Koester showed, care must be made when adjusting this parameter since it can easily have negative as well as positive effects on performance. Alternatively, individuals who have low throughput may benefit from simply augmenting an interface to make the targets larger or closer together (Gajos *et al*. 2008).

5.2.4.4. Movement speed: velocity and peak velocity

Metric description and calculation: We present metrics about overall velocity and also look at where in the movement the peak velocity was reached. *Velocity* is calculated by dividing the Euclidean Distance moved by how long it took to make that movement (in milliseconds). *Peak Velocity* is the highest velocity a participant experienced between two mouse move events, we investigate both the absolute peak velocity in a given movement, and where in the movement the participant reaches this peak velocity.

Divisions of data: In our base statistics we compared velocity, peak velocity and relative time to peak velocity across Internet, MS Office and games. In our more detailed regression analysis, we use the log (base 10) of the velocity and peak velocity.

For both velocity and peak velocity we first present regression models across our three groups during Internet and MS Office use. We then present regression models investigating how game data differed from MS Office and Internet use for older adults and individuals with motor impairments. Finally, we investigate relative time to peak velocity during Internet and MS Office use.



	Games		Internet		MS Office	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
Able			0.270518	0.052330	0.271869	0.079846
Older Adult	0.248280	0.102295	0.182916	0.059150	0.229581	0.063095
Motor Impaired	0.133242	0.055645	0.100271	0.037953	0.115208	0.016822

Figure 29 Mean and standard deviation of velocity (pixels/ms) organized by group and application

	Velocity (simple)				Velocity (complex)			
Variable	**Estimate**	**Std Error**	**Prob>ltl**		**Estimate**	**Std Error**	**Prob>ltl**	
Intercept	-4.4737	0.0491	<.0001	***	-4.2530	0.0535	<.0001	***
Group: Older Adult	-0.0575	0.0650	0.3888		-0.0127	0.0667	0.8519	
Group: Motor Impaired	-0.1676	0.0701	0.0316	*	-0.2603	0.0724	0.0029	**
Application: Internet	-0.0253	0.0170	0.1362		-0.0171	0.0170	0.3144	
Older Adult X Internet	0.0202	0.0273	0.4577		0.0111	0.0272	0.6816	
Motor Impaired X Internet	-0.0112	0.0254	0.6587		0.0042	0.0254	0.8675	
Log10(Target Size)	0.0630	0.0045	<.0001	***	0.0529	0.0106	<.0001	***
Log10(Euclidean Distance)	1.1874	0.0037	<.0001	***	1.0630	0.0069	<.0001	***
Log10(Target Size) X Older Adult					-0.0146	0.0126	0.2796	
Log10(Target Size) X Motor Impaired					-0.0048	0.0199	0.8081	
Log10(Euclidean Distance) X Older Adult					0.0381	0.0086	<.0001	***
Log10(Euclidean Distance) X Motor Impaired					-0.1995	0.0125	<.0001	***
	R^2 = .67				**R^2 = .68**			

Table 14: Our simple model presents estimated velocity (log base 10 of pixels per millisecond) by group: older adult, motor impaired, able bodied (omitted), and application type: Internet or MS Office (omitted) and log10(Target Size) and log10(Euclidean Distance). Our complex model includes interactions between user group and Index of Difficulty (ID).
*p < .05, **p < .01, ***p < .001.

Regression results for velocity for Internet and MS Office use: We created a hierarchical linear model of velocity treating participant as a random effect. Table 14 presents the results of two of these models. Our simple model presents the main effects and interactions between user group (able bodied, motor impaired, and older adults) and application type (MS Office, Internet) and a main effect for target distance (log10 of Euclidean distance) and target size (log10 of smaller target size dimension) (R^2= .67). In this analysis, we have isolated the components of Index of Difficulty (distance and target size) so we can understand the effect of movement difficulty on velocity metrics.

Our simple model illustrates the following trends:

- Motor impaired participants had lower velocity than able bodied users ($p < .05$), and there was no significant difference in velocity between older adults and able bodied users ($p = .388$) across all application types, holding target size and distance constant.
- There was no main effect for application on velocity ($p = .136$).
- As one would expect based on Fitts' Law (Fitts 1954), there was a main effect for target size and distance on velocity (both $p < .001$) across all application types and groups. Velocity increased with target size and distance.

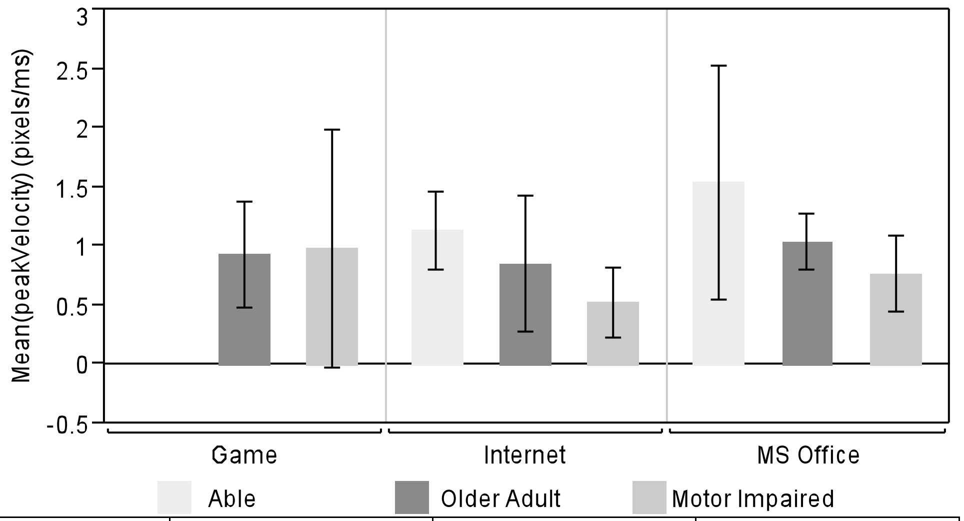
Our complex model adds an interaction between user group and target size, and user group and target distance (R^2 = .68).

- There was no significant interaction between target size and user group.
- There was a significant interaction between target distance and user group. Table 15 shows the estimated velocities using MS Office for a small target (10 pixels) when that target is 100 pixels (near) and 700 pixels (far) away.
    - o As target distance increases (holding target size constant), able bodied adults have a higher velocity than older adults ($p < .001$). For both groups, velocity increases as target distance increases.

- As target distance increases (holding target size constant), able bodied adults have a higher velocity than individuals with motor impairments (p < .001). However, for target distances less than 3 pixels, motor impaired participants have higher estimated velocity than able bodied participants. For both groups, velocity increases as target distance increases.

	Target Size = 10 pixels Distance: 100 pixels	**Target Size = 10 pixels Distance: 700 pixels**
Able	0.0162 pixels per millisecond (16.22 pixels per second)	.1758 pixels per millisecond (175.75 pixels per second)
Older Adults	.0084 pixels per millisecond (8.41 pixels per second)	.0717 pixels per millisecond (71.67 pixels per second)
Motor Impaired	.0049 pixels per millisecond (4.94 pixels per second)	.0265 pixels per millisecond (26.5 pixels per second)

Table 15 Estimated velocities using MS Office for a target whose size is 10 pixels, and is either 100 or 700 pixels away.



	Games		**Internet**		**MS Office**	
	Mean	**Std Dev**	**Mean**	**Std Dev**	**Mean**	**Std Dev**
Able			1.13536	0.325628	1.54139	0.992193
Older Adult	0.932058	0.44551	0.85402	0.579987	1.04040	0.234322
Motor Impaired	0.980048	1.01222	0.52889	0.298059	0.76626	0.323082

Figure 30 Mean and standard deviation of peak velocity (pixels/ms) organized by group and application

	Peak Velocity (simple)				Peak Velocity (complex)			
Variable	Estimate	Std Error	Prob>ltl		Estimate	Std Error	Prob>ltl	
Intercept	-1.4525	0.0335	<.0001	***	-1.4411	0.0334	<.0001	***
Group: Older Adult	-0.0201	0.0440	0.655		-0.0118	0.0430	0.7879	
Group: Motor Impaired	-0.0502	0.0488	0.3249		-0.0613	0.0477	0.2246	
Application: Internet	-0.0165	0.0080	0.0377	*	-0.0170	0.0080	0.0328	*
Older Adult X Internet	0.0010	0.0128	0.4482		0.0053	0.0128	0.68	
Motor Impaired X Internet	0.0113	0.0118	0.3404		0.0147	0.0119	0.2157	
Log10(Target Size)	0.0155	0.0021	<.0001	***	0.0147	0.0046	0.0016	**
Log10(Euclidean Distance)	0.5737	0.0009	<.0001	***	0.5660	0.0017	<.0001	***
Log10(Target Size) X Older Adult					0.0220	0.0055	<.0001	***
Log10(Target Size) X Motor Impaired					-0.0199	0.0087	0.0219	*
Log10(Euclidean Distance Traveled) X Older Adult					0.0003	0.0021	0.8969	
Log10(Euclidean Distance Traveled) X Motor Impaired					-0.0105	0.0031	0.0007	***
	$R^2 = .89$				$R^2 = .89$			

Table 16: Our simple model presents estimated peak velocity (log base 10 per millisecond) by group: older adult, motor impaired, able bodied (omitted), and application type: Internet or MS Office (omitted) and log10(Target Size) and log10(Euclidean Distance). Our complex model includes interactions between user group and Index of Difficulty (ID).
$^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$.

Regression results for peak velocity for Internet and MS Office use: We created a hierarchical linear model of peak velocity treating participant as a random effect. Table 16 presents two of these models. Our simple model presents the main effects and interactions between user group (able bodied, motor impaired, and older adults) application type (MS Office, Internet) and a main effect for the target distance (log10 of Euclidean distance) and target size (log10 of smaller target size dimension) ($R^2 = .89$).

Our simple model illustrates the following trends:

- There was no significant difference in peak velocity between older adults and able bodied users ($p = .655$) or between motor impaired adults and able bodied users across application types ($p = .325$) while holding target size and distance constant.
- There was a main effect for application on peak velocity ($p < .05$) across all groups. Peak velocity was slower for Internet use than for MS Office use.
- There was a main effect for target size and distance on peak velocity (both $p < .001$).

Our complex model adds an interaction between user group and target size, and user group and target distance ($R^2 = .89$). Table 17 shows the estimated peak velocities using MS Office for 10 and 100 pixel targets 100 pixels away.

- There was a significant interaction between user group and target size.
    - As target size increases (holding target distance constant), able bodied adults have a higher peak velocity than older adults ($p < .001$).
    - As target size increases (holding target distance constant), able bodied adults have a higher peak velocity than motor impaired participants ($p < .05$). Peak velocity for motor impaired participants decreases as target size increases, which we found to be an unexpected result.

	Target Size = 10 pixels Distance: 100 pixels	**Target Size = 100 pixels Distance: 100 pixels**
Able	.6594 pixels per millisecond (659.43 pixels per second)	.6789 pixels per millisecond (678.88 pixels per second)
Older Adults	.4914 pixels per millisecond (491.44 pixels per second)	.5347 pixels per millisecond (534.70 pixels per second)
Motor Impaired	.4544 pixels per millisecond (454.37 pixels per second)	.4489 pixels per millisecond (448.89 pixels per second)

Table 17 Estimated peak velocities using MS Office for a target whose size is either 10 or 100 pixels, and is 100 pixels away.

- There was a significant interaction between user group and target distance between able bodied and motor impaired individuals.
    - As target distance increases (holding target size constant), able bodied adults have higher peak velocity than individuals with motor impairments ($p < .001$).

	Velocity				**Peak Velocity**			
Variable	**Estimate**	**Std Error**	**Prob>ltl**		**Estimate**	**Std Error**	**Prob>ltl**	
Intercept	-1.8764	0.0343	<.0001	***	-1.4847	0.0308	<.0001	***
Group: Older Adult	0.0107	0.0343	0.7618		-0.0023	0.0308	0.9425	
Application MS Office	-0.0585	0.0094	<.0001	***	-0.0070	0.0108	0.5167	
Application: Internet	-0.0230	0.0049	<.0001	***	-0.0194	0.0056	0.0005	***
Older Adult X MS Office	-0.0127	0.0094	0.1765		-0.0038	0.0108	0.7264	
Older Adult X Internet	0.0056	0.0049	0.248		-0.0135	0.0056	0.0158	*
Log10(Euclidean Distance)	0.2166	0.0004	<.0001	***	0.2469	0.0003	<.0001	***
	R^2 = .58				**R^2 = .73**			

Table 18: This model presents estimated velocity and peak velocity (log base 10) in pixels per millisecond, by group: older adult and motor impaired (omitted), and application type: Internet, MS Office, or games (omitted). *p < .05, **p < .01, ***p < .001.

Regression results for velocity and peak velocity for game, Internet and MS Office use: We created two hierarchical linear models treating participant as a random effect. Table 18 presents the results of velocity and peak velocity models. This model presents the main effects between user group (motor impaired adults and older adults) and application type (MS Office, Internet, games) (velocity $R^2 = .54$, peak velocity $R^2 = .73$).

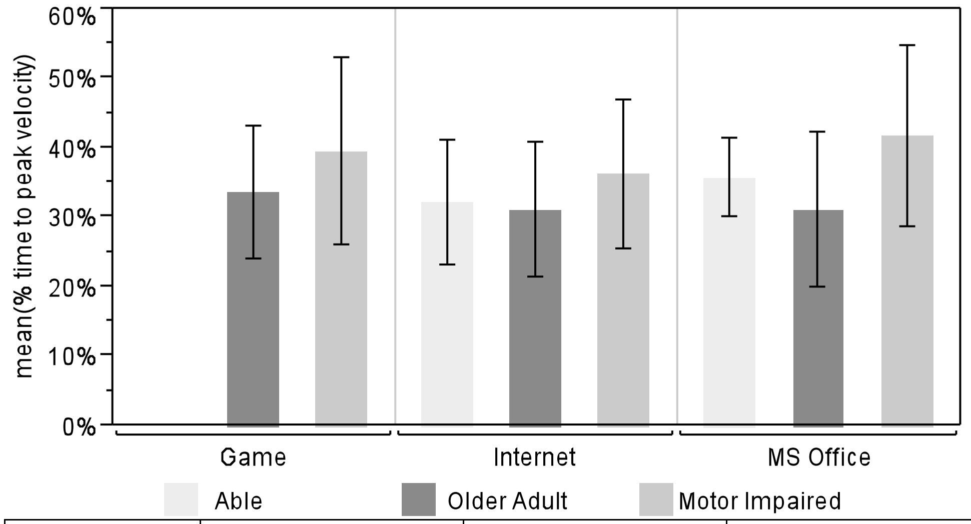
This model illustrates the following trends:

- There was no significant difference between motor impaired participants and older adults in velocity ($p = .762$) or peak velocity ($p = .9526$) across application types.
- Velocity was higher in games than during Internet ($p < .001$) or MS Office ($p < .001$) use, across both groups.
- Peak velocity was higher during game use than Internet use ($p < .001$), but peak velocity during game use was not statistically different from MS office use, across both groups.
- The only interaction effect we found between application type and group was that older adults had a significantly higher peak velocity than motor impaired users during Internet use ($p < .05$).
- There was a main effect for target distance on both velocity and peak velocity ($p < .001$) across all application types and both groups. Velocity and peak velocity increase with target distance.

Relative time to peak velocity: The relative time to reach peak velocity is another interesting movement metric to investigate. Pointing motions can be modeled as having two phases: a rapid ballistic motion towards the target, followed by one or more, typically overlapping, corrective motions. The second phase of motion consists of corrections to the inaccuracies the user finds in their initial ballistic movement towards the target. This phase of motion can itself be modeled as a series of smaller ballistic motions, but it is most typically done in an overlapped form which appears more as an extended tail on the initial ballistic motion, hence this phase is often called the *corrective tail*.

If the initial ballistic motion were perfect (with no corrective tail) then we would expect the peak velocity to be at the center of the motion (center of the bell shaped min-jerk curve) in both time and distance. Since the corrective motions are smaller and slower than the initial ballistic motion, the peak still occurs within that first part of the motion. As more correction occurs, the corrective tail becomes longer as a proportion of the overall motion. This stretches the motion out in time and proportionally makes the peak occur earlier in the overall motion. Hence, a proportionally earlier peak velocity corresponds to a proportionally longer corrective tail. (Grossman and Balakrishnan 2005; Meyer *et al.* 1990).

In our data, we compute relative time to peak velocity using the time when peak velocity was reached and the total time for the motion. Figure 31 illustrates the relative time to peak velocity across groups and organized by application. Figure 32 illustrates that peak velocity was reached earlier in the movement (or, equivalently, corrective tails were proportionally longer) as ID increased.



	Games		Internet		MS Office	
	Mean	**Std Dev**	**Mean**	**Std Dev**	**Mean**	**Std Dev**
Able			0.322302	0.089189	0.357881	0.056031
Older Adult	0.336674	0.095072	0.310913	0.097031	0.311174	0.112251
Motor Impaired	0.394898	0.134229	0.362295	0.106406	0.417937	0.130663

Figure 31 Mean and standard deviation of relative time to peak velocity (pixels/ms) organized by group and application.

