Predicting Endpoint of Goal-Directed Motion in Modern Desktop Interfaces using Motion Kinematics

by

Jaime Ruiz

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2012

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

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

Abstract

Researchers who study pointing facilitation have identified the ability to identify–during motion–the likely target of a user's pointing gesture, as a necessary precursor to pointing facilitation in modern computer interfaces. To address this need, we develop and analyze how an understanding of the underlying characteristics of motion can enhance our ability to predict the target or endpoint of a goal-directed movement in graphical user interfaces.

Using established laws of motion and an analysis of users' kinematic profiles, we demonstrate that the initial 90% of motion is primarly balistic and submovements are limited to the last 10% of gesture movement. Through experimentation, we demonstrate that target constraint and the intended use of a target has either a minimal effect on the motion profile or affects the last 10% of motion. Therefore, we demonstrate that any technique that models the intial 90% of gesture motion will not be affected by target constraint or intended use.

Given, these results, we develop a technique to model the initial ballistic motion to predict user endpoint by adopting principles from the minimum jerk principle. Based on this principle, we derive an equation to model the initial ballistic phase of movement in order to predict movement distance and direction. We demonstrate through experimentation that we can successfully model pointing motion to identify a region of likely targets on the computer display. Next, we characterize the effects of target size and target distance on prediction accuracy. We demonstrate that there exists a linear relationship between prediction accuracy and target distance and that this relationship can be leveraged to create a probabilistic model for each target on the computer display. We then demonstrate how these probabilities could be used to enable pointing facilitation in modern computer interfaces.

Finally, we demonstrate that the results from our evaluation of our technique are supported by the current motor control literature. In addition, we show that our technique provides optimal accuracy for any optimal accuracy when prediction of motion endpoint is performed using only the ballistic components of motion and before 90% of motion distance.

Acknowledgements

I had a lot of help and support as I worked towards this degree. First I'd like to thank my supervisor, Edward Lank, who was abundantly helpful and ofered invaluable assistance, support and guidance throughout my PhD. I'm also grateful for help from Michael Terry who provided invaluable feedback (using his bun, meat, bun technique) that greatly helped improve my research focus and output.

I would also like to thank all my coauthors and collaborators including: Alec Azad, Andrea Bunt, Bill Cowan, Yang Li, Richard Mann, Matei Negulescu, Eric Saund, David Tausky, and Daniel Vogel. I would like to acknowledge the administrative staff, especially Wendy Rush for all her support getting reimbursed for the numerous user studies conducted during my PhD.

While a Ph.D. student I met many new and interesting people who made the time I spent in Waterloo enjoyable. Overall, my colleagues in the HCI lab were top notch, and I especially enjoyed working and chatting with Christine Szentgyorgyi, Richard Fung, Matei Negulescu, Ben Lafreniere, Ryan Stedman, and Adam Forney. I would also thank Greg Zaverucha, John Chapman, Nick Miller, and Jeff Dicker for the good times outside of the lab.

I would also like to thank my good friend Christopher Ferguson for allowing me to stay at his house and use his car on several occasions. Without his generosity, my fruitful collaborations with PARC and Google would not have been possible.

Lastly, I'd like to thank my best friend and wife, Christina Boucher, who has been a relentless source of support for me. She very generously took the time to proofread my papers, introduce me to pottery, and up with me during CHI submission season.

Dedication

This dissertation is dedicated to my loving and supportive parents, Armando and Cathy. Thank you for teaching me at an early age the importance of higher education.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Interfaces on personal computers have evolved from conversational interfaces, i.e. command-line interfaces, to WIMP (Windows, Icons, Menus, Pointer) and direct manipulation interfaces where users use a mouse, electronic stylus, touchpad, trackpoint, trackball, or finger to point at on-screen targets. To facilitate pointing, HCI researchers have proposed techniques that alter the target [7, 24, 29, 44, 72], alter the distance between the cursor and target [3, 5, 25, 31, 32], or change pointing interaction [70]. In his survey of pointing facilitation techniques, Balakrishnan [4] noted that a fundamental assumption made by many of these techniques is that salient targets are relatively sparse on the display and are separated by whitespace. However, researchers have also noted [4, 44] that salient targets are frequently tiled into small regions on the display, e.g. into ribbons or toolbars. As well, in many modern computer programs, such as spreadsheet programs, word processors, and bitmap drawing programs, any cell, character, or pixel might constitute a legitimate target for pointing. Thus, researchers who study pointing facilitation have identified the ability to identify–during motion–the likely target of a user's pointing gesture, as a necessary precursor to pointing facilitation in modern computer interfaces [44, 4]. In addition, researchers [44, 74] have demonstrated that an endpoint prediction technique should identify the user's likely target during the first 90% of gesture motion if the predicton is to be used to facilitate pointing. Therefore, to enable pointing facilitation on modern desktop interfaces there is a need to be able to identify user's endpoint only using the first 90% of gesture motion.

The goal of this thesis is to develop and analyze how an understanding of the underlying characteristics of motion can enhance our ability to predict the target or endpoint of a goal-directed movement in graphical user interfaces.

Research investigating goal-directed movement [71, 46] (e.g., pointing) theorizes that motion occurs in two phases, an initial ballistic phase that is responsible for bringing the hand close to the target, and a homing phase consisting of small corrective submovements that allows the hand to acquire the target (illustrated in Figure 1.1). Influenced by this research, we begin by examining the underlying characteristics of pointing motion in order to gain an understanding of what properties of motion can, and should, be modeled to predict motion distance. More specifically, since our goal is to model pointing motion to predict endpoint, we must determine if the corrective submovements occur early enough to be incorporated into any modeling technique, as the ability to incorporate these submovements will lead to a more accurate predictor. We also examine how constraints within the interface (e.g., target size and intended use) affect the characteristics of pointing motion.

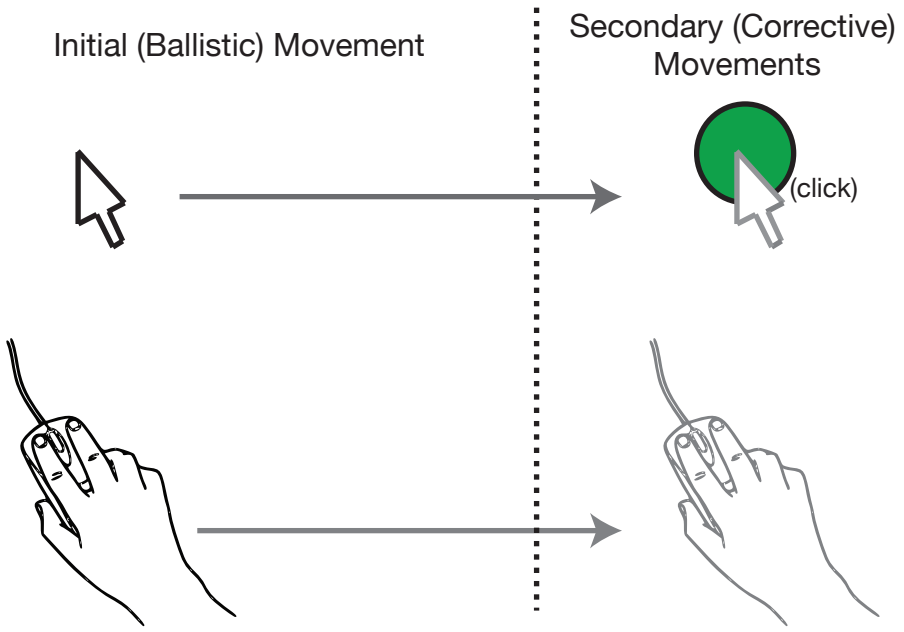


Figure 1.1: An illustration showing the two phases of pointing motion. The initial ballistic phase consumes as much as 90% of the distance traveled and is responsible for bringing the hand/cursor close to the target. The secondary (corrective) phase often occurs during the last 10% of gesture distance and allows the hand/cursor to acquire the target.

Using established laws of motion and an analysis of user's kinematic profiles, we demonstrate that the initial 90% of motion is primarily ballistic and submovements are limited to the last 10% of gesture movement. Through experimentation, we demonstrate

that target constraint (i.e. whether the target consists of an amplitude or steering constraint) and the intended use of a target has either a minimal effect on the motion profile or affects the last 10% of motion. Therefore, we demonstrate that any technique that models the initial 90% of gesture motion will not be affected by target constraint or intended use.

Given these results, we develop a technique to model the initial ballistic motion to predict user endpoint by adopting principles from the *minimum jerk principle* [27], which states that the motor planning of ballistic motion is planned to minimize jerk, i.e. the rate of change in acceleration. Based on this principle, we derive an equation to model the initial ballistic phase of movement in order to predict movement distance and direction. We demonstrate through experimentation that we can successfully model pointing motion to identify a region of likely targets on the computer display. Next, we characterize the effects of target size and target distance on prediction accuracy. We demonstrate that there exists a linear relationship between prediction accuracy and motion distance and that this relationship can be leveraged to create a probabilistic model for each target on the computer display. We then demonstrate how these probabilities could be used to enable pointing facilitation in modern computer interfaces.

1.1 Research Goals and Overview

The research objective of this thesis can be simply stated as:

> *Develop and analyze how an understanding of motion kinematics can enhance our ability to predict the target or endpoint of a goal-directed movement in graphical user interfaces.*

To ultimately reach this goal, we investigate a series of primary research problems, many of which build on (or are dependent on) research outcomes from previous steps. The seven research problem statements are:

(a) What characteristics need to be accounted for when modeling pointing motion to predict gesture endpoint?

(b) How do the constraints in the interface affect motion characteristics?

(c) Can we identify a design space describing the use of endpoint prediction to enable pointing facilitation?

(d) Can we develop a technique to predict user endpoint?

(e) How does target size and target distance affect our ability to predict user endpoint?

(f) What level of prediction accuracy is necessary to improve pointing for specific display configurations that are currently resistant to standard pointing facilitation techniques?

(g) Can an endpoint predictor that models the characteristics of motion be used to enable pointing facilitation techniques?

To answer these research questions, we took the following steps (illustrated in Figure 1.2):

1. To answer (a), we use the minimum jerk principle and examine kinematic profiles collected from participants. Our results show that the first 90% of gesture motion distance is primarily ballistic.

2. To investigate how target constraints in the interface affect motion kinematics (b) we conduct two user studies. The first study examines how target constraint (i.e., whether the target has an amplitude or steering constraint) affects kinematic profiles. Results from this study are two-fold. First, we demonstrate that changes in kinematic profiles occur in the motion perpendicular to the motion gesture and these differences can be detected using the first 70% of motion gesture. However, when examining the speed profiles of the motion, these changes are "washed-out" by the dominant colinear motion. Therefore, target constraint does not have a significant effect on the initial ballistic speed profile of a pointing gesture.

   The second study examines the effects of intended use on motion kinematics. Results from this study demonstrate that any observable and statistically significant temporal or kinematic changes resulting from the task a user wishes to perform once a target is acquired is mainly limited to the final 10% of motion.

3. Using the results from (b), we present an endpoint prediction technique that models the initial ballistic kinematic profile of a pointing gesture to predict gesture length. We answer question (d) by validating the technique using stylus and mouse-based pointing motion. We also provide a refined technique that allows interaction designers to predict endpoint in real-time and calculate the reliability of the prediction.