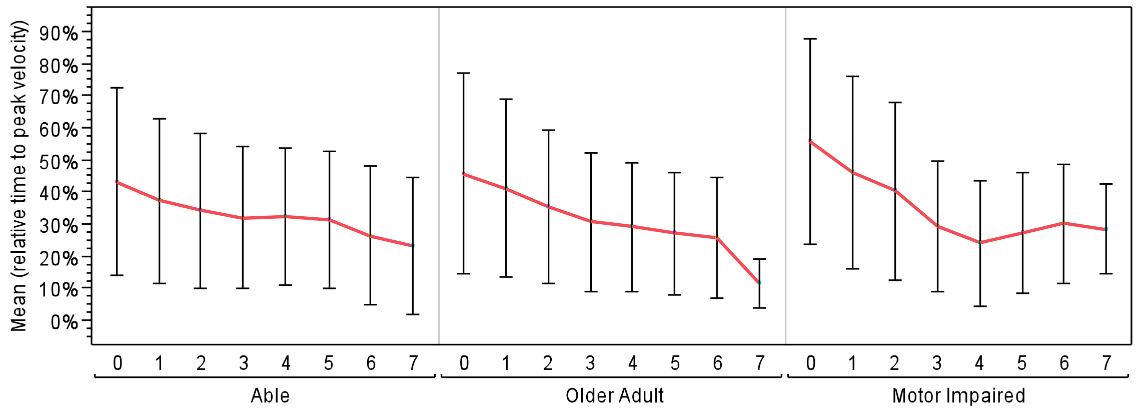


Figure 32 Chart of means and standard deviations for relative time to peak velocity (in percentage) by Index of Difficulty for Internet use. Smaller IDs represent targets that are easier to reach, while larger IDs are harder (farther away or smaller).

	Relative Time to Peak Velocity (simple)				Relative Time to Peak Velocity (complex)			
Variable	**Estimate**	**Std Error**	**Prob>ltl**		**Estimate**	**Std Error**	**Prob>ltl**	
Intercept	0.542	0.0104	<.0001	***	0.5496	0.0127	<.0001	***
Group: Older Adult	-0.0329	0.0145	0.0272	*	-0.0268	0.0142	0.0646	
Group: Motor Impaired	0.0455	0.0135	0.0023	**	0.0351	0.0133	0.013	*
Application: Internet	0.0003	0.0069	0.9693		-0.0006	0.0069	0.9315	
Older Adult X Internet	0.0212	0.0114	0.0624		0.0195	0.0114	0.0866	
Motor Impaired X Internet	-0.0235	0.0098	0.017	*	-0.0225	0.0098	0.0221	*
Log10(Target Size)	0.0148	0.0019	<.0001	***	0.0260	0.0045	<.0001	***
Log10(Euclidean Distance to Target)	-0.0969	0.0016	<.0001	***	-0.1117	0.0030	<.0001	***
Log10(Target Size) X Older Adult					0.0018	0.0054	0.7381	
Log10(Target Size) X Motor Impaired					0.0122	0.0085	0.1498	
Log10(Euclidean Distance Traveled) X Older Adult					0.0071	0.0037	0.0557	
Log10(Euclidean Distance Traveled) X Motor Impaired					-0.0253	0.0054	<.0001	*
	R^2 = .07				**R^2 = .07**			

Table 19: Our simple model presents estimated relative time to peak velocity by group: older adult, motor impaired, able bodied (omitted), and application type: Internet or MS Office (omitted) and log10(Target Size) and log10(Euclidean Distance). Our complex model includes interactions between user group and Index of Difficulty (ID). *p < .05, **p < .01, ***p < .001.

Regression results for relative time to peak velocity for Internet and MS Office use: We created a hierarchical linear model of relative time to peak velocity treating participant as a random effect. Table 19 presents the results of two of these models. Our simple model presents the main effects and interactions between user group (able bodied, motor impaired, and older adults) application type (MS Office, Internet) and a main effect for target distance (log10 of Euclidean distance) and target size (log10 of smaller target size dimension) ($R^2 = .07$).

Our simple model illustrates the following trends:

- We found that older adults reached peak velocity before able bodied adults, and motor impaired participants reached it last. There was a significant difference in relative time to peak velocity between older adults and able bodied users ($p < .05$) and between motor impaired adults and able bodied users across all application types ($p < .01$), while holding target distance and size constant.
- There was no main effect for application on relative time to peak velocity ($p = .97$) across all groups, while holding target distance and size constant.
    - However there was a significant interaction between able bodied and motor impaired relative time to peak velocity during Internet use, where able bodied participants reached peak velocity first ($p < .05$).
- There was a main effect for target size on relative time to peak velocity ($p < .001$) across all application types, groups, while holding target distance constant. As target size increases (and target distance is controlled for), relative time to peak velocity increases, indicating shorter corrective tails.
- There was a main effect for target distance on relative time to peak velocity ($p< .001$) across all application types and groups while holding target size constant. As target distance increases (and target size is held constant), relative time to peak velocity decreases, indicating longer relative corrective tails on far targets.

Our complex model adds an interaction between user group and target size, and user group and target distance ($R^2 = .07$). Table 20 shows the estimated relative time to peak velocity for small and large targets, and short and long distances.

- There was a significant interaction between target distance and user group, such that relative time to peak velocity decreases more for motor-impaired participants headed to farther away targets, than it does for able bodied participants headed to far away targets ($p < .001$).
- There was no significant interaction between target distance and user group.

	Target Size = 10 pixels Distance: 100 pixels	Target Size = 100 pixels Distance: 100 pixels	Target Size = 100 pixels Distance: 700 pixels
Able	34.9%	36.07%	28.18%
Older Adults	30.5%	33.28%	24.41%
Motor Impaired	40.4%	44.29%	32.72%

Table 20 estimated relative time to peak velocity using MS Office for a target whose size is either 10 or 100 pixels, and is 100 or 700 pixels away.

Summary of regression results: We found a main effect of target distance and target size on velocity, peak velocity, and relative time to peak velocity. For velocity, there was an interaction between group and the target distance, but not the target size. There was also an interaction between group and target distance and target size for peak velocity. This suggests that able bodied participants' velocity and peak velocity were significantly higher than motor impaired and older adult velocity when target distance was controlled for. However, when we controlled for target size we found that able bodied participants' peak velocity was significantly higher than the other two groups, while their velocity was not.

We found that older adults' relative time to peak velocity was lower than able bodied participants', and motor impaired participants had the highest relative time to peak velocity. Note that while we might expect the corrective tail component of able bodied motion to be proportionally smaller, indicating greater accuracy in their initial ballistic movement, this is not the case. Combined with the observed higher peak velocities, this might indicate that able bodied individuals are choosing higher velocity for their initial ballistic motion at the expense of lower precision, when compared to our other participant groups.

Comparison of results to related work: We found that our able bodied participants moved faster than any other group with a higher velocity and peak velocity. This finding is consistent with Ketcham's results (Ketcham *et al.* 2002), likewise higher peak velocity for able bodied participants than older adults or individuals with Parkinson's Disease is consistent with Keates' results (Keates and Trewin 2005).

When investigating relative time to peak velocity, Ketcham found that younger adults reached peak velocity proportionally earlier (after 45% of their movement, SD = 3.4) than older adults (peak velocity reached after 48.1% of movement, SD = 6) (Ketcham *et al.* 2002). In an evaluation of pointing by individuals with and without motor impairments, Hwang saw great variability in where peak velocity was reached across participants (Hwang *et al.* 2004). In a laboratory task, Keates and Trewin found that adults reached peak velocity after 19.86% (SD = 10.70), older adults reached it after 26.17% of their movement (SD = 16.98) and individuals with Parkinson's Disease reached it after 25.25% of their movement (SD = 25.25) (Keates and Trewin 2005). Our finding that able bodied adults reached peak velocity before motor impaired participants is consistent with this related work, but our finding that older adults reached peak velocity first (out of all three groups) is surprising. We do not know why this happened in our data, but feel it is an important question to investigate in future work.

Our findings that the relative time to peak velocity decreases (or corrective tail increases) with ID is consistent with findings from Ketcham (Ketcham *et al.* 2002).

Design implication for assistive adaptations: As with throughput, participants who experience low velocity or peak velocity may benefit from adaptations that increase the cursor gain. Given that we found significant differences for older adults and motor impaired participants' velocity and peak velocity across applications, these adaptations should likely be application specific. Individuals who reach peak velocity early due to a long corrective tail may be experiencing targeting problems. If this is the case, they may benefit from one of the targeting adaptations we have already discussed. However, the key to successful adaptation is to determine why the corrective tail is happening early or late in the movement.

5.2.5. Summary: Laptop Study Performance

In our analysis of laptop data we focused on differences across our three groups of participants. While this is a relatively simple way to analyze the data, we found some dramatic trends in pointing performance across these groups. We used the following trends in our data for our automatic assessment of pointing performance using this data (Chapter 6.3).

5.2.5.1. Individuals with motor impairments had most targeting and movement problems

Not surprisingly, in our analysis we consistently observed that our participants with motor impairments had more targeting and movement performance problems than the other two populations, and able bodied participants had the fewest problems. This confirms our impression that this demographic could benefit from automatic assessment and adaptive systems.

5.2.5.2. Able bodied participants moved the fastest, yet didn't have the longest relative time to peak velocity.

Able bodied participants consistently had the highest overall velocity and peak velocity of our three groups, while our participants with motor impairments were the slowest. This also held true when we accounted for target size and Euclidean distance traveled. However, we observed that older adults reached peak velocity first, and motor impairments reached it last.

5.2.5.3. Older adults and motor impaired participants have "better" performance during games than in Internet or MS Office

We saw statistically significant better performance for both click duration, peak velocity and velocity older adults and motor impaired participants during games than during Internet or MS Office use. Across these two groups there was faster velocity, faster peak velocity, and shorter click durations while playing games than while using MS Office or the Internet. Improved performance during games could be due to high motivation to complete the task, practice in selecting these targets (since pointing tasks in games can be very repetitive), or confidence in knowing what to do next.

6. Assessing Performance with Predictive Models

This chapter demonstrates that pointing performance can be frequently and unobtrusively assessed with predictive statistical models constructed with machine learning techniques. Assessment (or classification) can be done by building learned statistical models of distinct categories of performance found during real world use. With high accuracy, we were able to distinguish between performance by older adults and individuals with disabilities from performance by individuals without. We were also able to predict with high accuracy when current performance would be followed by problematic performance.

In this chapter we will first discuss our success building three learned statistical models that are able to distinguish between different groups of individuals having pointing problems during a single laboratory study. We next discuss how our investigation of real world data has shown that pointing performance from a single laboratory session may not be sufficient to fully understand a particular user's pointing ability. Instead multiple sessions of real world use need to be evaluated. In the third section of this chapter, we present our results investigating pointing performance across multiple sessions of use while using Microsoft Word and/or while browsing the Internet from our laptop deployment that included able bodied individuals, older adults, and individuals with motor impairments.

6.1. Classifying Pointing Behavior During Pointing Performane during a Single Session of Laboratory Use

To test the feasibility of building learned statistical models of pointing performance to detect pointing problems (going beyond the scope of the work in Chapter 3) we started with laboratory data. In this section we will describe learned statistical models we constructed from three datasets to distinguish between pointing performance in different populations. All three of these datasets were collected in independent laboratory studies (Koester *et al.* 2005), (Keates and Trewin 2005), and (Trewin *et al.* 2006) and reanalyzed here to test the feasibility of creating predictive models, as described in (Hurst *et al.* 2008A). Participants in all of these studies performed Fitts' law style pointing tasks where they were told to move a mouse to a specific target and click on it.

Machine learning techniques were employed to build models to distinguish between different levels of performance. Each dataset was segmented into a set of movement trials that served as training instances. All three of the datasets we analyzed define an instance as mouse movement followed by a click on the target. Each such instance is labeled with an indication of the properties of the user who performed it, such as if they had a motor impairment or not.

All of our predictive models in this chapter used C4.5 decision trees that were validated with a 10 fold cross-validation test on wrapper-selected features in Weka. The features used to build the models are organized into four categories in all three datasets. These categories are features calculated from the movement that are specific to the task, features that describe what happened during the click, features that describe the pointer's motion, and features that describe pauses in the pointer's motion.

6.1.1. Distinguishing between Movement Behaviors of People With Pointing Difficulties

Our first analysis focused on performance differences between people who have pointing and clicking difficulties and those who do not. We used the dataset described in (Koester *et al.* 2005) which was gathered from individuals who had physical impairments that affected their ability to use a mouse. To complement this dataset, we collected a corresponding new dataset from students at our university. The combined dataset included data from 33 participants (21 able bodied, 17 female). The diagnoses in the motor impaired group varied included the following conditions {6 Spinal Cord Injury (2 C4/5, 2 C5/6, 1 C7, 1 unknown), 1 Traumatic Brain Injury, 2 Cerebral Palsy, 1 Friedrich's Ataxia, 1 Multiple Sclerosis, 1 Muscular Dystrophy}. All but two of the motor impaired participants completed these trials using a standard mouse, and the other two used a trackballs.

In both the original study and the companion data collection we performed, participants completed Fitts' Law-style pointing tasks using the IDA Software Suite (Koester *et al.* 2005). IDA is a software tool to assess an individual's ability to access a computer based on performance on a range of computer skills tasks. To evaluate pointing performance, all participants completed 32 trials of a pointing task with at least 10 different mouse gain settings. Each trial presented the user with a square box that varied in size and distance from the box in a previous trial. All participants started the tasks with a mouse gain of 10, with Enhanced Pointer Precision set to “on". After each block of 32 targets at a given gain setting, IDA would calculate the user’s performance using an algorithm described in (Koester *et al.* 2005) and would then try another gain setting until it had enough data to predict the “best” one.

6.1.1.1. Features Available for this Dataset

<u>Task Specific Features</u>

- Did the user correctly select the target?
- How long did it take to finish the trial?
- How many times did the cursor enter the target?
- Maximum distance traveled beyond the target, or “Overshoot”

<u>Features Related to the Click</u>

- Count of “Missed” or accidental clicks

<u>Features Related to Movement</u>

- Deceleration time, or how long it took to move from peak velocity to maximum displacement, divided by total movement duration
- Total distance traveled during trial
- Mean instantaneous velocity during initial movement towards target
- Number of direction changes

<u>Pause Features</u>

- Time spent before first movement of the trial, or “Reaction Time”

6.1.1.2. Predictive Model Results

Using decision trees with random 10 fold cross validation and wrapper selected features, we were able to distinguish between pointing behaviors from individuals with pointing problems vs. individuals without for the combined Koester dataset with 92.7% accuracy (Kappa = .85) as estimated by per-person hold out. Since many separate models were constructed (one for each person held out) slightly different feature sets were selected for each of these models. All of the individual feature selection runs selected these two features as predictive: the total time it took to complete the action and the number of clicks that occurred during the action.

We also conducted analysis with decision trees and random 10-fold cross validation and wrapper selected features which correctly classified the test instances with 94.5% accuracy (Kappa = .89). This feature selection also chose the following features as predictive: the same two selected during the individual model runs (the total time to complete the action, number of clicks that occurred during the action), and in addition, the number of times the cursor entered the target.

6.1.2. Distinguishing Between Movement Behaviors of Young Adults, Adults, Older Adults and Individuals with Parkinson's Disease

The previous section indicated that statistical models are capable of identifying user's ease of pointing with high accuracy. While it is an important and valuable first step to distinguish between groups with and without pointing difficulties, this classification may be too broad to make many accommodations. Instead, we may want to be able to make a finer distinction based on what kinds of errors those individuals are making. Unfortunately, the dataset discussed in the previous section does not have enough examples of particular types of motor impaired performance to confidently distinguish multiple classes of performance. To address this limitation, we looked at another dataset from four groups of users, three of which have significantly different performance abilities (Keates and Trewin 2005).

This data was gathered in a study that examined the effects of age and Parkinson's Disease on a point-and-click task using a mouse. It includes pointing performance from the following four groups: Young Adults (8 participants, ages 20-30), Adults (8 participants, ages 35-65), Older Adults (7 participants, ages 70 and older), and individuals with Parkinson's Disease (6 participants, ages 48-63). A more detailed summary of this population is described in their paper (Keates and Trewin 2005).

6.1.2.1. Features Available for this Dataset

This data set provided detailed recordings of point-and-click task performances, allowing a more sophisticated set of features to be employed. This feature set differs from the previous one because it has more features describing when the pauses occurred, as well as features related to acceleration and velocity changes. Having these additional features enabled us to build a more detailed picture of the difference between groups.

Task Specific Features

- Total time trial time
- Number of times the cursor entered the target

Features Related to the Click

- Length of click
- Distance and angle moved during the click, or "slips"
- Time between mouse down event and preceding movement
- Count of "missed" or accidental clicks

Features Related to Movement

- Average and peak velocity and acceleration during the movement phase of the trial.
- Number of direction changes
- Total distance traveled during trial
- Movement error, offset and variability

Pause Features

- Count of the number of pauses of different lengths (from 0 milliseconds to 2500+ milliseconds) during the trial

6.1.2.2. Predictive Model Results

Two-way classification: We were able to reproduce the high accuracy of distinguishing motor impaired from able-bodied use in this dataset even though it involved a slightly different task and feature set. In a two-way classification between a group of adults and young adults versus a group of older adults and Parkinson's individuals a learned statistical model gave a classification accuracy of 94.6% (Kappa = .89) using a decision tree and validated with random 10-fold cross validation and using wrapper selected features.

Labels	Classification Accuracy	Kappa
A, P	97.6	.95
A, OA	93.8	.87
A, YA	59.3	.19
P, OA	91.4	.83
P, YA	96.7	.93
OA, YA	93.3	.86

Table 21 Classification results for all pairings {YA = Young Adult, A = adult, OA = Older Adult, P = Individual with Parkinson's Disease}

In order to further understand how well learned statistical models behaved on this dataset, we built models to distinguish between each pairing of the groups. We used features selected with a wrapper based feature selection to build a statistical model using a decision tree and random 10-fold cross validation for each paring of the groups (Table 21). These models found the highest accuracy when distinguishing between the Adult and Parkinson's (96.7%, Kappa = .95), and the lowest accuracy at distinguishing between the Adult and Young Adults (59.3%, Kappa = .19). The low accuracy at distinguishing between the young adult and adult groups suggests (not surprisingly) that these groups perform very similarly and are hard to distinguish.