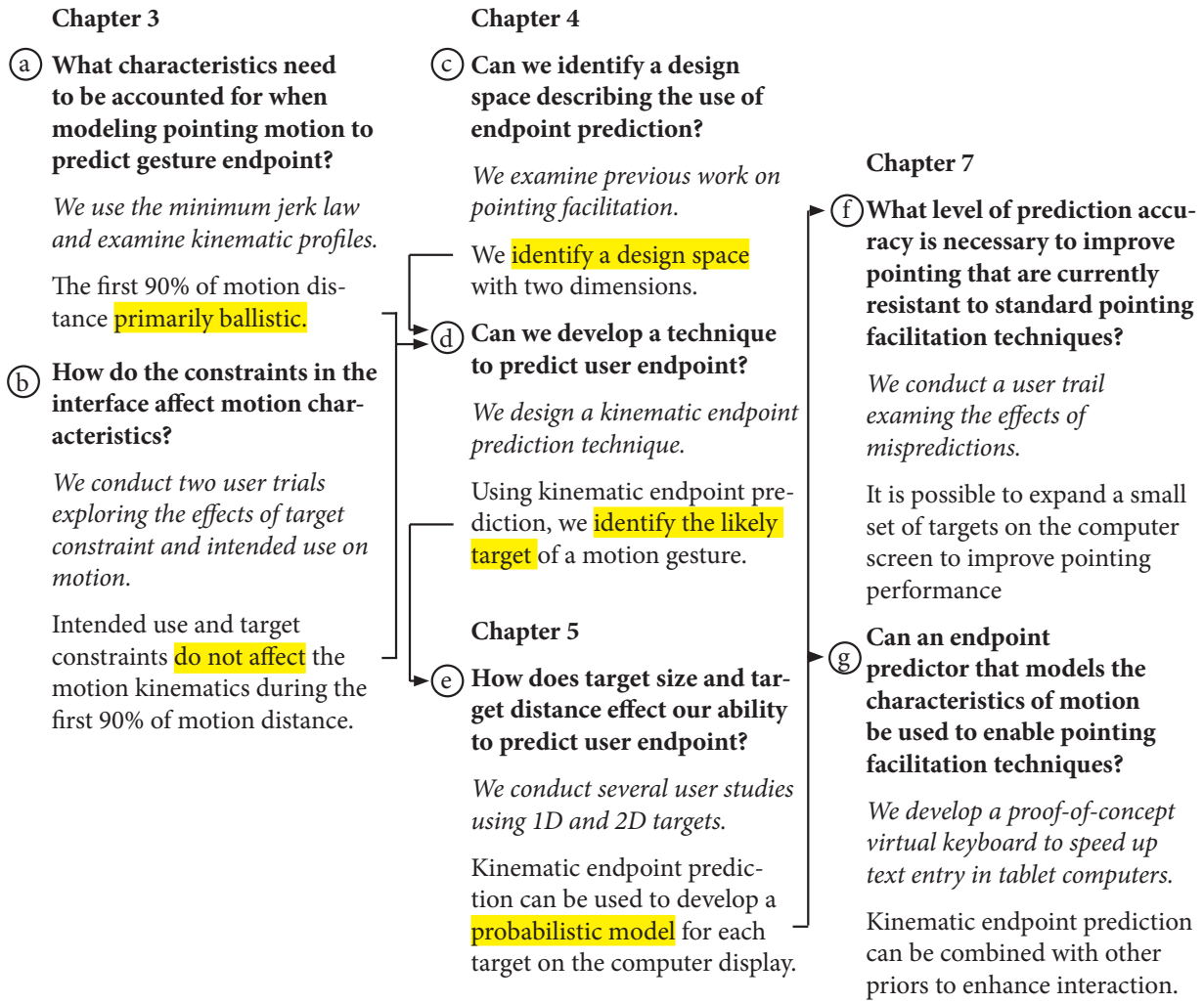


Figure 1.2: Research path showing research problems, activities, and main results. Bold text is the research problem statement; italic text is the research activity; and the final block of text is the primary contribution which leads to the next stage. Highlighted text and arrows illustrate dependencies forming the research path used in this thesis.

4. To answer question (c), we examine proposed pointing facilitation techniques to provide a general design space for endpoint prediction techniques. Using this design space, we illustrate how the variants of our technique can be used to enable pointing facilitation techniques in modern interfaces.

5. Using our technique from (d), we perform several user trials using 1D and 2D targets to answer question (e). Our results demonstrate that there exist a linear relationship between prediction accuracy and motion distance. We then leverage this relationship to create a probabilistic model for each target on the computer display. To demonstrate how these probabilities can be used to facilitate pointing (question (g)), we develop a virtual keyboard that uses out endpoint prediction technique and tetra-gram letter frequencies to incorporate target expansion and visual cues to speed text entry on Tablet PCs.

6. To answer question (f), we conduct a study exploring the effects of misprediction and expanding widgets on pointing performance. Our results allow us to characterize the accuracy necessary to improve pointing for specific display configurations that are currently resistant to standard pointing facilitation techniques.

1.2 Contributions

We make the following contributions to goal-directed pointing in interfaces, endpoint prediction, and pointing facilitation in user interfaces.

1.2.1 Goal-directed motion in computer interfaces

Our investigation into motion profiles and the effects of interface constraints on motion kinematics in Chapter 3 demonstrates that the first 90% of gesture motion is primarily ballistic. We demonstrate that while we can extract additional information about the user's target, specifically the primary direction of the constraints on the user's movement, the speed by distance profile remains unaffected. We also show that any effects of intended use are limited to the last 10% of motion distance.

1.2.2 Predicting endpoint distance using motion kinematics

In Chapter 4, we present a kinematic endpoint predictor that models the initial ballistic motion of a pointing gesture to predict gesture endpoint. We also provide a method to

measure the stability of the technique that allows interface designers to quantitatively measure the quality of a prediction provided by the technique. Using an implementation of the kinematic endpoint predictor technique, we demonstrate the technique can be used to predict motion distance for both stylus and mouse-based motion.

Through the experiments presented in Chapter 5, we characterize the effect of movement distance on the accuracy of the kinematic endpoint predictor for both one and two-dimensional targets and measure the effects of cursor acceleration on kinematic endpoint prediction. We leverage this characterization to develop a probabilistic model for each target on the computer display.

Using results from Chapters 3 and 5 and results from the psychology and kinesthetics literature, Chapter 6 provides a discussion about the limitations of modeling ballistic motion to predict user endpoint and argues that our kinematic endpoint predictor provides the optimal accuracy of a predictor based on ballistic motion.

1.2.3 Pointing facilitation and endpoint prediction

In Chapter 4, we define the design space for endpoint predictors in relation to the proposed pointing facilitation techniques presented in Chapter 2. Using this design space we present two variants of our endpoint prediction technique.

Results from our user study presented in Chapter 7 characterizes the accuracy necessary to improve pointing for specific display configurations (e.g., tiled target arrangements) that are currently resistant to standard pointing facilitation techniques. Using the probabilistic model presented in Chapter 5, we develop a proof-of-concept virtual keyboard that demonstrates how kinematic endpoint prediction can be used in modern graphical interfaces to facilitate pointing (Chapter 7).

1.3 Dissertation Outline

The remainder of this document is organized as follows:

In chapter 2, we summarize relevant background work on modeling and predicting goal-directed movement and proposed pointing facilitation techniques. We then highlight the open questions regarding pointing in computer interfaces and pointing facilitation that this thesis addresses.

In chapter 3, we describe the characteristics of motion found in the first 90% of motion and present user trials examing how target constraints and intended use affect motion kinematics.

In chapter 4, we evaluate the design space for endpoint predictors and present a technique that models the initial ballistic movement in a pointing gesture to predict gesture endpoint.

In chapter 5, we present several user studies that characterize predictor accuracy for 1D and 2D targets as target distance and target width changes. We also demonstrate the effects of cursor acceleration on our ability to predict endpoint.

In chapter 6, we describe the implications of our results to the motor control and endpoint prediction literature.

In chapter 7, we explore the effects of mispredictions, demonstrate that the predictor can be used to visually expand a subset of targets, and present a proof-of-concept virtual keyboard that illustrates the utility of our endpoint predictor.

In chapter 8, we draw conclusions, summarize limitations, and suggest possible future work.