Three-way classifications: We were able to build a learned statistical model with a decision tree and wrapper-based feature selection that performed with 91.6% accuracy (Kappa = .85) as validated with random 10-fold cross validation. Table 22 shows the results of other possible three-way pairings.

Labels	Classification Accuracy	Kappa Statistic
YA + A, P, OA	91.6	.85
A, P, OA	89.7	.84
YA, P, OA	98.9	.85

Table 22 Performance of statistical models using 3-way analysis {YA = Young Adult, A = Adult, OA = Older Adult, P = Individual with Parkinson's Disease}

	Classified as		
Actual	**YA + A**	**OA**	**P**
YA + A	977	17	8
OA	105	533	207
P	182	306	439
	Accuracy: 74.1% Prior 36.1%		

Table 23 Per-person holdout confusion matrix {YA = Young Adult, A = Adult, OA = Older Adult, P = Individual with Parkinson's Disease}

Unfortunately, this level of accuracy did not appear for this dataset when we employed per-person holdout. We conducted a three-way classification of this data using per-person holdout using 7 randomly selected young adults or adults, the 7 older adults, and the 6 individuals with Parkinson's Disease as testing data. Predictions were made with a decision tree using wrapper selected features, and validated with 10-fold cross validation. The models were able to predict the test participant's class with 74.1% accuracy (Kappa = .58). The performance of this model is probably not as high as with the random holdout because there was high variability in the data due to the small number of participants in each group, and variability in performance differences between a few of the older adults and individuals with Parkinson's Disease (Table 23).

Four-way classification: We were able to use decisions tree with random 10-fold cross validation and wrapper selected features to distinguish between the four groups with 70.0% accuracy (Kappa = .59). Not surprisingly, analysis of the confusion matrix (Table 24) for this classification problem shows that the classifier had the most difficulty distinguishing between the young adult and adult groups.

	Classified as			
Actual	**YA**	**A**	**OA**	**P**
YA	711	352	38	8
A	498	599	53	18
OA	58	48	755	61
P	22	13	45	770
	Accuracy: 70.0% Prior: 28.8%			

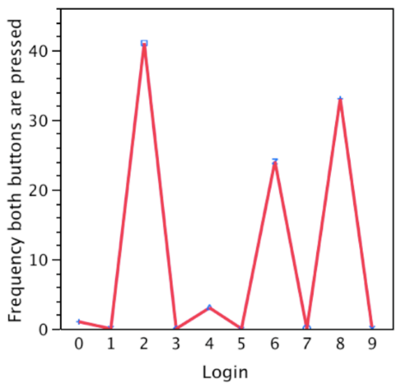
Table 24 Random 10-fold 4-way prediction {YA = Young Adult, A = adult, OA = Older Adult, P = Individual with Parkinson's Disease}. Note the confusion between Adult and Young Adults.

The wrapper based feature selection selected the following features for all pairs: length of the click, a count of amount of continuous movement, and a count of pauses between 1500 and 2000 milliseconds before pressing the target.

6.2. High Variability in Pointing Performance Confirms Need for Frequent Assessments

In the previous section we described models we were able to build that could accurately distinguish between different groups of pointing performance that could be used for automatic assessments. However, one key question for the design of an assessment system is how frequently assessments should be made. To explore this question, we analyzed data from our desktop deployment of real world pointing performance to see how performance changed across sessions. We analyzed data from six individuals with pointing problems (4 female, mean age 40.5 years) who used our computers between 3 and 120 sessions. We compared pointing performance in a controlled pointing task to their pointing performance during multiple sessions of real world use. This comparison helps illustrate how much an individual's performance can vary and points to the need to assess performance at least once a session.

Across all participants, there was high variability across and within sessions. Levene's test, a test for homogeneity of variance, found unequal variance across all measures ($p < .001$) across sessions and across participants ($p < .001$). Participants experienced an unpredictably wide range of performance across sessions. This extreme variability is visible in our data: there are several instances where performance during baseline collection (done as a laboratory study) differed dramatically from real world use. For example, Figure 33 illustrates how one participant did not have a problem with overlapping button presses during the baseline task, but did several times during real world sessions. There was no significant correlation between the session number and performance for any participant across any of the measures (Desktop deployment measures are summarized in Section 5.1).



Figures 33 This graph shows the frequency of one user accidentally pressing both buttons on a mouse, when they should only press one, by login session. Note the variation between login 0 (laboratory session), and real world use (login 1-9).

In addition to high variance across sessions, there was also high variation within sessions, as evidenced by the wide error bars in Figures 13 and 14 (Section 5.1.3), especially prominent on sessions with worse performance. Variance was correlated among three of our measures, providing evidence that in bad sessions, users experienced multiple types of difficulties. For a given participant, a cross correlation showed that the variance of distance slipped is highly correlated with the variance of the number of direction changes in the Y direction (r = .90, p < .05) and the variance of the excess distance travelled between targets (r = .82, p < .05). Other pairings of our measures were not significantly correlated.

Given the high variance we saw amongst individuals during real world use, even within sessions, one interpretation of our analysis is that any system that will assess and adapt to pointing problems must frequently assess use (i.e. at least once during each session, and ideally more often). In the following section we describe our work on building predictive models that can assess real world pointing performance across multiple sessions of use. To improve computer access, these classifiers should be utilized in a system as frequently as possible to assess performance, to enable just-in-time intervention.

6.3. Classifying Pointing Behavior During Multiple Sessions of Real World Use

This section describes our success at classifying pointing behaviors using real world pointing data that was collected in our laptop deployment (described in Chapter 4 and 5) with 12 participants: 4 able bodied, 8 older adults, and 4 individuals with motor impairments. We will first describe the data we used to classify performance; next we will describe what metrics or features we used to classify performance; and finally we will describe our two-level classifier that distinguishes between performance groups with high accuracy.

6.3.1. Data Subsets and Labels

This analysis uses data collected in our laptop deployment (described in Chapter 4 and 5). We collected over 400,000 samples from a wide range of applications (see section 5.2.1.1 for a detailed list of the applications we gathered), but we only used a subset of this data to build performance models because we wanted to ensure the generality of our performance models by using data from applications that were used by almost all of our participants. The two applications most commonly used were web browsers and the Microsoft Office Suite. This subset of data was about 80,000 samples (with about 2,500 from MS Office and the remainder from web browsing).

The distribution of data across our three groups was very skewed for this subset of our data (able bodied participants had almost 3-times as much data as any other group). This skew is because the able group spent most of their time on the Internet, while most of the older adults spent their time playing games. In order to have a more evenly distributed dataset, we combined the data from older adults and individuals with motor impairments (accounting for 30% of the data). Our model building on real world data makes a two-way classification between able bodied participants and our other participants, which is very similar to the work we did in Section 6.1.

6.3.2. Wrapper Selected Features on MS Office Data

We generated 77 features that describe our real world data, using both segmentation techniques (by click only, and with click and pauses). This set includes movement, clicking, targeting, keyboard, and window event features. Since not all features necessarily contribute to a good classifier, we used wrapper-based feature selection. The following features were selected from the dataset of MS Office use:

- Width and height of target selected
- Duration of click
- Count of X direction changes in entire sample (when segmented by click)
- Count of X direction changes in segmented sample (when segmented by pause)
- Efficiency of movement in entire sample (when segmented by click)
- Count of times “tab” key was pressed during between clicks (these events were most frequently the user switching between applications).

The following features were selected from the dataset of Internet use:

- Width and height of target selected
- Duration of click
- Interactor type
- Time between clicks
- Pause time between clicks
- Frequency of pressing too many buttons
- Count of times arrow keys were pressed
- Count of Minimize/Maximize window events fired

Of the features chosen, only target size and click duration were selected in both datasets. Of the differing features between data sets, the features selected from MS office measure efficiency (minimizing unnecessary movement) and the number of times the "tab" key was pressed, while features from Internet data tended to measure timing, frequency of pressing too many buttons, frequency of using arrow keys, and maximizing and minimizing windows.

6.3.3. Two Level Classifier Results

We used a two level classifier to predict if a particular motion was performed by a participant in either the able bodied group, or the group that combined older adults with individuals with motor impairments. The first level is a two-way decision tree that classified each motion as belonging to either the able bodied group or the group which combined older adults and individuals with motor impairments. In the MS office data, our tree correctly classified 90.15% of the data (kappa = .71). For the Internet only data, our tree correctly classified 92.04% of the data (kappa = .80). We also built a model on a combined dataset using the union of the selected features from each application. We tested this model against the combined dataset it correctly classified 91.94% of the data (kappa = .80).

These results prove that it is possible to distinguish between these two groups with high accuracy by looking at a single pointing action. However, as we have already shown, there can be a lot of variance in real world data. Our second model minimizes the effect of a single action by looking at a group (or window) of consecutive actions and uses those to generate a prediction.

The second level of our classifier aggregated successive predictions over several movements, this is known as a "sliding window". We aggregated predictions from a variety of window sizes (1, 5,10, 20, 40, 80,160, 320 samples) and used a voting model to classify predictions of able-bodied vs. OA or MI use. As we increased the window size, we got better estimates. Given that we are studying real world use, it is relatively easy for us to collect dozens or hundreds of examples of use for these windows. Note that as we increase our window size, we require more samples before we can make our first prediction. Additionally, there is a diminishing returns effect where adding more samples provides a smaller improvement to accuracy.

Figure 34 shows the accuracy of this second level classifier. In the MS Office dataset, we quickly improved our accuracy from 90% on a per-movement classification to 100% accuracy with a window of size 40, a 10% increase. In both the Internet and combined datasets we improved performance from 92% to 99% with a window of 320 samples. In all three datasets, we saw substantial improvements using a window size of 5 (5% in MS Office, and 3% in Internet and the combined dataset).

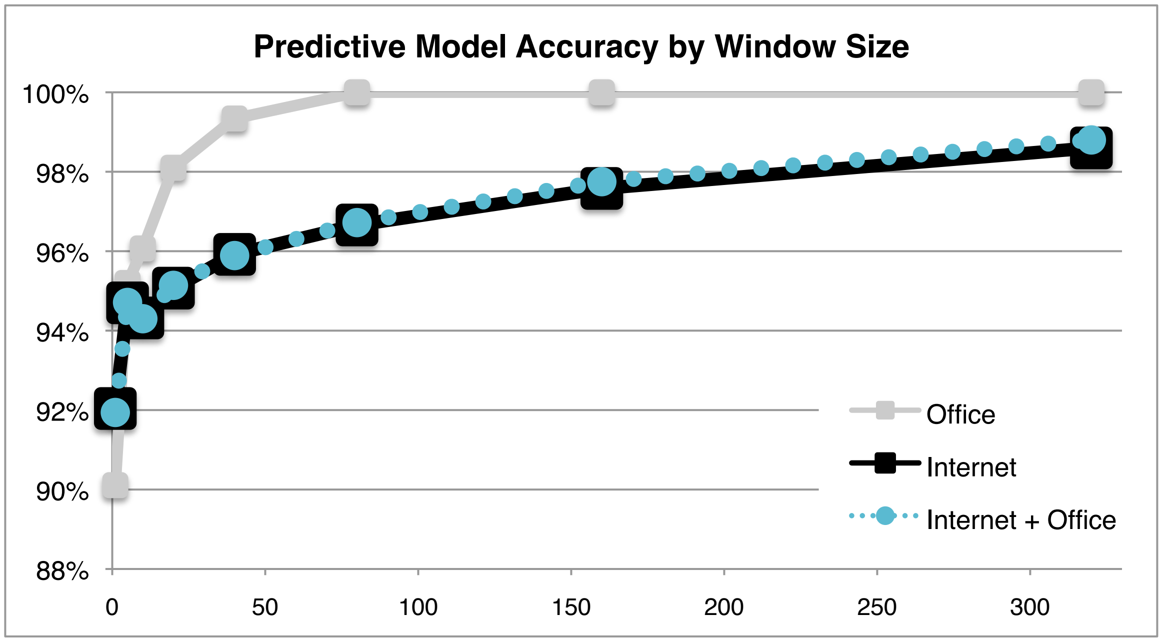


Figure 34 Plot of accuracies of 2nd level model by window size for Internet use, MS office use, and use from both applications.

6.3.4. Assessing Real World Pointing Performance

The results from our two level model show that it is possible to accurately distinguish between these two groups of pointing performance with high accuracy. We believe that these models could be easily used in an assistive adaptation to support individuals who experience pointing problems. When designing these adaptive systems, application designers will need to evaluate the tradeoffs between variability, accuracy, and number of predictions required to make a prediction when configuring these models. Applications that need frequent predictions (and can handle high variability) may be best served with our first level models, but applications that need less variability and higher accuracies may be better with our second level models and large window sizes.

6.4. Summary Assessing Performance with Predictive Models

This chapter described how we assessed pointing performance using predictive models. The first section described our success at distinguishing between performance from individuals with and without motor impairments, and then our results distinguishing between young adults, adults, older adults, and individuals with Parkinson's Disease using one session of laboratory data. In the next section we presented findings from our desktop deployment that individuals with motor impairments can have highly variable pointing performance during real world tasks. Based on this variability, we argued that these assessments should be made on real world data (which has higher variability). In the last section of this chapter we discussed our ability to successfully assess performance during real world Internet and MS Office use. The work described in this section proves that it is possible to make performance assessments on both real world and laboratory pointing performance.

7. Improvements to Our Automatic Assessment Techniques

The previous two chapters have described how we have analyze real world pointing data to learn about trends in performance (Chapter 5) and how we have used predictive models to identify information about the individual performing the action. While these results are novel and exciting, we were interested in investigating additional techniques that could improve automatic assessment. In the following chapter we describe two unrelated techniques that we have developed that can improve automatic assessment.

The first section describes an approach to automatic target identification that combines the Microsoft Active Accessibility (MSAA) API with computer vision and predictive models, in a hybrid technique that can recognize more targets a user selected than the MSAA API can alone (Hurst *et al.* 2010). In the second section, we take our predictive model work further than only detecting pointing problems by building a model that predicts whether or not a participant would benefit from a specific adaptation (Hurst *et al.* 2008A).

7.1. Improving Target Identification

Knowledge of the targets a user interacts with can be used for many potential applications. In our case, this information can be useful in the automatic assessment of a user's movement performance and the deployment of appropriate software to improve this performance. Access to the size and location of targets could also be used for usability evaluations to analyze sequences of buttons used in an application, similar to the analysis described by (Hilbert and Redmiles 2000). A history of target interactions could also be used to create assistance software that automatically creates or suggests shortcuts or macros to automate a sequence of tasks.

We are interested in automatic solutions to target identification since most real world computer use involves interacting with many targets, and it would be unrealistic to hand label targets in these extremely large datasets, or worse, to ask the user to do it. In the previous chapter, we used a software hook to the interface or application through an Accessibility Programming Interface (API) to get target information. Accessibility APIs can be used to automatically identify some of the targets in a graphical user interface. Specifically the Microsoft Accessibility API (or MSAA API) gives programmers access to many application and Windows operating system events and interactors. This API can be used to get the size and type of many interactors such as push buttons, menus, and icons.

Unfortunately, this API only provides access to some of the interactors a user may encounter during real world use. Variations in toolkits, cross-application compatibility (such as web content presented in different web browsers), legacy as well as custom code, and new interactors are some of the reasons not all targets are supported. For example, the MSAA API does not support many specialty interactors including the drawing area of Microsoft PowerPoint, pre-loaded Windows games, and specialty dialogs including the Character Map, custom color selector panels, and the Microsoft Paint controls.

An additional drawback to relying only on accessibility APIs for automatic target identification is that not all applications are equally supported. For example, two popular web browsers, Microsoft's Internet Explorer and open source Firefox, treat content in a very different way, limiting the API's access to them. Internet Explorer treats embedded Flash applets in such a way that the API is able to access most of the targets; however these targets are not accessible through the API in Firefox. Additionally, the AJAX web development techniques used by many of the Google widgets (i.e. Google Mail, Google Maps, and Google Calendar) is supported by the API in some browsers (e.g. Firefox), but not all (e.g. Internet Explorer). In our real world dataset discussed in chapters 4 and 5, the MSAA API only covered 68.5% of all the targets we collected during Internet use, 39.5% of targets in MS office, and only 2% of targets in Games (our most popular application type).

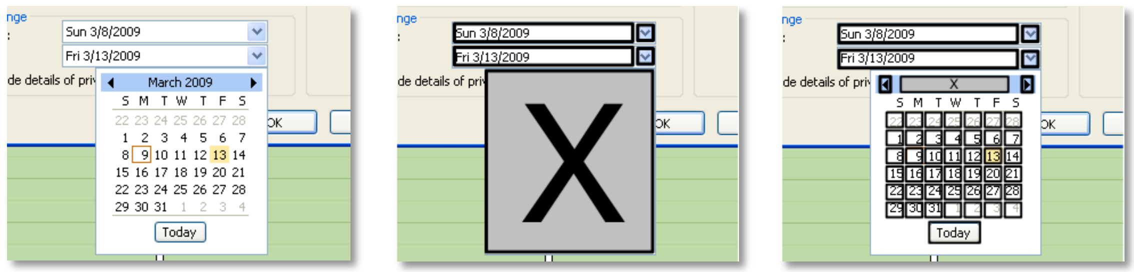


Figure 35 Our hybrid technique is able to identify significantly more targets than the Accessibility API alone. **A)** A screenshot of an open dropdown calendar object in Microsoft Outlook 2003. Interactive targets visible in **(A)** include: two textboxes (with dates); two dropdown handles next to the textboxes; arrows on either side of the month; the month and year; any of the dates in the calendar; and the Today button. **B)** The Microsoft Accessibility API found 4 targets correctly (shown with a framed rectangle) and 46 targets incorrectly (shown with a filled gray rectangle). **C)** Our hybrid technique found all but one target correctly.