Chapter 2

Related Work

In this chapter, we present previous work on modeling and predicting goal-directed movement and pointing facilitation techniques. We begin by presenting the temporal prediction models that are often used by Human-Computer Interaction (HCI) researchers to evaluate interfaces and interaction techniques. Next, we describe an overview of motor-control motion models from disciplines outside of HCI. This is followed by an overview of literature exploring the effects of intended use on the kinematic profiles of motion. We then switch focus and present previous research on pointing facilitation techniques and endpoint prediction. We conclude this chapter with a discussion of some open questions that need to be addressed before modeling motion to predict user endpoint.

2.1 Movement Time Prediction Models

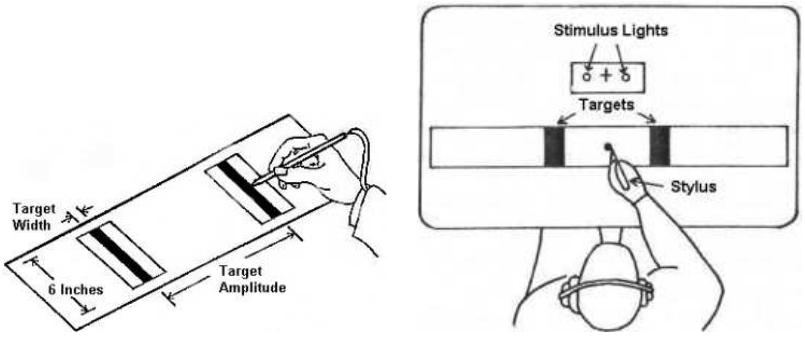
Prediction models, especially Fitts' Law [18], have become an important research tool in HCI. In addition to predicting movement times [13, 11, 37], prediction models have also been used to evaluate input devices [11], evaluate interaction techniques [13, 44, 57], and motivate target acquisition (see [4] for review). In this section, I discuss Fitts' Law and describe extensions relevant to computer interfaces.

2.1.1 Fitts' Law

Fitts' Law is one of the most robust and highly adopted models of human movement [12]. Influenced by Shannon's Theorem 17 on information theory [63], Fitts argued that the

amplitude of an aimed movement is analogous to an electronic signal, and that the spatial accuracy of the movement is analogous to electronic noise. Furthermore, he proposed that the human motor system is like a communication channel, wherein movements are viewed as the transmission of signals.

Fitts presented his analogy in two highly celebrated papers, one in 1954 [18] and the second in 1964 [19]. In the first paper, Fitts described a reciprocal target-acquisition task, wherein, subjects alternatively tapped on targets of width W separated by amplitude A (Figure 2.1(a)). The second paper described a similar experiment using a discrete task, where subjects selected one of two targets in response to a stimulus light (Figure 2.1(b)).



(a) (b)

Figure 2.1: Experimental setup for Fitts' experiments: (a) the original serial task from Fitts 1954; (b) the discrete task from Fitts' 1964 follow-up. (Reproduced from [12])

From his studies, Fitts quantified a movement task's difficulty, known as the index of difficulty (ID). Specifically,

$$ID = log_2 \left(\frac{A}{W}\right) \tag{2.1}$$

where A represents the amplitude of the movement (i.e. the distance) and W the width of the target. Using Equation 2.1, the movement time (MT) to complete a task is predicted using the simple linear equation:

$$MT = a + b \times ID \tag{2.2}$$

where a and b are empirically defined constants. Although using Fitts' Law in computer interfaces was initially proposed by Card et al. [11], MacKenzie's dissertation [37] is credited for showing the reliability and robustness of Fitts' Law for predicting pointing times in computer interfaces. In addition, MacKenzie also introduced a revision to Fitts'

original equation for ID to improve the information-theoretic analogy. MacKenzie's revision, known as the Shannon Formulation, shown below has become the accepted equation for calculating ID within computer interfaces.

$$ID = log_2\left(\frac{A}{W} + 1\right) \tag{2.3}$$

2.1.2 Extensions of Fitts' Law

Bivariate Pointing

As described above, Fitts' original experiments consisted of targets that were constrained in one dimension. However, targets in graphical user interfaces are two-dimensional which results in targets having both amplitude and directional constraints as illustrated in Figure 2.2. An amplitude (or stopping) constraint, pictured on the left, exists when the height of the target is greater than the width of the target. With an amplitude constraint, the distance traveled must be controlled. A directional (or steering) constraint, pictured at the right, exists when the width of the target is greater than the height. The user needs to steer into the region of the target, i.e., to control the direction.

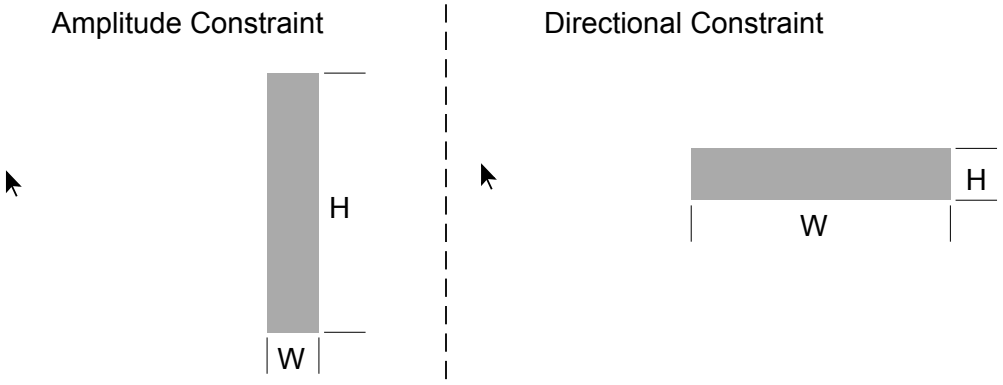


Figure 2.2: Pointing task constraints. On the left, an amplitude or stopping constraint. On the right, a directional or steering constraint.

To examine the influence of bivariate constraints on movement time, Accot and Zhai [2] performed a study varying the ratio of width and height on targets at different distances. From their observations, Accot and Zhai concluded that amplitude and steering constraints

do not have an equal effect on movement times. They observed that when target width (amplitude constraint) is less than target height (steering constraint), height has no influence on movement times. However, when target height is the limiting constraint (i.e. height is less than width), target width still has an effect on movement time. Based these results, Accot and Zhai proposed the following modification of Fitts' Law to address bivariate pointing:

$$T = a + blog_2\left(\sqrt{\left(\frac{A}{H}\right)^2 + \eta\left(\frac{A}{W}\right)^2} + 1\right) \tag{2.4}$$

where a, b, and η are empirically determined constants and a varies approximately in the range of [-50,200], b in [100,170], and η in [1/7, 1/3].