In order to increase the number of real world targets whose size we know, we developed a novel technique to automatically identify most of the targets a user would encounter during real world use. Our solution leverages information from the MSAA API combined with one of three techniques that leveraged visual information interaction with a GUI. This set of four techniques complement each other as each was chosen to detect certain types of targets. Since the techniques may yield multiple target hypotheses, we use a two-level classifier to choose one hypothesis to use as the predicted target. This technique currently runs offline from data collected by our logging software CRUMBS (Section 4.3), but future work for this component is to update it so it can run immediately after target acquisition.

This section first discusses accessibility APIs and their limitations for automatic target identification during real world use. The following section describes our technique that leverages visual cues from interaction in a GUI to detect the targets a user interacted with. These visual cues are analyzed using computer vision techniques to identify targets. One of these techniques leverages the visual cues from interaction (e.g., change in button color) by subtracting the image captured during the button press from the image captured during the button release to determine the correct size and location of the target.

We conclude this section by describing how we leveraged predictive models to combine the information available from the vision techniques and the information from the MSAA

API to estimate the size and location of the target. The current implementation of this technique correctly identified 89% of the targets in a real world dataset where the MSAA API only correctly identified 74% of the targets. Figure 35 illustrates the representative differences in target accuracy that we were able to achieve in one panel in Microsoft Outlook using our hybrid technique (Figure 35C), compared to the performance of the MSAA alone (Figure 35B).

7.1.1. Leveraging Visual Cues To Automatically Identify Additional Targets with Computer Vision

We use three computer vision techniques that process captured screen images taken before and after a button press to find the target a user interacted with. These computer vision techniques use the full-screen screenshots from CRUMBS that were collected on the pointer button up and down events. *Difference Images* are used to detect the visual change caused by a button press. *Template Matching* (Forsyth and Ponce 2002) is used to detect interactors with similar borders such as buttons, boxes and even playing cards in a game. *Color Matching* is used to find selected items.

The template matching, difference image, and color matching techniques are written in JAVA and use the JAVA Advanced Image Library (JAI Library https://jaistuff.dev.java.net). Our implementation currently runs offline, but could easily be augmented to run as the data was collected.

7.1.1.1. Comparing the Difference Image

In the difference image technique, targets that produce a salient visual change upon interaction are detected by calculating the difference image before and after the click. This technique uses the images captured before the button press and after to identify targets by calculating and analyzing the difference image between those two images (Figure 36).

Image Preparation. To minimize difference image processing time, we automatically crop full screen images collected by CRUMBS to a smaller size. Since it is common for users to *slip*, or accidentally move the pointing device a few pixels during the click, we cannot simply crop images to a certain size around the location of the cursor. Instead we

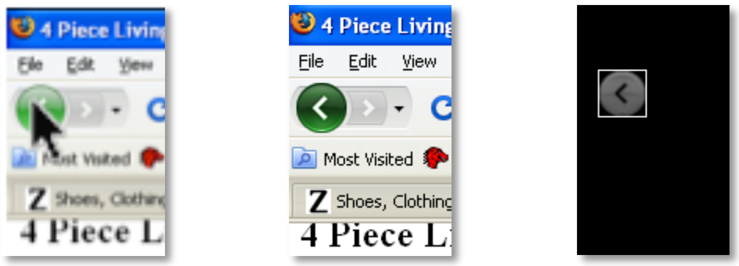


Figure 36 A) The user is pressing on the "Back" button in Firefox (the cursor here is for illustration only and not captured in our data). **B)** The image captured after the user releases the locator. **C)** The bounding box around the resulting difference is the correct target.

must align the images taken during the button down and up events according to the length of the slip. We align these images by calculating the difference between the capture coordinates for the images taken during both events, and crop any overlap to produce new images that are the same size. Image preparation is done with PERL scripts that call the ImageMagick software suite (ImageMagick www.imagemagick.org).

After we have identified the size of the difference image, and cropped the two input images, a Gaussian blur is applied to both images. Applying this blur helps minimize the likelihood of very small differences being incorrectly identified as several discrete targets when they are actually part of the same target. This additional step is most helpful when the user clicked on text, and helps the connected component algorithm (below) find words instead of single characters.

After the images have been prepared, we iterate through each image and literally subtract the pixel values from one image to another at the corresponding locations. This subtraction yields an output image with only the differences between the two images. This difference image is then analyzed to identify the size of any potential targets. Since not all image pairs have only one visual change, we perform an additional step to account for multiple visual changes, as illustrated in Figure 37. These changes are frequently caused when tooltips are displayed, interactors lose their focus, or visual changes unrelated to the pointing device occur. We handle these cases by analyzing the difference image with a connected component filter (Forsyth and Ponce 2002). This filter is applied to the difference image to group it into "blobs". Next, the bounding box of each connected component is calculated. To minimize false positives, any potential targets smaller than 5x5 pixels are discarded. When this algorithm finds multiple connected components, we only choose the potential target that contains the cursor, and if there happen to be two of these we choose the one with the smallest area. If the connected component algorithm does not find any potential rectangles larger than 5x5 pixels, the bounding box of the full difference image is set as the target's size.

Difference Image Errors. A difference image can only reveal the target when there is a clear visual change after the button press. Difference images consistently failed to find the target when the button press triggers visual changes to non-target pixels that overlap the target or the target is replaced with new interactors on the button release event. Examples when this type of error occurs include tooltips (Figure 37), dropdown menus, tabs and popup menus.

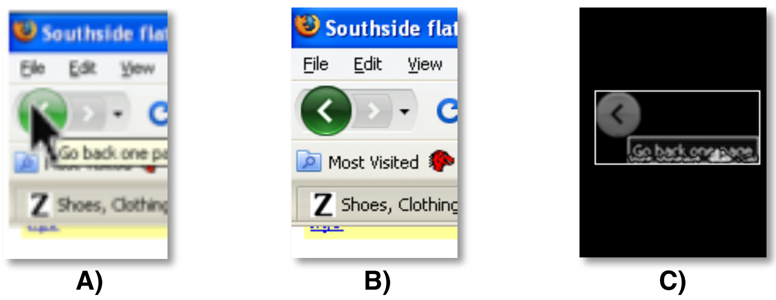


Figure 37 Example of a visual change error in difference imaging: New visual content appears that overlaps or touches target. **A)** When the user presses the pointer button, a tooltip appears overlaps the target. **B)** When the pointer button is released, the tooltip disappears. **C)** When the two images are subtracted from each other the resulting difference region includes both the target and the tooltip, which is incorrect.

The difference image technique also fails to identify the correct target size when there is no visual feedback indicated by a button press, so the up and down image are identical. Identical images most frequently occur when the target is already highlighted before the button press, such as the dropdown menus found in Microsoft Word and the message list in Google's online email. Fortunately our color matching technique (discussed next) is able to handle many of the cases when difference imaging is not able to identify the target in identical images.

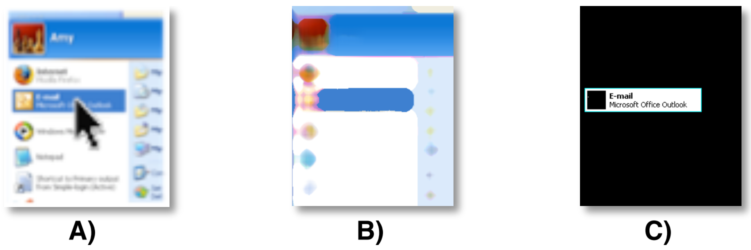


Figure 38 Target identified using color matching. Difference imaging failed to find this target because images captured from the button down and button up events were identical. **A)** Original image. **B)** Image after median blur. The color over the cursor (dark blue) is selected for color matching **C)** The matching algorithm selects the smallest blob of that color over the cursor (the correct target).

7.1.1.2. Color Matching

We use a color matching technique to find targets that are predominantly one color. This technique applies a median blur on an image and selects the color at the pixel under the cursor to use as the color to match (Figure 38) The image is then searched for this color and a corresponding blob of that color is found. We use the same blob finding techniques used with difference imaging to identify the target from the color blob. This technique works well when difference imaging fails because there is no visual change on a click as long as the item stays highlighted before and after the button press.

Color Matching Errors. Color Matching complements difference imaging because it works even when there is no visual change. It works best when there are separable blobs of the same color and does not support targets with a multi-color background (such as a

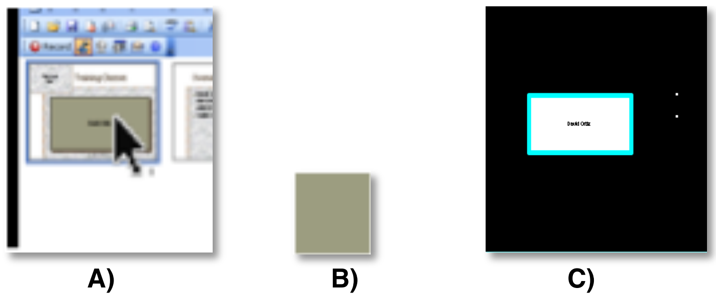


Figure 39 Example of false positive from color matching from Microsoft PowerPoint. Target in image **A)** was the thumbnail of slide 1 which is highlighted by the interface with a blue square. **B)** Color chosen to use for color matching after applying median blur to A. **C)** Results of color matching. In this example, choosing the color at the pixel under the cursor after applying the median blur was incorrect because the bounds of the selected color were much smaller than the target size.

gradient or shading as found on the Microsoft Windows Start Button). Additionally, it does not support targets that are not surrounded by a differently colored background (such a list items or checkboxes that share the same background). Figure 39 illustrates an example where color matching failed to identify the correct target because it matched a color that was connected to a smaller shape within the target.

7.1.1.3. *Template Matching*

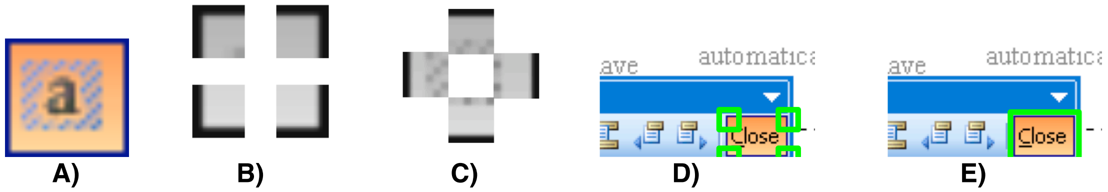


Figure 40 Example of template matching. **A)** Image that the template is formed from **B)** Corner templates: 5x5 pixels **C)** Edge templates 6x6 Pixels **D)** Source image with found corners marked with squares **E)** Source image with line around found target. Note that this technique is size independent, as the aspect ratio in E is different from A.

Template matching is a relatively simple computer vision technique that takes in two images, a source image and a template image, and tries to find occurrences of the template in the source (example in Figure 40). The algorithm does this by convolving (a process that is similar to taking the moving average of a subset of pixels over the whole image) the source image with the template. This technique quickly identifies all regions of the source where template pixels match the target image. Template matching can be very accurate, with few false positives, on templates it knows about. Figure 40 illustrates the process of matching edges and templates from a source image and a type template.

Corner and Edge Detection. Since template matching is being used to identify common visual cues, specifically rectangular borders, we use templates of corners and edges. These edges and corners (Figure 40B and 40C) are automatically extracted from a type template (Figure 40A), which is described later in this section. Potential corners and edges are restricted to locations that are valid within the parameters of the image capture. Specifically, any potential top left corners must be above and to the left of the cursor, etc.

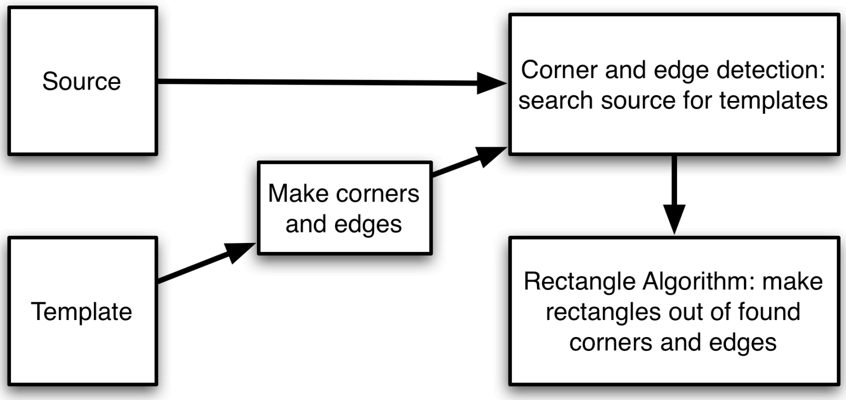

Source
Corner and edge detection: search source for templates
Make corners and edges
Template
Rectangle Algorithm: make rectangles out of found corners and edges

Figure 41 Corner and edge template matching algorithm takes in a source image and a template, extracts the edges and corners from the template, and searches the source for them with the template. Finally, rectangles are formed from matched corners and edges.

Rectangle Algorithm. The matched edges and corners are combined to build rectangles. Our rectangle algorithm creates potential corners out of two strong adjacent edges (lines which had 2 or more edge template matches). Next the algorithm takes all corner template matches and corners created from adjacent edges, and searches the set for possible rectangles. Finally, the rectangle algorithm chooses the rectangle with the smallest area when it finds multiple possible rectangles (Figure 41).

Template Acquisition. Templates are systematically added when the other two vision techniques (difference image and color matching) fail to identify a target. To create an initial set of templates, we consider each target that failed the preliminary version of the system without template matching (on data that was hand labeled with ground truth). For each of these failures we semi-automatically create a set of edge and corner templates. Edge and corner templates are created from a template example image (Figure 40A). Currently, this image is acquired semi-automatically by first cropping an image of a sample target to appropriate boundaries manually. Once this image has been identified, the corners and edges of these targets are automatically extracted to form the template.

Template Matching Errors. Fixed size template matching (*a.k.a.* icon matching) works best on icons or unique targets that do not have a border. Variable size template matching (a.k.a. *corner matching*) works best on targets with distinct borders and is able to handle items that differ in size from the original template, something icon matching cannot do. Thus the two techniques complement each other. However, both have difficulty finding the correct target in cases where an animation has not completed when the screenshot is taken, because in that case the screenshot may not contain the final image that the template matches.

7.1.2. Choosing Amongst Multiple Targets

The four automatic target identification techniques described above (accessibility API and 3 computer vision techniques) are each well suited to identify different types of targets. However, there are some cases where multiple techniques will each identify targets, but with slightly different size or position. Thus, it is difficult to know in advance which recognition technique will best identify the target. Instead of developing heuristics or rules to determine which technique to use for a given target, we built a two-level machine learning based classification scheme that is able to predict which technique should be used. Figure 42 illustrates the architecture of our two-level recognizer.

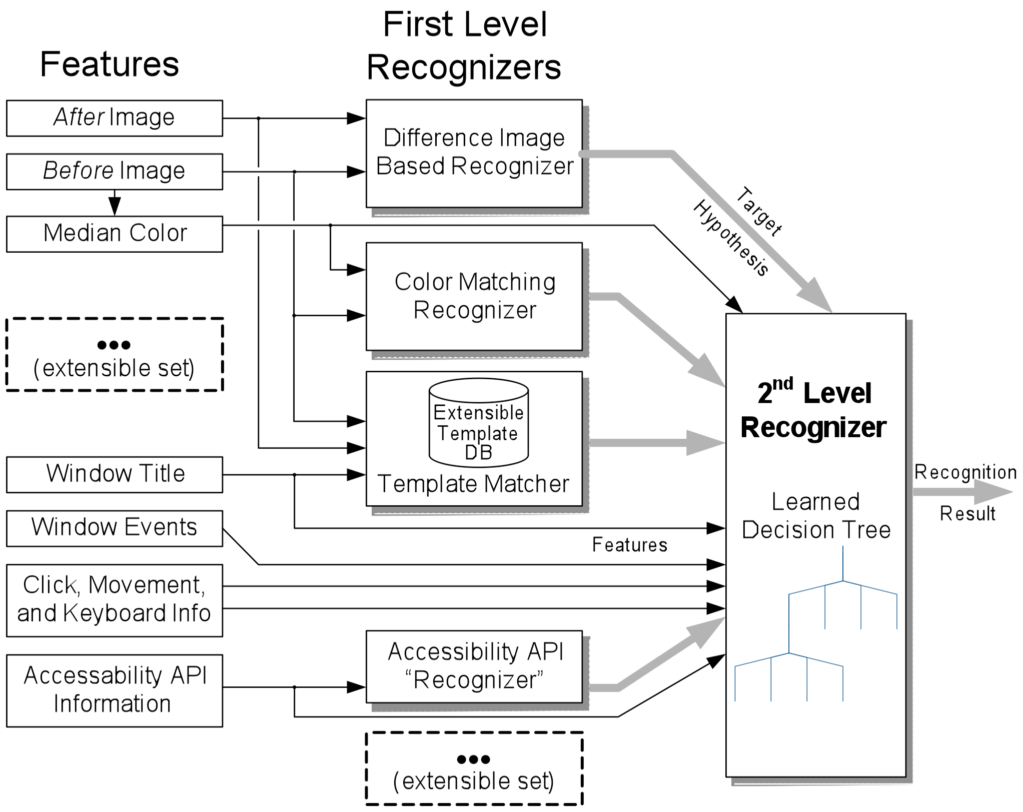


Figure 42 The architecture of our two-level recognizer. Thick gray lines represent target hypotheses, while thin black lines are features. Boxes at left represent feature generators. Middle column boxes are first level recognizers. The right hand box is the second level recognizer (classifier).

7.1.2.1. Structure of Two-Level Classifier

The first level of our classification scheme uses a set of classifiers each of which identifies some types of targets well, and others less well. We refer to these as *recognizers* to distinguish them from the *learned classifier* that intelligently picks from among these results based on features from the interaction. We use ADABoosting (Freund and Schapire 1996) on a decision tree, created with a variant of the well known C4.5 algorithm (Quinlan 1993), to implement this second level classifier. Our architecture is sufficiently general that first level recognizers can be added over time. Figure 43 illustrates the accuracy of each technique independently, plus the hybrid approach including the second-level classifier architecture on our simulated real world dataset (Section 7.1.2.2).

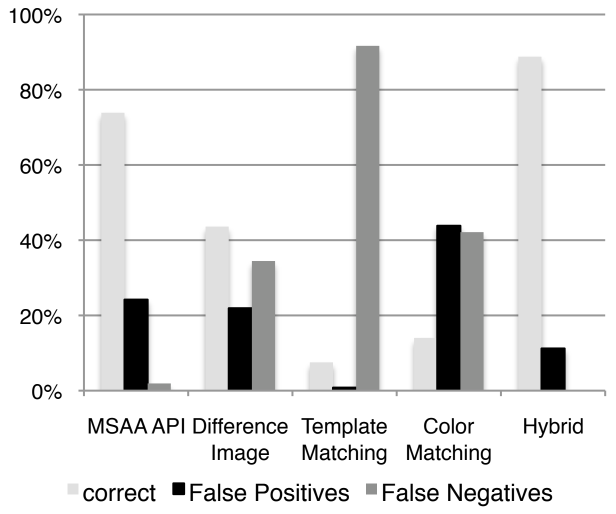


Figure 43 Raw accuracies of four first level recognizers (API, Difference Image, Template Matching, Color Matching) and the overall accuracy of our hybrid technique.

Features used by the first level classifier are extracted from the input event stream, MSAA API, and computer vision techniques. These features include the number and size of potential targets found, details from the computer vision calculations (such as whether or not the target was found using the connected component algorithm, or using a bounding box), timing information about the user's clicking actions, and information about window interactions.

7.1.2.2. Evaluation

In order to evaluate this technique a dataset with validated ground truth is required that contains the size and location of all targets. To do this, we evaluated our hybrid technique on a dataset we generated to be realistic and representative. Specifically, we carried out the exact steps specified in a set of tutorials for Windows and Microsoft products (called the Step By Step Tutorials). These tutorials give clear, step-by-step instructions for how to accomplish common tasks. Hand labeling data from these tutorials was much easier than labeling real world data because we had script of tasks and the targets that were selected, and never had to guess what targets users were trying to click on.

Because they provide the instructions for what to do, the tutorials remove any potential bias on the part of the experimenters in selecting targets. Because they focus on common tasks for common products, they provide reasonably representative data. We used portions of the following tutorials to create our initial dataset: Microsoft Outlook (Outlook Step by Step 2003), PowerPoint 2003 (PowerPoint Step by Step 2003) Microsoft Word 2003 (Word Step By Step 2003), and Learning Windows XP (Windows XP Step by Step 2005).

7.1.2.3. Evaluation: Results

In assessing accuracy we considered the size and position of each classification made by the system. We considered a classification to be correct if it contained the click point of the interaction and was within 15% or 5 pixels (which ever was larger) of both the width and height of our human labeled ground truth.

To estimate the overall accuracy of the resulting classifier we used the common 10-fold stratified cross validation method. We estimated the accuracy of the final classifier as the average accuracy found in 10 classifiers, each built from 90% of the test data and tested against the remaining 10%. Since the Accessibility API recognizer has the highest overall accuracy, we handle cases where the second level recognizer cannot make a clear choice by giving preference to the API result. This caused a 1% improvement in our final results, but slightly increased the number of false positives. Our tests indicate an overall accuracy of 89% (kappa = .8), and Figure 43 illustrates the overall accuracy of our hybrid approach compared the accuracy of each individual recognizer.

The logging software used for this analysis only captured 300x300 pixel images. Fortunately, this covered 92% of our targets. However, this limitation did cause some misclassification, since in the end 7.9% of the targets in our dataset ended up being larger than 300 pixels in some dimension (and so were clipped in the images our classifier uses). Of these 2.6% were actually misclassified (with the remaining 5.3% being correctly identified by the API without the need for images). However, CRUMBS now captures full size images, so we can avoid this hit in accuracy for future use.

Because templates may be added over time, it is useful to consider the impact of templates on overall accuracy. Our intuition was that a fairly small number of templates will provide sufficient accuracy benefits, and that this will allow us to distribute a manageable library of initial templates with a system and still get high accuracy from the beginning. As we progressed through 154 initial error cases (in random order) we eventually created 22 distinct templates. These templates successfully covered 102 of the initial 154 error cases, illustrated in Figure 44. In this figure we show that a small number of templates is sufficient to cover a majority of the missing targets (65 targets were solved with only 1 template).

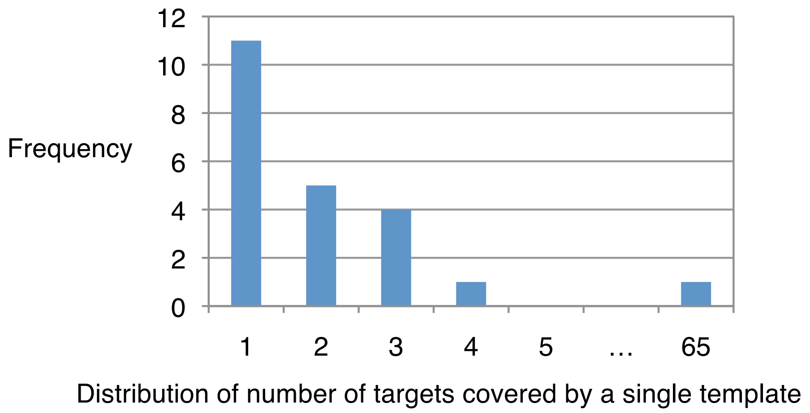


Figure 44 Most templates (11) found only one target (the one they were created from). The most successful template of the 22 we created found 65 targets, and the rest found four or fewer targets. While a relatively small number of targets were found by each template, the cost of creating each one is small.

7.2. Analyzing Movement Behaviors To Predict if an Adaptation is Needed

Given our goal of using automatic assessments to improve computer access for individuals who experience pointing problems, we were interested in using predictive models to choose appropriate adaptations based on performance. Specifically, we wanted to build on our success at predicting with high likelihood which group a given movement came from (Section 6.1), and predict whether or not an individual would actually benefit from a specific adaptation based on data from their unadapted use. This classification is exactly what a full-featured accessibility adaptation system would need to assess whether or not an adaptation should be deployed.

In order to investigate this problem, we looked at a dataset that had data from individuals with and without a pointing adaptation. This dataset was collected to evaluate the performance of the "Steady Click" (Trewin *et al.* 2006) adaptation that was designed to minimize pointer slips during a click. This adaptation creates a “Steady Click” by disabling dragging during a click (i.e. the user is not able to move the pointer, or slip, while one of the buttons is pressed). In that work, it was found that this adaptation significantly reduced slipping errors for 8 of the 11 participants, and lead to significantly improved target acquisition times for 5 participants. For some participants, suppression of slipping errors did not significantly improve performance because other targeting errors remained. For others, target acquisition times were reduced, but the improvement was not statistically significant, perhaps due to insufficient data. The work presented in this thesis uses error rates as the fundamental measure of the “helpfulness” of Steady Clicks.

The Steady Click dataset consists of 18 participants performing clicking tasks. This previously collected dataset consists of 11 motor impaired adults (5 female, mean age 49) who were pre-screened as having slipping problems, or self reported such problems, and 7 able-bodied adults (4 female, mean age 41) who did not. Two of the participants with disabilities had Parkinson’s Disease, two had Cerebral Palsy, three had impairments results from a stroke, one had Multiple Sclerosis, another had spinal cord damage resulting from a gunshot accident, and two had impaired manual dexterity caused by unspecified neuromuscular conditions. All were familiar with a standard mouse and used that as their input device. All participants completed a task where they were presented with a 19-column and 30-row grid of rectangles. Each rectangle was 52 pixels wide and 22 pixels tall and contained a 2 to 5 character word in it. For each trial, one of the rectangles would be highlighted in blue and the user would need to move their mouse to that target and click in the rectangle (Figure 45).

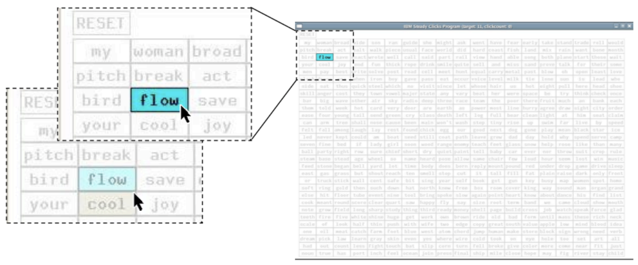


Figure 45 Screen shot of study task showing how target cell is highlighted and how the user’s current location is also highlighted.

7.2.1.1. Fetaures available for this dataset

Task Specific Features

- Was the click on the target?
- How long did it take to complete the move-click sequence?

Features Related to the Click

- Duration of click
- Number of other button events that occurred during a click
- Distance pointer moved during the click

Features Related to Movement

- Velocity and acceleration at mouse down and mouse up events.
- Peak velocity reached during the movement
- How far did the pointer travel during movement?

Pause Features

These features measure the state of the cursor at the time of the last pause before the click. There are 5 categories of features, with individual features varying the definition of a pause (in 50 milliseconds increments between 100 and 300 milliseconds).

- How long was the last pause prior to the click?
- How far away was the cursor from the target at the start of the pause?
- How much time was spent moving between the pause and the click?
- How far did the cursor move between the end of the pause and the click?
- What was the peak velocity between the pause and the click?

7.2.1.2. Predictive Model Results

All the participants were grouped into three categories based on whether or not they had a motor impairment and whether the Steady Clicks adaptation significantly reduced their slip errors: Group 1: (H-I) Steady Clicks helped and they were motor impaired (8 participants) Group 2: (NH-I) Steady Clicks did not help and they were motor impaired (3 participants) Group 3: (NH-A) Steady Clicks did not help and no motor impairment (7 participants). There were no instances of users without motor impairment for whom Steady Clicks did help. Essentially, those who were helped by Steady Clicks were those who frequently slipped while clicking the mouse. We divide this data into three groups (instead of two purely based on Steady Clicks helping vs. not helping) because early analysis indicated that the NH-I and NH-A had fairly different characteristics. Attempting to treat these disparate groups as a single category proved problematic for the classifier.

The disabilities of the three participants for whom Steady Clicks was not helpful were Parkinson's Disease (2 individuals) and Cerebral Palsy.

We used a two-level classifier to predict if individuals would benefit from the Steady Click adaptation. The first level of this classifier is a three-way decision tree classification. The results of this three-way classification, performed on each instance, are then used to classify each person (as "helped" or "not helped") by considering whether the majority of their movement trials predicted them to be in the H-I group vs. the NH-I or NH-A groups.

Predicting need for Steady Clicks Level One: First we employed per-person hold out to build train/test sets, which were used to build decision trees using wrapper based feature selection. The most common features selected by the wrapper-based feature selection algorithm were: 1) the distance the participant slipped, 2) the total time it took to complete the trial, and 3) the length of the click. This three-way classification distinguished between instances from these three groups with 82.7% accuracy (Kappa = .71).

Predicting need for Steady Clicks Level Two: As indicated above, since our goal is to predict if Steady Clicks would help a given individual, in the second level of our model, we aggregated the results for the NH-I and NH-A groups, and then classified each person based on the most frequently predicted group among their individual movements. Using this technique, we were able to correctly predict if Steady Clicks would reduce the user's clicking errors for 17 out of 18 people, or 94.4% accuracy.

7.2.2. Discussion on How to Extend Steady Click Results to Other Adaptations

The strategy we took to predict whether or not an individual's performance would improve with a specific adaptation requires performance data with and without that adaptation. In order to make more performance-based predictions about the need for an adaptation, we will need real world data with and without specific adaptations. The study protocols we have used for real world data collection do not work for these predictions because they do not require participants to use adaptations. As a result, there is no ground truth measure of whether an adaptation would have helped.

In order to use this technique to predict the need for more adaptations, we would need more examples of adapted and unadapted use. It would be possible to repurpose more pre-existing datasets that were collected to measure performance benefits with a particular adaptation to perform this investigation. However, we would like to understand the relationship of real world GUIs to these adaptations and then determine if these models should be built from real world data instead of laboratory data. Our current protocols could be easily modified to support this kind of collection by automatically deploying certain adaptations for a given time period to collect enough data to do this analysis.

7.3. Summary of Improved Automatic Assesment Techniques

This section discussed two extensions to the core work of this thesis: improving automatic target identification and predicting the need for a specific adaptation. As we discuss in future work, there is more to be done before we fully achieve our vision of automatic assessment of real world pointing data.

8. Dissertation Summary

This thesis has made important strides towards making automatic assessments of real world pointing performance. In the previous chapters, we have described a body of work that has produced a novel, large corpus of real world pointing data and described pointing in terms of targeting and movement performance metrics. We have also described success at assessments of pointing behavior on both laboratory and real world data. The four main contributions of this thesis are summarized below.

Collected Large Dataset of Real World Data with Custom Logging Tool from Individuals with and Without Pointing Problems: We developed collection tools to capture real world pointing use. These tools were used to collect a dataset of more than 400,000 real world clicking tasks from individuals with and without motor impairments over two years. Data was collected from both laptops and shared computers in community centers from 13 individuals with motor impairments, and 8 older adults, and 4 individuals without motor impairments.

Novel Analysis of Real World Data: We analyzed real world data in terms of performance metrics, software applications covered, and distribution of target sizes according to the MSAA API. We investigate these measures by group {older adult, able bodied, or motor impaired individuals}. We found that 1) older adults and motor impaired individuals had better clicking and movement performance during games than Internet or MS Office tasks, 2) individuals with motor impairments had the most difficulty with clicking tasks, and 3) able bodied individuals had the fastest movements. Previously, there had been little investigation of these issues for these individuals performing real world tasks.

Accurate Assessment of Pointing Behavior: We demonstrated that we can detect problematic pointing performance using predictive models constructed using machine learning techniques. We demonstrated that target size, click duration, and number of X direction changes are important to predicting pointing performance in real world data. Using these features, we were able to detect problematic pointing performance while surfing the web and using the Microsoft Office suite with over 90% accuracy.

Improved Automatic Detection of Interactive Targets in Real World Applications: We demonstrated that accessibility APIs are able to accurately detect the size and location of about70% of all targets during real world use. While this number may seem large, it leaves many applications unsupported (of specific interest to our target populations are games). To improve the number of targets identified, we developed a technique that combines information from Accessibility APIs with computer vision techniques to automatically identify targets. Our target identification technique finds 15% more targets than the standard accessibility API alone.

8.1. Reflection about Limitations of Real World Data

Throughout this thesis we have discussed the many benefits to moving away from laboratory pointing tasks to study real world use. However, we did observe our share of limitations with this dataset. Unlike laboratory evaluations, when studying real world use the experimenter can never know fully know the user's intent, making it difficult to calculate a performance-based error metric. When analyzing CRUMBS logs, it is not possible to tell if the user slipped off a target, tried to drag it, or was just idly playing with the cursor. Of course, the only way to really know the user's intent is to ask them. However, it is unlikely that users would be happy with (or even tolerate) a system that constantly probed them and asked about their performance whenever there was a change.

Another major problem that with real world deployments is the lack of control over what tasks users perform, and how much data will be collected. All of our participants started with the same set of applications on their computers, but all of them used these computers differently. In general, the able bodied group used the computers to surf the web and watch movies, older adults used them to play games, and participants from UCP used them to complete class assignments. Depending on participant usage patterns, this approach may result in collecting a small number of samples from any single application. For example, as mentioned in Chapter 7, out of more than 360,000 samples, only 2,500 were of MS Office use. Experimenters who wish to use this approach might need to plan for longer deployments in order to ensure a large enough sample size from the desired applications.

8.2. Future Work

This thesis has laid much of the necessary groundwork to automatically assess pointing performance during real world computer tasks. Here we discuss some ways in which our tools and data could be extended, used, and applied.

Figure 42 Our goal is to improve computer access for individuals with pointing problems, however the contributions of this thesis have much broader applications.

8.2.1. Real Time Automatic Assessment During Real World Use

In this thesis we have shown how we can collect, analyze and classify real world pointing data, but we haven't yet tested our technique in a real time application. We have designed our assessment approach so it can be easily adapted to run in real time. We expect this pipeline to be similar to what we did when assessing menu actions in real world data (Section 3.2 and 3.3). In order to build a real time assessment system, CRUMBS data will need to be analyzed periodically during real world use, and then this recent data will be given to a predictive model to classify. These classifications could then be used to inform an adaptation of a user's current ability so assistive software can be deployed or configured.

8.2.2. Designing Interfaces that use Automatic Assessments to Improve Computer Access

Automatic assessments of pointing performance can be used to improve accessibility problems. However, it is unclear how these assessments should be used to best serve individuals with pointing problems. Approaches we have considered range from informing the user about the problem they are experiencing to deploying an adaptation specifically designed for their pointing problem. Another application of this technology would be to create reports about an individual's performance over time that could be shared with their caregiver.

However, we recommend that any investigation into the design of these systems should be user-focused. Specifically, we advocate the use of participatory design between the individual having pointing problems, their caregivers, instructors, and assistive technology specialists due to the highly personal nature of disability and impairments. One of the issues that needs to be explored in these design sessions includes how much should the user be made aware of their performance? Do people want to be told that their performance was worse today than it was yesterday? Additionally, there are many questions about how much control the user wants to have over any automatic adaptations, particularly considering that this kind of system will make errors. Our intuition is that some of these issues will vary by individual and their ability, but it is important to investigate these issues thoroughly before building and deploying automatically adapting systems.