2.1.3 Probabilistic Predictive Model of Pointing

The probabilistic predictive model of pointing introduced by Grossman and Balakrishnan [23] is an attempt to extend Fitts' Law beyond the variables of width and amplitude. In particular, Grossman and Balakrishnan were interested in being able to measure ID in terms of target width, target height, and *approach angle*, the angle at which the user moves towards the target. The probabilistic predictive model claims that in addition to target size and distance, approach angle has an effect on a target's ID by either making the target's constraining dimension larger or smaller. Therefore, the probabilistic predictive model proposes using the region defined by the target in relation to the direction (i.e., approach angle) of the pointing motion. This target region is then used to determine a target's ID by calculating the probability of hitting the region using an open movement (i.e., the initial movement without making any corrections). Regions with a higher probability (i.e., requiring no or few corrections) have a lower target ID compared to those with a lower probability (i.e, requiring several corrections).

2.1.4 Temporal Models and Endpoint Prediction

In this section, we presented several temporal models found in the literature that aim at predicting the time required to move and acquire a target. While these temporal models, for example Fitts' Law, accurately predict movement time, they are unable to model trajectory of a gesture, but instead model the entire task. Therefore, these models cannot reliably segment a task into phases, making them unsuitable for the purpose of endpoint prediction.

2.2 Models of Motor Control

In this section we discuss different proposed models based on motor control. Although often influenced by temporal models, motor control models focus on describing the underlying characteristics of motion rather than predicting movement time. We begin the section with a thorough discussion of models for voluntary movement and Woodworth's pioneering study [71] on speed-accuracy trade-offs. This is followed by a discussion of models describing ballistic and curved movements.

2.2.1 Voluntary Movement

Voluntary movements are often referred to as goal-directed movements, where the movement is the means of accomplishing a goal, i.e., a change in the state of the environment, or the goal itself. For example, a common goal of goal-directed movements in computer interfaces is to move the on-screen pointer to a specified location by means of moving a physical device (e.g., a mouse).

Woodworth's Two-Component Model

Woodworth [71] is credited as the first researcher to examine goal-directed movement. Woodworth's doctoral dissertation, published as a monogram, reports on a number of studies examining the perceptual, cognitive, and motor processes associated with performing precise, goal-directed action. The paper made important empirical and theoretical contributions in diverse areas such as speed-accuracy trade-offs, manual asymmetries in motor control, coordination, movement perception, and motor learning.

Impressed by the speed and accuracy of construction workers hammering nails, Woodworth wondered how the workers could achieve the speed and accuracy they did. To answer this question, he designed an experiment in which participants used a pencil to draw lines back and forth between two target lines that were separated by a fixed distance. To record the subjects' movements, the target lines were drawn on paper that was secured to a rotating drum. To control the participants' movement time, Woodworth used a metronome to specify the speed at which participants were required to move back and forth between the target lines. This setup allowed Woodworth to measure not only the accuracy and consistency of movement but also the spatial and temporal characteristics of the movement trajectories.

Woodworth found that all but the fastest movements are made up of two components which he termed the *initial adjustment phase* and the *current control phase*. The initial adjustment phase of the movement is relatively rapid and has the purpose of bringing the limb into the vicinity of the target. This phase is often referred to as the *ballistic phase* or the *initial impulse* in succeeding research. The current control phase, the second half of the movement, varies to a greater degree with each aiming attempt. Woodworth hypothesized that during this phase of the movement the participants adjust the movement in order to hit or come close to the target. That is, participants add "little extra movements" to the initial impulse, or there is "a subtraction or inhibition of the movement, making it shorter than it otherwise might have been" [71]. The idea is that, during this portion of the movement participants use visual and other forms of feedback to reduce aiming error inherent in the initial impulse. When visual feedback is unavailable, either due to eyes being closed or the movement happening too quickly such that time needed is less than that required for visual feedback, the movement only contains the initial impulse phase resulting in decreased accuracy.

Woodworth's two-component model of goal-directed action has had a profound influence on contemporary accounts of motor control and learning [15].

Iterative Corrections Model

The *iterative corrections model* [14, 30][1] is an attempt to explain the relation between movement time and target amplitude and width as described by Fitts' Law. Crossman and Goodeve hypothesized that Fitts' Law is mainly attributed to current control. According to the model, an aiming movement consists of a series of discrete submovements, each of which is ballistic and triggered by feedback that the target has yet to be attained. Each submovement takes the hand (or a handheld stylus) a fixed proportion of the distance to the target. Therefore, as the width of the target decreases, the hand falls within the target later in the series of submovements. Similarly, as the distance of the target increases (for a given target width) the first submovement for which the hand falls within the target is also delayed. Qualitatively the model accounts for the relationship implied by Fitts' law. Quantitatively, the model predicts a linear increase of total movement time with index of difficulty, provided one assumes that each correction takes a constant amount of time [30].

While the model did a good job of mathematically describing the relationship between movement time and endpoint error [19], the model was invalidated as technology allowed

[1] Crossman and Goodeve [14] are credited for proposing the Iterative Corrections Model, however, it was follow-up work by Kelle [30] that is often referred to when describing the model.

the recording of kinematic profiles. According to the model, the kinematic profiles of movement should show normalized and distinct movements corresponding to each submovement. However, this is not the case as movements did not always show distinct velocity peaks and durations were not always consistent [54].

Impulse Variability Model

Although Woodworth's two-component model and the iterative corrections model differ in their hypothesized processes, fundamental to both models is the role of visual feedback in determining endpoint accuracy. An alternative to both of these models is Schmidt's impulse variability model [59, 60] that states endpoint variability is independent of feedback processing and is related to the muscular forces required to accelerate and decelerate the limb movement. Schmidt proposed that rapid arm movements are achieved by flinging the arm toward a target. The flinging is achieved with a neuromotor impulse delivered to the arm muscles. The impulse causes the muscles to exert a burst of force for the first half of the movement time. During the second half of the movement time, the limb passively moves towards the target. The model assumes that there is variability in the forces driving the arm toward the target as well as variability in the the time during which the forces are produced. The standard deviation of force is assumed to be proportional to the amount of force, and the standard deviation of the time is assumed to be proportional to the time during which the impulses are delivered. Thus, if more force is used to cover a larger distance, more force variability results, and if more time is spent propelling the limb toward the target, more time variability results as well. Because time and force can be independently controlled in the model, the participant's challenge is to find the time and force that minimizes the variability of both factors. According to Schmidt et al., Fitts' law represents the solution to this problem.

Although the impulse variability model does a good job at predicting the relationship between movement time and effective target width (i.e., variable error) for very rapid movements [60] and movements made without visual feedback [67], the model begins to break down for movements where participants have the ability to respond to any visual feedback present. The main contribution of the impulse submovement model is that it provided a platform for what has become the stochastic optimized submovement model, which incorporates aspects of iterative corrections models and the impulse variability model.