8.2.3. Automatic Target Recognition

Our hybrid target recognition technique was more successful at detecting targets in an interface than the Accessibility API was alone. However, this technique has many applications beyond our initial motivation for creating it. Below we describe how this technology could be used to support automatic usability analysis of pre-existing software and how it could be used in end-user programming systems.

8.2.3.1. Supporting Automatic Extraction of a Task Sequence

In usability evaluations, having a list, or *task sequence*, of the targets a user interacted with to achieve a goal is desirable. Task sequences can be analyzed to reveal differences between steps a user performed and some pre-defined sequence, or to compare the actions of multiple users. Experimenters typically generate a task sequence through logging software built into a custom application (Findlater and McGrenere 2004) or by hand coding video logs (Kaufman *et al.* 2003). Our automatic target identification technique (discussed in Section 7.1) could help usability specialists by automatically extracting these sequences from real world use, and could be used in existing video annotation tools such as Transana (http://www.transana.org).

8.2.3.2. Automatically Scripting Common Actions

During real world use, it is common to encounter repeated sequences of UI interactions. Tools such as Automise (http://www.automise.com) enable users to easily generate scripts to perform multiple UI tasks. We envision automatic target identification being incorporated into these tools to automatically identify targets in frequently performed task sequences and suggest scripts to automate these tasks. Yeh et al. have begun to explore this domain with a system called Sikuli, which allows users to graphically write GUI automation scripts (Yeh *et al.* 2009), a form of end-user programming.

8.2.4. Future Analysis of Real World Pointing Data

The analysis in this paper highlights the performance difference between our three participant groups using a few clicking, targeting and movement metrics that were easily collected. While our analysis is informative of real world pointing behavior, there are still numerous other metrics that can be investigated. A few examples of these metrics would include looking at how many times the cursor entered a target's boundaries, detecting frequency of accidental clicks, and more sophisticated analysis of pauses during movement.

Additionally, we see a range of work that needs to be done in segmenting real world data into samples that describe a particular movement. In this thesis, we were interested in targeting movements and segmenting data by button events and pause durations (using a single pause threshold for all our participants). An important area of future work is to investigate how this threshold should be determined and individually tailored for users who may have pointing problems. Another promising area of future work is an investigation of detecting and segmenting pointing actions other than targeting such as dragging and steering.

8.2.5. Future Uses of our Real World Dataset

For this thesis, we collected a novel and large dataset of real world computer use that we analyzed to assess pointing performance. However, this dataset can be useful for many other analyses. Two promising areas of investigation are typing performance and understanding usability problems in pre-existing interfaces. For example, this dataset could be used to analyze typing behaviors during real world use. It could also be used to understand real world task sequences (by analyzing the screenshots captured during mouse events).

In the future, we hope to release this dataset to other researchers. However, in order to do so, we would need to scrub the data for anonymity. Cleaning the mouse data is straightforward as it does not have much personal information: however, the keyboard and image data are more complicated. Currently our keyboard data contains passwords, usernames, email addresses, and private emails. One simple technique to make this data available for other researchers would be to omit the keycodes in the data, and merely report the keyboard timings. However not having the exact keycodes makes many types of keyboard analysis tricky. Anonymizing the image data is tricky because both pictures and on-screen text can reveal sensitive information. The best way to tackle this problem would be to detect text and images and blur or replace this information.

Acknowledgements

I thank my family, friends, and thesis committee for their continued support and enthusiasm.

I also thank all our participants and the staff at United Cerebral Palsy Center in Pittsburgh for their time and commitment to help collect this data.

This work was supported in part by the National Science Foundation under grants EEC-0540865, IIS-0713509, IIS-0325351, IIS-0205644, and Amy Hurst's NSF Graduate Student Research Fellowship. Additional financial support came from IBM Research; and the Pennsylvania Infrastructure Technology Alliance, the Quality of Life Technology Center, and the Richard King Mellon Foundation.

9. Bibliography

J. Accot and S. Zhai "Beyond Fitts' law: models for trajectory-based HCI tasks." Proceedings of the SIGCHI conference on Human Factors in Computing Systems (1997): 295-302.

J. Accot and S. Zhai "More than dotting the i's - foundations for crossing-based interfaces." Proceedings of the SIGCHI conference on Human Factors in Computing Systems (2002): 73-80.

"Adobe Accessibility: Information and news about accessibility in Adobe products for people with disabilities, authors, and developers." http://blogs.adobe.com/accessibility/

D. Ahlström "Modeling and improving selection in cascading pull-down menus using Fitts' Law, the Steering Law and force fields." Proceedings of the SIGCHI conference on Human Factors in Computing Systems (2005): 61-70.

J. Alexander and A. Cockburn "An empirical characterisation of electronic document navigation." Proceedings of Graphics Interface 2008 322 (2008): 123-130.

G. Apitz and F. Guimbretière "CrossY: a crossing-based drawing application." Proceedings of the ACM symposium on User Interface Software and Technology (2004): 3-12.

R.S. Amant, H. Lieberman, R. Potter, and L. Zettlemoyer "Programming by example: visual generalization in programming by example." In Communications of the *ACM*, ACM Press, 43, 3, (2000): 107-114.

R.S. Amant and M.O. Riedl "A perception/action substrate for cognitive modeling in HCI." In International Journal of Human-Computer Studies, Elsevier, 55(1), (2000): 15-39.

R. Balakrishnan ""Beating" Fitts' law: virtual enhancements for pointing facilitation." International Journal of Human-Computer Studies 61.6 (2004): 857-874.

P. Baudisch, E. Cutrell, K. Hinkely, and A. Eversole "Snap-and-go: helping users align objects without the modality of traditional snapping." Proceedings of the SIGCHI conference on Human Factors in Computing Systems (2005): 301-310.

T. Beauvisage "Computer usage in daily life." Proceedings of the SIGCHI conference on Human Factors in Computing Systems (2009): 575-584.

J. Bigham, A. Cavender, J. Brudvik, J. Wobbrock, and R. Ladner "WebinSitu: a comparative analysis of blind and sighted browsing behavior." Proceedings of ACM SIGACCESS conference on Computers and Accessibility (2007): 58-66.

N. Brownlow, F. Shein, D. Thomas, M. Milner, and P. Parnes "Direct manipulation: Problems with pointing devices." Resna 89: Proceedings of the 12th Annual Conference, (1989): 246-247.

S. Carter, A. Hurst, J. Mankoff, and J. Li "Dynamically adapting GUIs to diverse input devices." Proceedings of ACM SIGACCESS conference on Computers and Accessibility (2006): 63-70.

O. Chapuis, R. Blanch, and M. Beaudouin-Lafon "Fitts' Law in the Wild: A Field Study of Aimed Movements." LRI Technical Repport. Number 1480. (2007): 1-11.

K. Coyne and J. Nielsen "How to Conduct Usability Evaluation for Accessibility: Methodology Guidelines for Testing Websites and Intranets With Users Who Use Assistive Technology." Nielsen Norman Group (2001).

S.A. Douglas, A. Kirkpatrick, and I.S. MacKenzie "Testing Pointing Device Performance and User Assessment with the IS0 9241, Part 9 Standard." Proceedings of the SIGCHI conference on Human Factors in Computing Systems (1999): 215-222.

S. Dumas and T.K. Landauer "Using Examples to Describe Categories." Proceedings of the SIGCHI conference on Human factors in computing systems (1983): 112- 115.

P.M. Fitts "The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement." Journal of Experimental Psychology 47 (1954): 381-391.

D. Forsyth and J. Ponce Computer Vision – A Modern Approach (first ed.), Prentice Hall (2002) ISBN 0130851981.

Y. Freund and R. Schapire "Experiments with a new boosting algorithm." Machine Learning: Proceedings of the Thirteenth International Conference (1996): 148-156.

L. Findlater and J, McGrenere "A comparison of static, adaptive, and adaptable menus." Proceedings of the SIGCHI conference on Human Factors in Computing Systems, (2004): 89-96.

K. Gajos, J. Wobbrock, and D. Weld "Improving the performance of motor-impaired users with automatically-generated, ability-based interfaces." Proceedings of the SIGCHI conference on Human Factors in Computing Systems (2008): 1257-1266.

"Gnome Took Kit (GTK+)" 2.6.10 http://www.gtk.org

"GNU Image Manipulation Program (GIMP)" 2.2.8 http://www.gimp.org

T. Grossman and R. Balakrishnan "A probabilistic approach to modeling two-dimensional pointing." ACM Transactions on Computer-Human Interaction 12, 3 (2005): 435-459.

Y. Guiard, R. Blanch, and M. Beaudouin-Lafon "Object pointing: a complement to bitmap pointing in GUIs." Proceedings of Graphics Interface (2004): 9-16.

D. Hilbert and D. Redmiles "Extracting usability information from user interface events." Computing Surveys (CSUR) 32, 4 (2000): 384-421.

K. Hinckley, E. Cutrell, S. Bathiche, and T. Muss "Quantitative analysis of scrolling techniques." Proceedings of the SIGCHI conference on Human Factors in Computing Systems (2002): 65-72.

A. Hurst, S.E. Hudson, and J. Mankoff "Dynamic detection of novice vs. skilled use without a task model." Proceedings of the SIGCHI conference on Human Factors in Computing Systems (2007): 271-280.

A. Hurst, J. Mankoff, A.K. Dey, and S.E. Hudson "Dirty desktops: using a patina of magnetic mouse dust to make common interactor targets easier to select." Proceedings of the ACM Symposium on User Interface Software and Technology (2007): 183-186.

A. Hurst, S.E. Hudson, J. Mankoff, and S. Trewin "Automatically detecting pointing performance." Proceedings of the International Conference on Intelligent User Interfaces (2008): 11-19.

A. Hurst, J. Mankoff, and S.E. Hudson "Understanding pointing problems in real world computing environments." Proceedings of the ACM SIGACCESS conference on Computers and Accessibility (2008): 43-50.

A. Hurst, S.E. Hudson, and J. Mankoff "Automatically identifying targets users interact with during real world tasks." Proceedings of the international conference on Intelligent User Interfaces (2010): 11-20.

F. Hwang "Partitioning cursor movements in "point and click" tasks." Proceedings of the SIGCHI conference on Human Factors in Computing Systems (2003): 682-683.

F. Hwang, S. Keates, P. Langdon, and J. Clarkson "Mouse Movements of Motion-Impaired Users: A Submovement Analysis." Proceedings of the ACM SIGACCESS conference on Computers and Accessibility (2004): 102-109.

S.T. Iqbal and E. Horvitz "Disruption and recovery of computing tasks: Field study, analysis, and directions." Proceedings of the SIGCHI conference on Human Factors in Computing Systems (2007): 677-686.

"ImageMagick Software Suite" http://www.imagemagick.org/

Inspect32.exe, "Active Accessibility 2.0 SDK Tools." http://www.microsoft.com/downloads/details.aspx?FamilyID=3755582a-a707-460a-bf21-1373316e13f0 (2004).

ISO, Reference Number: ISO 9241-9:2000(E). Ergonomic requirements for office work with visual display terminals (VDTs)—Part 9—Requirements for non-keyboard input devices (ISO 9241-9) (Vol. February 15, 2002): International Organization for Standardization (2002).

Java Accessibility, http://java.sun.com/javase/technologies/accessibility/docs/jaccess-1.3/doc/core-api.html

JAVA Advance Imaging, Template Matching https://jaistuff.dev.java.net/
S. Jul ""This is a lot easier!": constrained movement speeds navigation." Proceedings of the SIGCHI conference on Human Factors in Computing Systems (2003): 776-777.

D.R. Kaufman, V.L. Patel, C. Hilliman, P.C. Morin, J. Pevzner, R.S. Weinstock, R. Goland, S. Shea, and J. Starren "Usability in the real world: assessing medical information technologies in patient's homes." Journal of Biomedical Informatics, 36, 1/2, (2003): 45-60.

S. Keates, F. Hwang, P. Langdon, P. Clarkson, and P. Robinson "Cursor measures for motion-impaired computer users." Proceedings of the ACM SIGACCESS conference on Computers and Accessibility (2002): 135-142.

S. Keates, P. Langdon, P.J. Clarkson, and P. Robinson "User models and user physical capability." User Modeling and User-Adapted Interaction (UMUAI), Wolters Kluwer Publishers 12(2-3), (2002): 139-169.

S. Keates and S. Trewin "Effect of age and Parkinson's disease on cursor positioning using a mouse." Proceedings of the ACM SIGACCESS conference on Computers and Accessibility (2005): 68-75.

S. Keates, S. Trewin, and J. Paradise "Using Pointing Devices: Quantifying differences across user groups." Proceedings Universal Access and Human-Computer Interaction (2005).

C.J. Ketcham, R.D. Seidler, A.W.A. Van Gemmert, and G.E. Stelmach "Age-related kinematic differences as influenced by task difficulty, target size, and movement amplitude." Journal of Gerontology, Series B: Psychological Sciences and Social Sciences 57.1 (2002): 54-64.

J.H. Kim, D.V. Gunn, E. Schuh, B.C. Phillips, R.J. Pagulayan, and D. Wixon "Tracking real-time user experience (TRUE): a comprehensive instrumentation solution for complex systems." Proceedings of the SIGCHI conference on Human Factors in Computing Systems (2008): 443-451.

H.H. Koester, E. LoPresti, and R.C. Simpson "Toward Goldilocks' pointing device: determining a" just right" gain setting for users with physical Impairments." Proceedings of the ACM SIGACCESS conference on Computers and Accessibility (2005): 84-89.

R. Kohavi and G.H. John "Wrappers for Feature Subset Selection." Artificial Intelligence 1.2 (1997): 273-324.

U. Kukreja, W.E. Stevenson, and F.E. Ritter "RUI—Recording User Input from Interfaces under Windows and Mac OS X." Behavior Research Methods 38.4 (2006): 656-659.

D.M. Lane, S.C. Peres, A. Sándor, and H.A. Napier "A process for anticipating and executing icon selection in graphical user interfaces." International Journal of Human-Computer Interaction 19.2 (2005): 243-254.

E. Lank, Y.N. Cheng, and J. Ruiz "Endpoint prediction using motion kinematics." Proceedings of the SIGCHI conference on Human Factors in Computing Systems (2007): 637-646.

I. Levine and M. Schappert "Method and adapter for performing assistive motion data processing and/or button data processing external to a computer". US Patent 2002/0158843 A1 (2002).

E. Mayo "The Human Problems of an Industrial Civilization." Cambridge, MA: Harvard University Press (1933).

I.S. MacKenzie and W. Buxton "Extending Fitts' law to two-dimensional tasks." Proceedings of the SIGCHI conference on Human Factors in Computing Systems (1992): 219-226.

I.S. MacKenzie, T. Kauppinen, and M. Silfverberg "Accuracy measures for evaluating computer pointing devices." Proceedings of the SIGCHI conference on Human Factors in Computing Systems (2001): 9-16.

P.D. Mahalanobis "On tests and measures of groups divergence." Journal of the Asiatic Society of Benagal. 26 (1930): 541–588.

D.E. Meyer, J.E.K. Smith, S. Kornblum, R.A. Abrams, and C.E. Wright "Speed-accuracy tradeoffs in aimed movements: toward a theory of rapid voluntary action." Attention and Performance XIII. (1990): 173–226.

"Microsoft Active Accessiblity API" http://msdn.microsoft.com/en-us/library/ms697707(VS.85).aspx

"Microsoft Developer Network SetDoubleClickTime function" http://msdn.microsoft.com/en-us/library/ms646263(VS.85).aspx (accessed 5/4/08)

"Microsoft Office Outlook 2003 Step By Step." Microsoft Press (2004).

"Microsoft Office PowerPoint 2003 Step By Step." Microsoft Press (2004).

"Microsoft Office Word 2003 Step By Step." Microsoft Press (2004).

I. Oakley, A. Adams, S. Brewster, and P. Gray "Guidelines for the design of haptic widgets." Proceedings of BCS HCI 2002 (2002): 195-212.

M. Orne "On the social and psychology of the psychoogical experiment: With particular reference to demand characteristics and their implications." American Psychologist 17 (1962): 776-783.

J. Paradise, S. Trewin, and S. Keates "Using Pointing Devices: Difficulties Encountered and Strategies Employed." Proceedings Unversal Access in Human Computer Interaction (HCII) (2005).

T.G. Phillips and T.J. Triggs "Characteristics of cursor trajectories controlled by the computer mouse." Ergonomics 44.5 (2001): 527-536.

M.E. Rodgers, R.L. Mandryk, and K. Inkpen "Smart sticky widgets: Pseudo-haptic enhancements for multi-monitor displays." Proceedings of Smart Graphics (2006): 194-205.

I. Sutherland "Sketch pad a man-machine graphical communication system." Proceedings of the SHARE design automation workshop (1964): 329-346.

C. Riviere and N. Thackor "Effects of age and disability on tracking tasks with a computer mouse: accuracy and linearity." Journal of Rehabilitation Research and Development 33 (1996): 6-15.

S. Trewin and H. Pain "Keyboard and mouse errors due to motor disabilities." International Journal of Human-Computers Studies (1999).

S. Trewin, S. Keates, and K. Moffatt "Developing steady clicks: a method of cursor assistance for people with motor impairments." Proceedings of the ACM SIGACCESS conference on Computers and Accessibility (2006): 26-33.

N. Walker, D.E. Meyer, and J.B. Smelcer "Spatial and temporal characteristics of rapid cursor-positioning movements with electromechanical mice in human-computer interaction." Human Factors 35.3 (1993): 431-458.

J. Whiteside, N. Archer, D. Wixon, and M. Good "How do people really use text editors?" Proceedings of the SIGOA conference on Office information systems (1982): 29-40.