Stochastic Optimized Submovement Model

The stochastic optimized submovement model is a hybrid of the iterative corrections model and the impulse variability model. The starting point for the model is shown in Figure 2.3. By hypothesis, the subject makes a first movement towards the target. If the movement lands within the target, the task is complete. If the movement lands outside of the target, another movement is needed, which again can land inside or outside of the target, and so forth. The subject's task is to reach the target as quickly as possible, so ideally the subject should make just one, high-velocity movement directly to the target. The problem is, according to the model, the spatial accuracy of movements are imperfect. The standard deviation, S_i, of the endpoint of any movement (i) is assumed to increase with the distance (D_i) covered by that movement and decrease with its duration, T_i. In equation form,

$$S_i = k\,(D_i/T_i) \tag{2.5}$$

where k is a constant. To get to the target as quickly as possible, one should make a movement with long distance (large D) and short time (small T), but this would result in a high standard deviation and a low probability of hitting the target. Alternatively, the subject could make a movement with long duration and a series of short movements. However, this would result in very long movement times. The optimal solution is to find a balance between D's and T's that minimizes the total movement time. According to Meyer et al. [46], Fitts' law represents such an optimal balance.

Summary

In this section we introduced several theories researchers have proposed to describe the underlying characteristics of motion of a goal-directed movement. While these theories may differ in describing the movement, they are similar in that they all propose that the initial movement is ballistic in nature. They differ in the means and the role feedback plays to acquire a target. Of the models presented in this section, the stochastic optimized submovement model is currently the accepted motor control model for goal-directed movements and influences our work in endpoint prediction presented in this dissertation.

In the next section, we describe models that aim at quantifying the kinematics of motion.

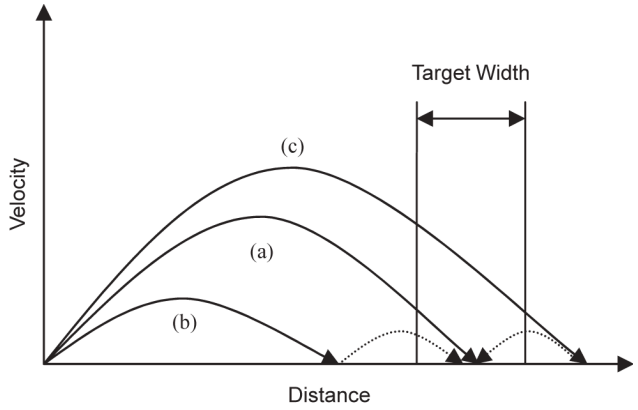


Figure 2.3: Possible sequences of submovements toward a target. (a): A single movement reaches the target. (b) and (c): The initial movement undershoots or overshoots the target, requiring subsequent corrective movements (dotted curves). Adapted from [46].

2.2.2 Point-to-Point Movement and Curved Movement

Influenced by previous work on single joint movements in primates, Flash and Hogan [20] set out to mathematically describe the kinematics of unconstrained point-to-point movements in human subjects. Flash and Hogan argued that, as demonstrated in primate motion, human movement is the result of an implicit optimization of a derivative of motion. Through experimentation, Flash and Hogan demonstrate that jerk, the third time derivative of position, is minimized during movement. In other words, movement trajectories of unconstrained multi-joint movement are planned to minimize jerk, the rate of change of acceleration. Assuming at the start of a movement the acceleration and velocity are zero, the hand trajectory of the movement can be modeled by:

$$x(t) = x_0 + (x_0 - x_{final})(15\tau^4 - 6\tau^5 - 10\tau^3) \tag{2.6}$$

$$y(t) = y_0 + (y_0 - y_{final})(15\tau^4 - 6\tau^5 - 10\tau^3) \tag{2.7}$$

where $\tau = t/t_{final}$, the normalized period of the gesture. x_0 and y_0 are the initial hand position coordinates at $t = 0$, and x_{final} and y_{final} are the final hand position coordinates at $t = t_{final}$. Flash and Hogan's model is often referred to as the *minimum jerk principle* in subsequent research.

Lacquaniti et al. [33, 34] introduced the two-thirds power model to describe angular velocity of movement in drawing tasks. The two-thirds power model states that angular

velocity is a function of the curvature of the movement, such that,

$$A(t) = kC(t)^{2/3} \tag{2.8}$$

where $A(t)$ denotes the angular velocity at time t, k is an empirical constant, and $C(t)$ is the curvature at time t.

Follow up work by Viviani and Flash also demonstrated that the minimum jerk model can also describe more complex movements [66], such as point-to-point movements through a middle point. They also argue that the two-thirds power and minimum jerk model are not conflicting models but instead are complimentary to quantifying the kinematics of several types of motion.

2.3 Effects of Intended Target Use on Kinematic Characteristics

Marteniuk et al. [42] investigated the effects of motion constraints on movement time and the kinematic profile of motion when pointing (i.e. touching) and grasping physical objects. Marteniuk et al. observed faster movement times for pointing than grasping. The observed differences were a result of both longer acceleration and deceleration phases during grasping movement. The researchers also found that the mean speed before making contact with the target was significantly higher when pointing than when grasping. Finally, Marteniuk et al. found that the perceived affordances of the target had an effect on the kinematic profile of movement. When asking subjects to grasp a tennis ball or light bulb, Marteniuk et al. observed that movement times and the deceleration phase were significantly longer for the light bulb versus the tennis ball.

Intention has also been shown to have a significant effect on motion kinematics and observed time. Marteniuk et al. [42] showed that movement times and motion profiles differed depending on if an object was to be thrown or carefully placed after grasping. Follow up work by Rossenbaum and his colleagues also supported and extended these findings, showing that the perceptions of affordances and the intended use of an object both significantly affect grasping movement [55].

Recently, Mandryk and Lough [41] extended the work of Marteniuk et al. [42] to 2D pointing in user interfaces. When pointing at targets in a computer interface, users can perform a number of different actions. They can click on the target, i.e. press and release the mouse button over the target. They can also double click the target by pressing and

releasing the mouse button two times within a short time period. Beyond clicking actions, users can also drag the target in various ways. For example, they can dock an object they are dragging by depressing the mouse button over a target and then move the mouse (with the button down) to a new location on the screen and place the object within a constrained region. They can also flick the object by rapidly and imprecisely dragging the object to a new location.

Mandryk and Lough looked specifically at the effect intended use, i.e. single target, dual target, flick and dock, have on movement times to the primary target of goal directed movement. In their experimental design, they measure movement time from the moment a user clicks within a start location until the moment when a user depresses the mouse button over the primary target, i.e. the target of their first targeting movement. Once the user has depressed the mouse button over the primary target, they then perform one of the three secondary tasks.

Mandryk and Lough make several observations. First, they show that, for their experimental configuration, flick and dock take significantly longer than single target and dual target. They do, however, note that their flick task may have been more difficult than they anticipated it should be, based upon comments from their participants [41, p. 1652]. As a result, flick data in their research may not be reliable. Second, Mandryk and Lough divide motion toward the target into two phases. The first phase, the acceleration phase, occurs from the time a user clicks within the start location until peak speed. The second phase, the deceleration phase, occurs from peak speed until the moment a user depresses the mouse button within the primary target, before beginning the secondary task. They show that the deceleration phase of movement is shorter for both single target and dual target. They also show that peak velocity is statistically faster for flick than for single target or dual target.

The researchers note that one significant implication of their work is that there appears to be a difference in the acceleration phase of movement (i.e. peak speed is higher for the flick task) and the deceleration phase of movement (i.e. in fraction of time spent after peak speed) depending on intended use of the primary target. The implication that they draw from these observations is that the differences in motion may affect interaction techniques that depend on an analysis of kinematics.

2.4 Pointing Facilitation

Pointing, with a mouse, electronic stylus, touchpad, track-point, or trackball, is a frequent task in modern graphical user interfaces. Due to the frequency of pointing, even a

marginal improvement in pointing performance can have a large effect on a user's overall productivity. As mentioned above, several researchers have demonstrated that pointing with an input device is predicted by Fitts' Law. However, unlike physical pointing, virtual pointing (i.e. pointing with an input device on a computer) is not constrained by the laws of the physical world. HCI researchers have proposed several techniques to "beat" Fitts' by manipulating, as indicated by Fitts' Law, the amplitude of the movement, A, or the size of the target, W. However, as mentioned by Balakrishnan [4], changing these two parameters obviously does nothing more than change the size and position of on-screen graphical elements, which are presumably already laid out in a reasonably optimal fashion due, in part, to the interface designer's basic appreciation of Fitts' Law. Therefore, the challenge becomes to indirectly affect further changes in A and/or W in ways that do not substantially alter the overall visual appearance of the graphical interface, but nevertheless results in shorter pointing times. Techniques in pointing facilitation can roughly be grouped into three categories: those that (1) primarily attempt to decrease A, (2) primarily attempt to increase W, and (3) both decrease A and increase W. Table 2.1 outlines proposed pointing techniques for each category. We also discuss each category in greater detail below.

2.4.1 Facilitating pointing by primarily decreasing A

Facilitation techniques that attempt to primarily reduce the distance between the display cursor and the target obviously only have two options: bring the target closer to the cursor, or move the cursor closer to the target. Techniques that bring targets closer to the cursor include pie-menus [10] and Baudisch et al.'s drag-and-pop [5]. The principle behind pie-menus is to minimize the distance needed to traverse items of a contextual menu by redesigning the menu from a linear vertical list to a circle around the cursor (Figure 2.4). Drag-and-pop is a technique proposed by Baudisch et al. [5] where the system responds to the directional cursor movements by temporarily bringing a virtual "proxy" of the most likely potential set of targets towards the cursor (shown in Figure 2.5).

An example of a technique that brings the cursor closer to the target is object pointing. In *object pointing* [25], pixels that serve no purpose other than to visually space potential targets are "skipped" as the cursor moves over them, thus effectively making the cursor act as if the targets are next to one another.

Category	**High-Level Strategy**	**Selection Technique**	**Eases selection of tiled targets?**
Decrease A	move cursor to target (or ease movement toward target)	pie-menus Callahan et al. [10]	no
		object pointing Guiard et al. [25]	no
		snap-to cursor Sutherland [65] Fiener et al. [17] Bier & Stone [6]	no
		haptic feedback Munch & Dillmann [45] Oakley et al. [48, 47]	possibly
		Multiple cursors Kobayashi & Igarashi[32] Blanch & Ortega [8]	no
	move target closer to cursor	drag-and-pop Baudish et al. [5]	no
Increase W	make cursor bigger	area cursor Kabbash & Buxton [29] Hoffmann [26] Worden et al. [72]	no
		bubble cursor Grossman & Balakrishnan [24]	no
	make target bigger	expanding targets McGuffin & Balakrishnan [43, 44] Zhai et al. [74]	possibly
		semantic pointing Blanch et al. [7]	possibly
Increase W & Decrease A		C-D Ratio adaptation Keyson [31] Worden et al. [72] Blanch et al. [7]	no

Table 2.1: Various techniques for facilitating pointing in interfaces broken down by the category and the high-level strategy they employ to reduce A or increase W.

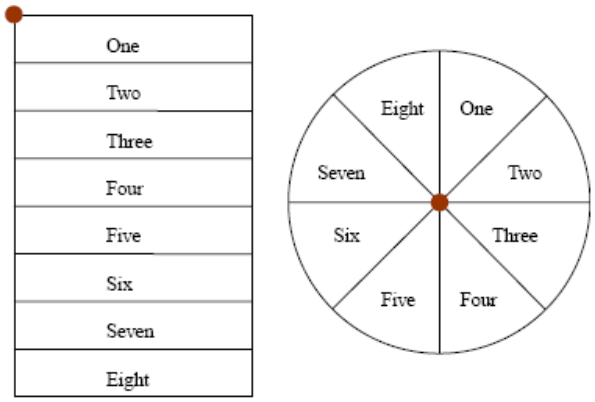


Figure 2.4: Linear vs pie menus. Distance of menu items from the red starting point varied in linear menus but is constant in pie menus. Adapted from [4].

Figure 2.5: Baudisch et al.'s Drag-and-pop [5] technique. Virtual proxies of icons on the far left of the screen are brought closer to the cursor to facilitate quick pointing by reducing A. The relationship between proxy and actual icon is indicated by the stretched lines. The proxies only exist for the duration of the cursor drag action, thus not affecting the overall interface. Adapted from [5].

2.4.2 Facilitating pointing by primarily increasing W

Target expansion is a popular technique designed to increase a target's width. Target expansion works by expanding a user's target on the display so the target is easier to acquire. These expanding widgets are displayed at significantly reduced size by default and dynamically expanded to a usable size only when required. Target expansion is an effective

strategy for maximizing the use of screen real estate without sacrificing target selection performance. McGuffin and Balakrishnan [43, 44] and Zhai et al.'s [74] investigations into expanding targets demonstrated that a target's index of difficultly is measured by the final target size (i.e. the expanded target size) and that a user can take advantage of an enlarged target even when the expansion occurs after 90% of the distance to the target has already been traversed.

Another technique to increase a target's activation size is *area cursors* [29, 72]. Area cursors is a technique where the activation area of the cursor is larger than the standard cursor. Kabbash and Buxton [29] showed that when the width of a target is less than the width of the cursor, the index of difficulty of the pointing task is accurately modeled by the width of the cursor rather than the target width. Therefore, for very small targets an enlarged cursor speeds pointing by reducing the index of difficulty.