"Windows XP Step By Step", 2nd edition, Microsoft Press (2005).

J. Wobbrock and K. Gajos “Goal Crossing with Mice and Trackballs for People with Motor Impairments: Performance, Submovements, and Design Directions.” ACM Transactions on Accessible Computing 1.1 (2008): 1-37.

A. Worden, N. Walker, K. Bharat, and S.E. Hudson "Making computers easier for older adults to use: area cursors and sticky icons." Proceedings of the SIGCHI conference on Human Factors in Computing Systems (1997): 266-271.

T. Yeh, T.H. Chang, and R.C Miller "Sikuli: using GUI screenshots for search and automation." Proceedings of the ACM symposium on User Interface Software and Technology (2009): 183-192.

L.S. Zettlemoyer and R.S. Amant "A visual medium for programmatic control of interactive applications.” In Proceedings of the SIGCHI conference on Human Factors in Computing Systems (1999): 199-206.

10. Appendix

10.1. Advertisement used to recruit participants for laptop study at Carnegie Mellon (posted to a campus message board).

Participate in research to collect a dataset of real world computer use.
This dataset will be used to analyze real world pointing use to develop software that is able to automatically detect pointing problems.

Requirements: Participants must be 18 years or older and be able to use a computer mouse.

Compensation: In addition to using a new laptop for the summer, you will be entered into a periodic raffle to win cash.

Schedule: This study to start in the middle of June. Before receiving the laptop you will participate in a one hour lab study (for which you will be compensated twenty five dollars).

Privacy: As part of this study we will record everything that you do on these computers (keystrokes, screenshots and operating system event logs).

As a participant in this study, you will have the ability to remove datasets that are sensitive and you do not want included in the dataset, or you will have the ability to periodically turn off the logging software.

To apply, please email amy@cmu.edu and answer the following questions:
• Please describe your computer use experience.
• How often do you use a computer?
• What do use computers for?
• What applications do use most frequently on a computer?
• What would change about your computer use if you had access to a laptop for the summer?

10.2. Questions about computer use that were asked in both laptop and desktop deployments

1. How long have you been using a computer?

☐1 Less than 6 months
☐2 About 1 year
☐3 2-3 years
☐4 3-5 years
☐5 5-10 years
☐6 10 or more years

2. On average, how many hours per week do you use your computer? (If you are having trouble thinking of an average, think back over the last two weeks.)

☐1 Less than 2 hours per week
☐2 2 to 5 hours per week
☐3 6 to 10 hours per week
☐4 11 to 20 hours per week
☐5 21 or more hours per week

3. Have you ever had any computer training?

☐1 Yes
☐2 No
If Yes, what kind?

4. For each of the following options, please indicate the extent to which you used a computer within the past 3 months.

	Not sure what it is$_1$	Never$_2$	Sometimes$_3$	Often$_4$
a. Email				
b. Getting information				
c. Conducting business (e.g., online purchasing, banking, bill-paying)				
d. Writing reports				
e. Preparing presentations				
f. Entertainment (games)				
g. Other Task you do often:				

5. Do you ever have difficulties reading information on a computer screen? If yes, please describe them.

6. Does the pointing device you most frequently use look like the one on the desk? If no, please describe the pointing device you use, and discuss why you don't use a mouse.

7. How long have you been using this pointing device?

8. If the pointing device you use most frequently isn't a mouse, have you used a mouse before?

☐1 Yes
☐2 No
If Yes, please describe why you don't usually use a mouse.

9. Have you ever changed any of the pointing device settings for your primary pointing device (movement speed, click lock, click speed)? If yes, please describe any of the settings that you prefer to use.

10. Which hand do you use to hold the mouse?

☐1 Left
☐2 Right

11. Would you say you are Left or Right-handed?

☐1 Left
☐2 Right

12. Does the keyboard you use most frequently look the same as the one on the desk? If no, please describe how the keyboard that you most frequently use is different.

13. How long have you used this keyboard?

14. Have you ever changed any of the keyboard typing settings (such as sticky keys, key repeat delay or repeat rate)? If yes, please describe any of the settings that you prefer to use.

10.3. Demographic Information for Participants in Laptop Deployment

Able Bodied Adults

	Gender	Age	Years using a Computer	Hours of Computer Use (per week)	Dominant Hand	Known Medical Conditions
A1	M	22	5-10 years	21+	right	
A2	M	26	10+ years	6 to 10	right	
A3	F	47	10+ years	21+	right	some arthritis
A4	M	22	10+ years	11 to 20	right	

Older Adults

	Gender	Age	Years using a Computer	Hours of Computer Use (per week)	Dominant Hand	Known Medical Conditions
OA1	F	80	less than 6 months	less than 2 hours	right	
OA2	F	58	1 year	6 to 10	right	used to have arthritis
OA3	F	66	10+ years	21+	right	arthritis
OA4	F	62	1 year	less than 2 hours	right	arthritis
OA5	F	74	1 year	less than 2 hours	right	arthritis
OA6	F	66	10+ years	6 to 10	left handed, uses mouse with right	arthritis
OA7	F	69	5-10 years	6 to 10	right	arthritis
OA8	F	66	10+ years	11 to 20	right	arthritis

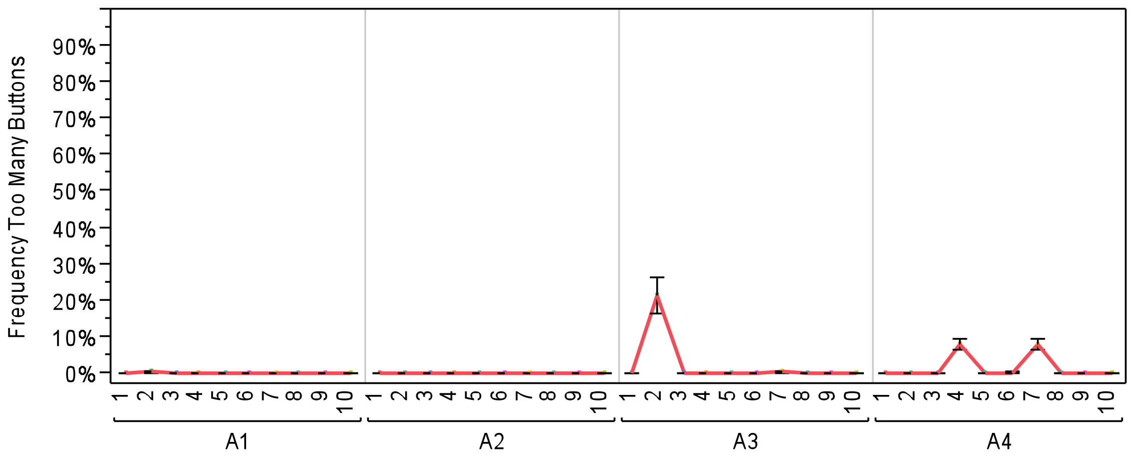
Individuals with Motor Impairments

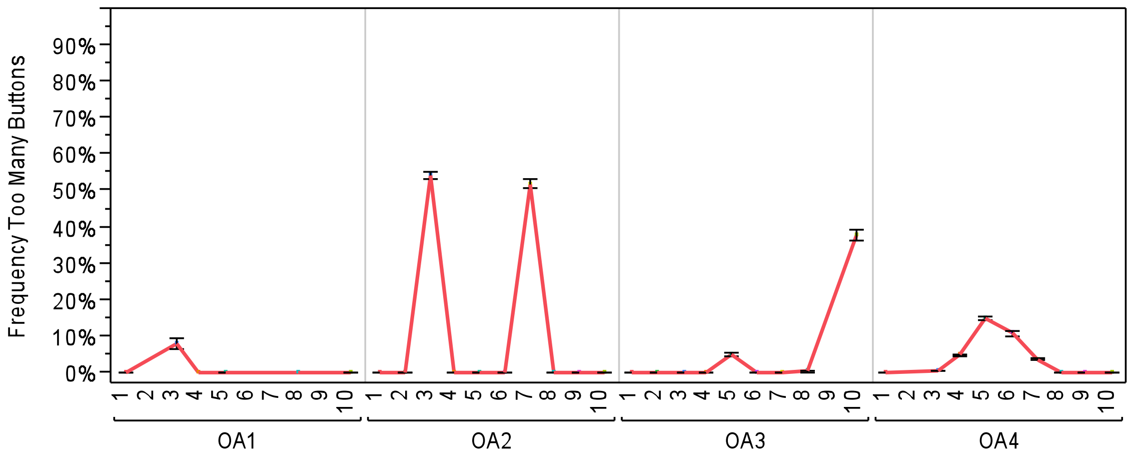
	Gender	Age	Years using a Computer	Hours of Computer Use (per week)	Dominant Hand	Known Medical Conditions
MI1	F	20	10+ years	21+	left	Cerebral Palsy, arthritis, upper extremity impairment, cannot move right limb.
MI2	F	47	3-5 years	2 to 5	right	Cerebral Palsy, Spinal Cord Injury, Traumatic Brain Injury
MI3	M	45	5-10 years	2 to 5	right	Traumatic Brain Injury, occasional tremor
MI4	M	55	3-5 years	11 to 20	right	Paralyzed from the waist down, memory and motor impairments (including an intermittent tremor).

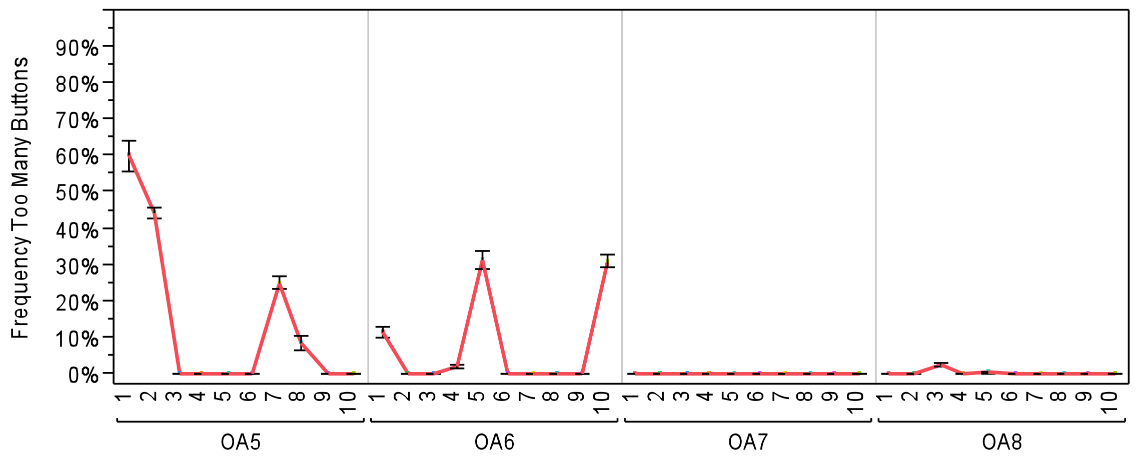
10.4. Laptop performance metrics across sessions

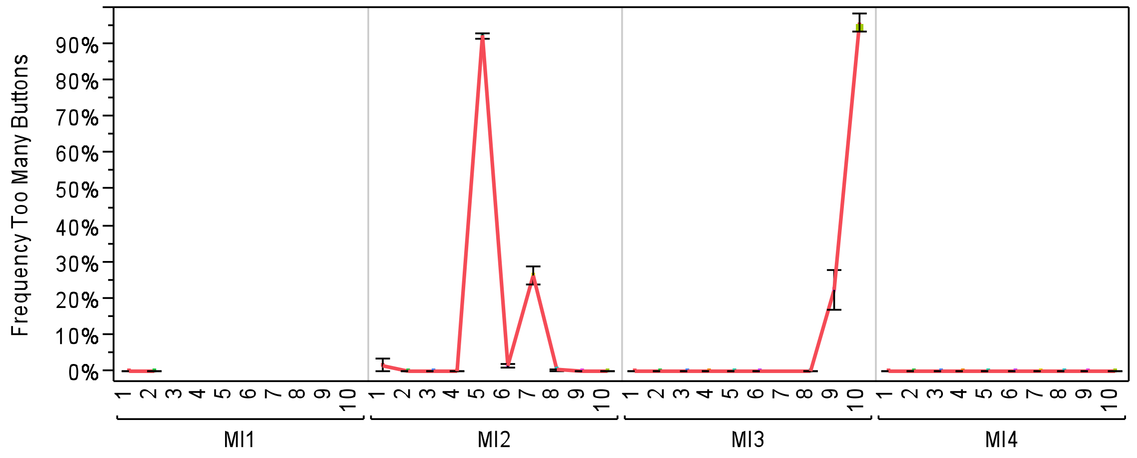
10.4.1. Frequency of pressing too many buttons

These four charts show the frequency of pressing too many buttons in ten randomly selected sessions. These charts are organized by group (A = able bodied, OA = older adult, and MI = motor impaired) and within each group, participant, and session number. Details about each participant are available in appendix 10.3.



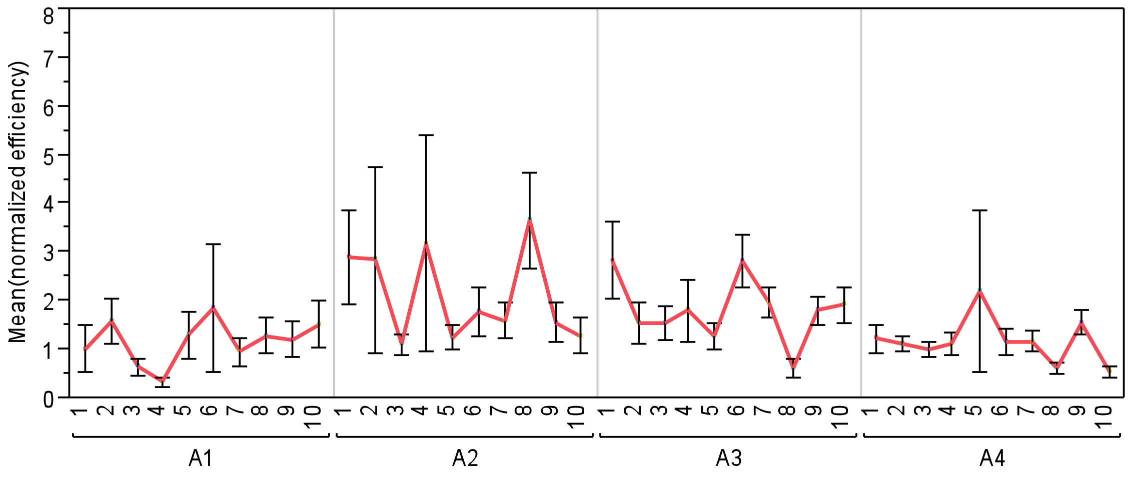


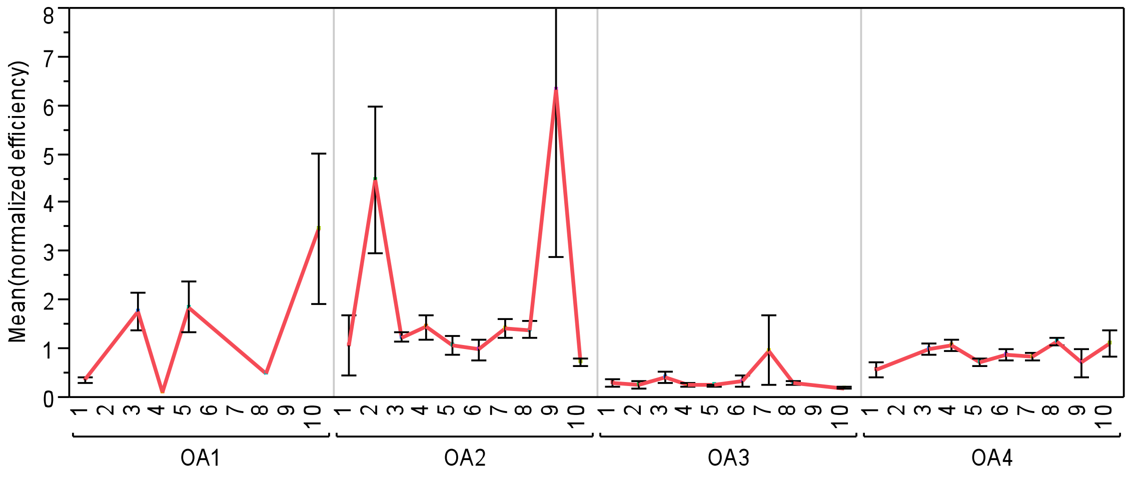


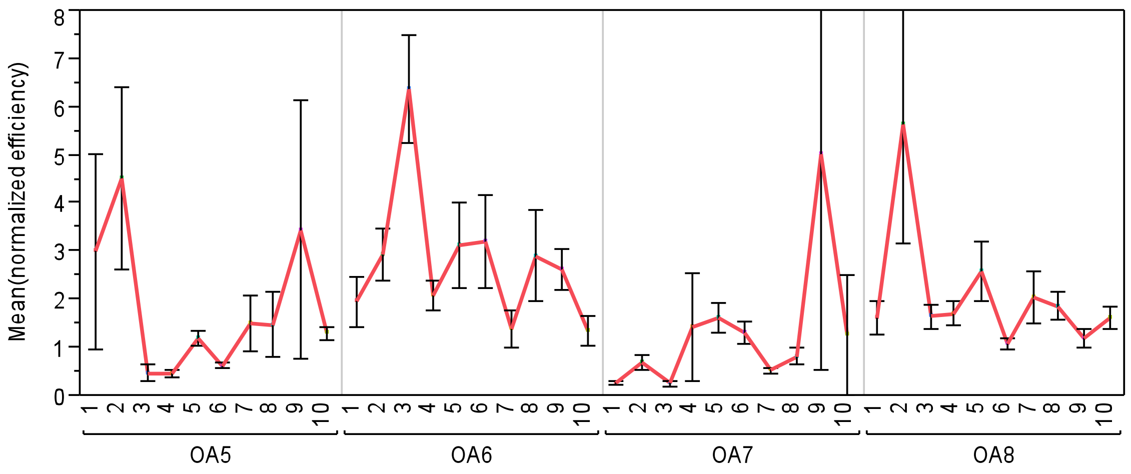


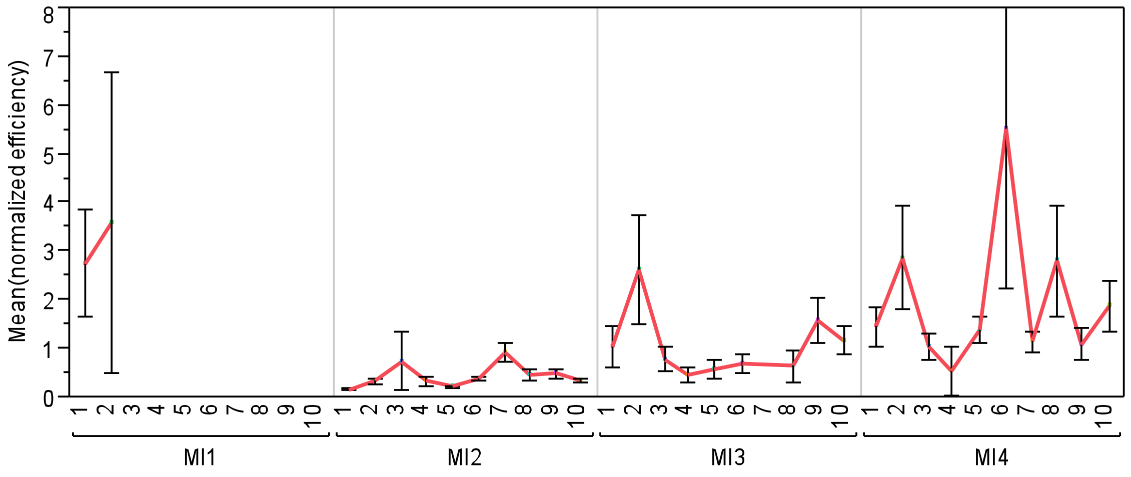
10.4.2. Normalized efficiency

These four charts show the mean normalized efficiency (equation 2) in ten randomly selected sessions. These charts are organized by group (A = able bodied, OA = older adult, and MI = motor impaired) and within each group, participant, and session number. Details about each participant are available in appendix 10.3.



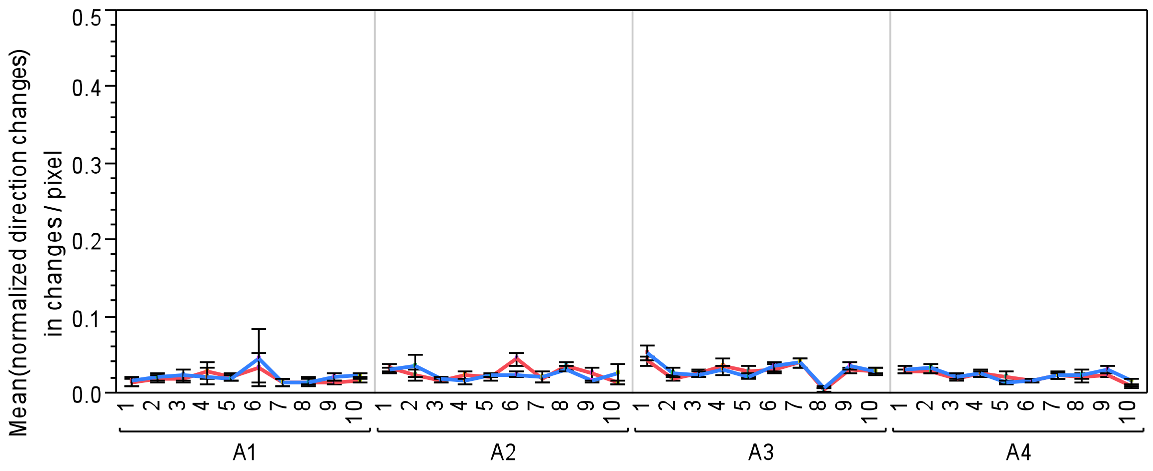

Mean(normalized efficiency)
0 1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9 10
OA1
OA2
OA3
OA4

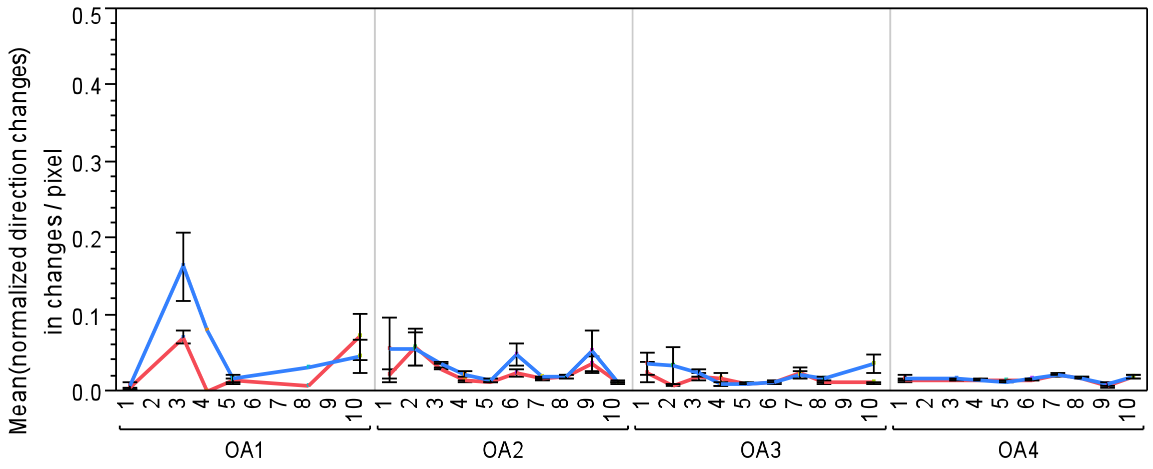

Mean(normalized efficiency)
0 1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9 10
OA5
OA6
OA7
OA8

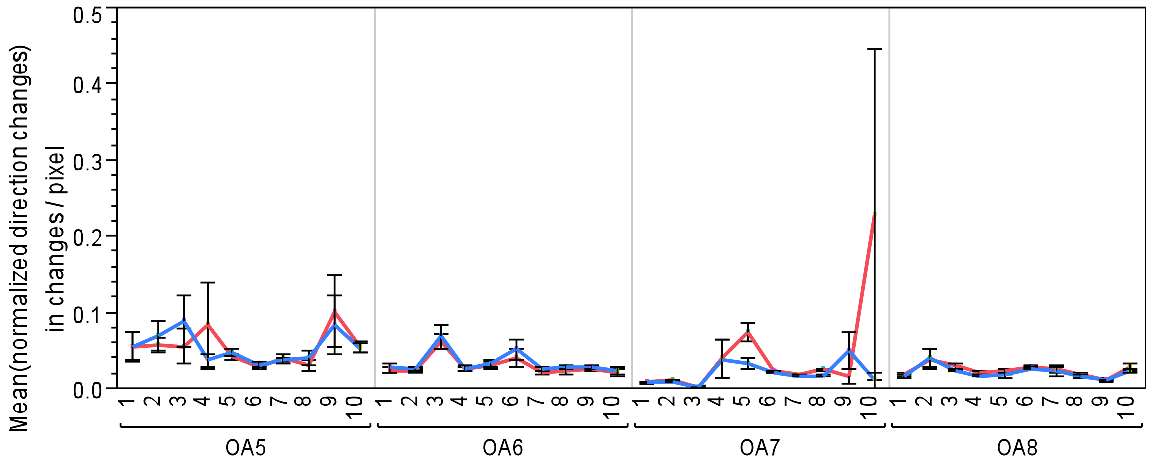

Mean(normalized efficiency)
0 1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9 10
MI1
MI2
MI3
MI4

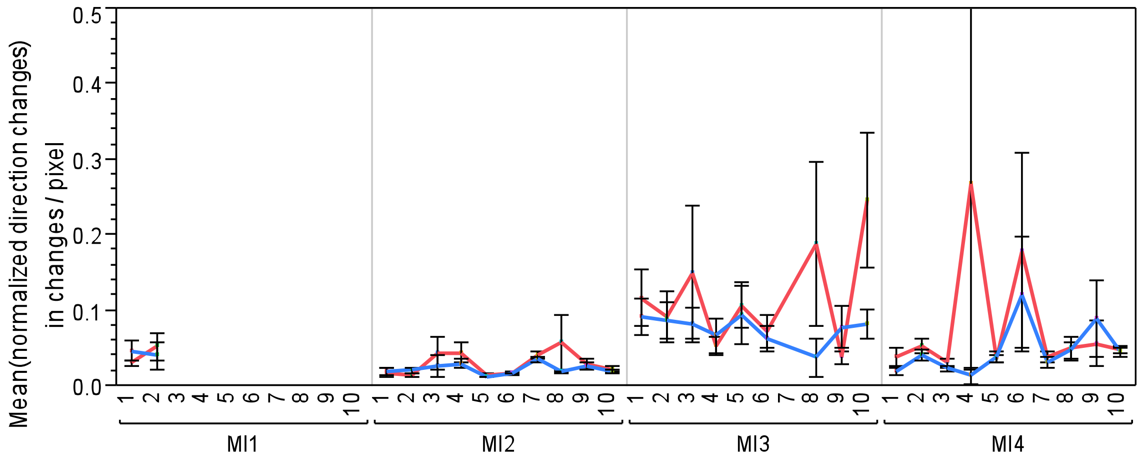
10.4.3. Normalized X and Y direction changes

These four charts show the mean number of X and Y direction changes (normalized by the Euclidean distance) traveled in ten randomly selected sessions. In these charts, the red line represents the mean number of direction changes in the X axis, and the blue line represents direction changes in the Y axis. These charts are organized by group (A = able bodied, OA = older adult, and MI = motor impaired) and within each group, participant, and session number. Details about each participant are available in appendix 10.3.



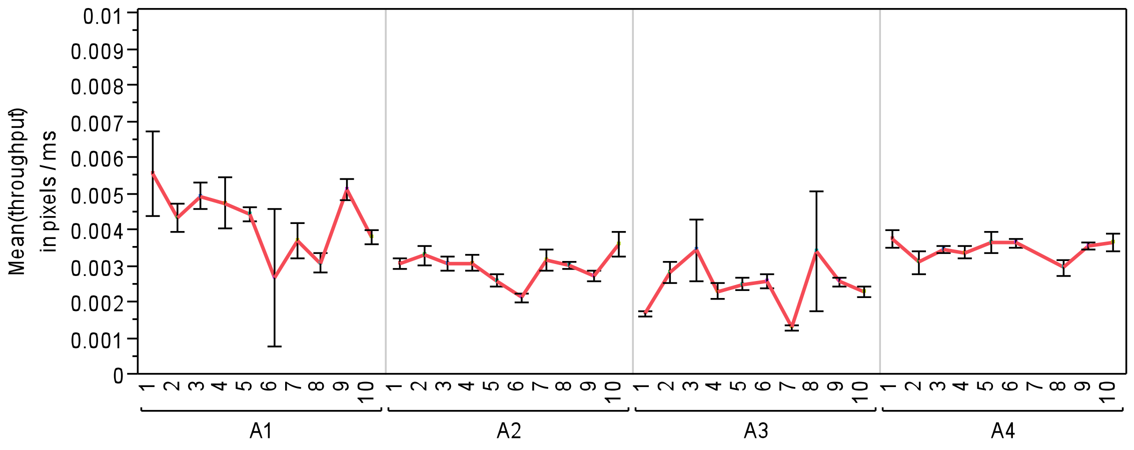


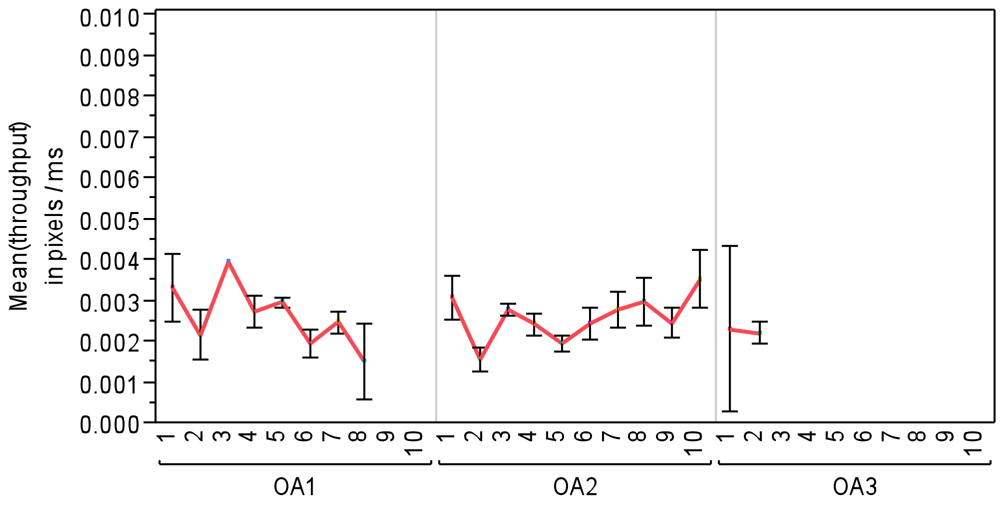


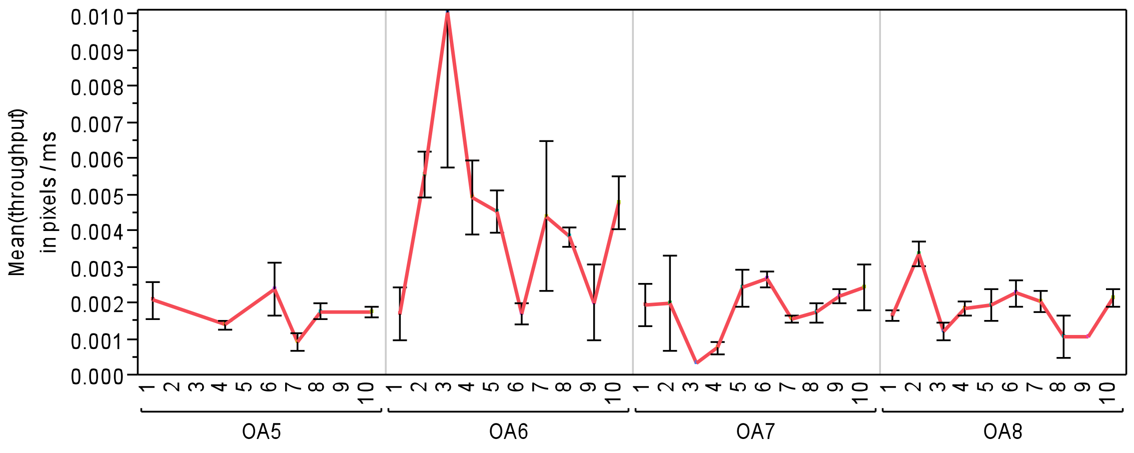


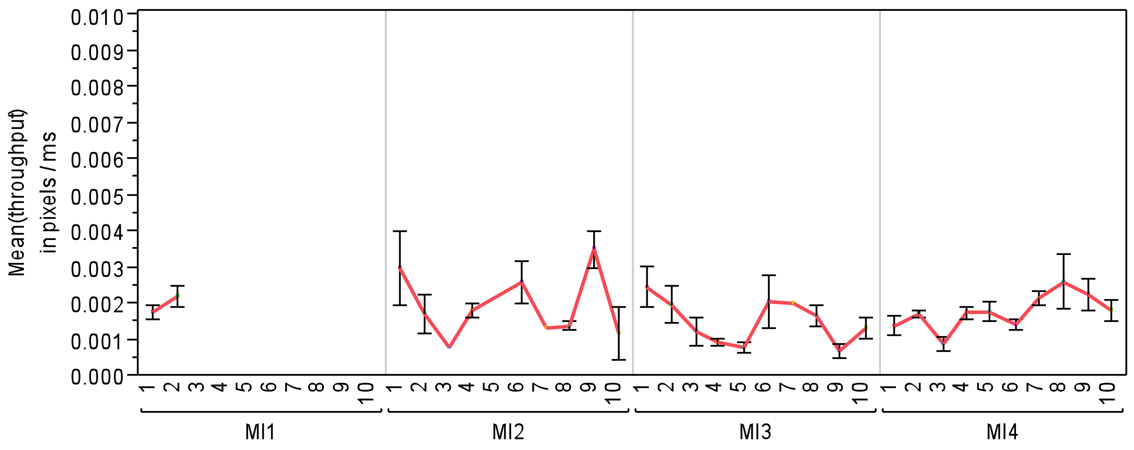
10.4.4. Throughput

These four charts show the mean throughput (equation 3) in ten randomly selected sessions. These charts are organized by group (A = able bodied, OA = older adult, and MI = motor impaired) and within each group, participant, and session number. Details about each participant are available in appendix 10.3. Note that we do not have throughput data for OA4 because we only collected game data from them.




Mean(throughput) in pixels / ms
0.000
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.010
1 2 3 4 5 6 7 8 9 10
OA1
OA2
OA3


Mean(throughput) in pixels / ms
0.000
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.010
1 2 3 4 5 6 7 8 9 10
OA5
OA6
OA7
OA8


Mean(throughput) in pixels / ms
0.000
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.010
1 2 3 4 5 6 7 8 9 10
MI1
MI2
MI3
MI4