2.4.3 Facilitating pointing by both decreasing A and increasing W

The ratio of the amount of movement of an input device to the movement of the controlled object (i.e. typically a cursor) is referred to as the Control-Display (C-D) gain. C-D gain manipulation is based on the observation that target width and distance in motor space more accurately predicts performance than the widget's visual distance and size [7, 9]. Therefore, using adaptive C-D gain a system can require smaller motor space movement when traveling to a target while still having an enlarged motor space region for target activation without modifying the target's appearance on the display. One common use of C-D gain is mouse acceleration in modern operating systems. Mouse acceleration works on the assumption that the user moves quickly when they intend to cover large distances on the display, and more slowly when they are covering shorter distance. A more elegant approach to dynamic C-D gain is to take into consideration the location and size of potential selectable targets. For example, sticky targets [31] uses adaptive gain within targets resulting in the user traveling further to exit a target causing the targets to appear "sticky".

2.4.4 Limitations of proposed techniques

A fundamental flaw of all the techniques presented above, and those listed in Table 2.1, is the assumption that targets are sparsely arranged on the screen. Techniques that decrease the distance to a target (A) take advantage of the empty space between a target and the

cursor by either "jumping" the cursor to the target or moving the target to the cursor. When this empty space no longer exists due to dense and/or tiled target arrangements, these techniques are no longer able to minimize the distance to the target and provide any benefit. Similarly, techniques that increase a target's dimensions (i.e., increase W) also rely on there being empty space to expand into. When this space is not present, expanding all the targets in a user's path will result in either the targets being occluded or the distance between the cursor and the user's intended target being increased. In the case of targets being occluded, the user is now unable to acquire their intended target if the target has been occluded by an enlarged target. If instead the enlarged targets push their neighboring targets away, the distance a user is required to travel is increased resulting in any potential benefits of having a larger target being negated by having to travel further to reach the intended target. Those technique that manipulate both A and W are susceptible to both the conditions described above.

Given that researchers have noted salient targets are frequently tiled into small regions on the display[7, 44], i.e. into ribbons or toolbars and the fact that in many modern computer programs, such as spreadsheet programs, word processors, and bitmap drawing programs, any cell, character, or pixel might constitute a legitimate target for pointing, a majority of the proposed pointing facilitation techniques will not provide any benefit in modern graphical user interfaces. However, there are some techniques (indicated in Table 2.1) that may provide pointing facilitation in tiled targets with some caveats.

As noted by McGuffin and Balakrishnan [44] and Zhai et al. [74], when targets are densely arranged on the screen, expanding or varying the C-D gain of all targets in a user's path results in no pointing advantage. Therefore, with dense or tiled target arrangements, performance gains are only possible if one could reasonably predict the trajectory of the cursor such that the system can identify, during user motion, the likely target of a user's pointing gesture [44]. We call this process *endpoint prediction*, and it is the focus of the research described in this thesis.

2.5 Endpoint Prediction

As mentioned above, there is an intrinsic need for current proposed interaction techniques to be able to identify the likely target of a user's pointing gesture. However, relatively little human-computer interaction work has been performed on endpoint prediction. The current techniques exist in two forms, both of which involve linear extrapolation. The first technique uses peak movement velocity as a basis point for linear extrapolation, while the

second involves linear extrapolation near gesture endpoint (during the last 10% of gesture motion).

Endpoint prediction using peak velocity is a two stage process. First, an interface must be tuned to aid the system in identifying peak velocity, and the typical location of peak velocity for individual users. By identifying peak velocity, looking at the gesture length prior to reaching peak velocity, and multiplying the distance to peak velocity by a scale factor, Asano et al. predict target location [3]. They use their algorithm to speed pointing by jumping the cursor from a position just beyond peak speed to its predicted motion endpoint. However, they judge their technique effective only for distances over 800 pixels on a 1024 X 768 resolution display. At shorter distances, for example, 500, 600, and 700 pixels, their technique does not speed pointing. They do not report predictor accuracies but previous psychology research [15, p. 33 – 34] suggests there is no relation to distance traveled at peak velocity and final distance.

In their work on expanding targets, McGuffin and Balakrishnan [44] develop a simple target predictor based on 3-point linear extrapolation to a future point where velocity is 0, and analyze the accuracy of their estimation technique for tiled target arrangements. If, during extrapolation, a 0-velocity point exists, and the distance remaining to that point is less than 10% of the distance traversed to the current point in the gesture, the algorithm predicts endpoint based on linear decelleration from the current point. Using this predictor, with only 9% of gesture remaining, they predict final target on a tiled button bar 21% of the time, are off by one button 26% of the time, and are more distantly incorrect 53% of the time.

2.6 Summary and Open Questions

In this chapter, we described past research related to pointing, pointing facilitation and endpoint prediction. We began by describing temporal models of motion that aim at predicting the time required to complete a particular motion. Next, we presented kinematic models of motion. Unlike temporal models, kinematic models aim at describing the underlying kinematics and processes of goal directed motion. We then presented work examining the effects of intended use on motion kinematics. This was followed by an overview of pointing facilitation techniques in the HCI literature and a discussion highlighting the need to be able to identify the likely target of a user's pointing gesture in order to enable these techniques in modern interfaces. We concluded this chapter by describing the current techniques for identifying user's endpoint in the HCI literature.

A goal of this thesis is to present a technique that uses motion kinematics to infer user endpoint of pointing to facilitate pointing. As mentioned above, researchers have identified that any prediction technique that aims at identifying a user's intended target must do so before 90% of motion distance. Based on the previous work presented in this chapter, there are two preliminary questions that must be addressed before proposing techniques that model motion to infer endpoint. First, when modeling the first 90% of motion distance, is it possible and necessary to model submovements? Given the various two-component models of motion mentioned above, the goal of the initial ballistic phase of motion is to primarily move get the hand close to the target and the second "homing" phase (i.e. the phase containing submovements) is responsible for acquiring the target. While incorporating submovements into an endpoint prediction technique will increase the likelihood of identifying the user's intended target, do these submovements occur early enough (i.e. before 90% of gesture motion) to be incorporated into any endpoint prediction technique?

Secondly, studies have shown that target constraint (i.e. whether a directional or amplitude constraint) and intended use have a significant effect on movement time. However, little is known about the effects these constraints have on motion kinematics. As mentioned above, Accot and Zhai [2] demonstrated that amplitude and directional constraints do not have an equal effect on movement times. However, the researchers do not explore what effects bivariate targets have on motion kinematics.

Similarly, work by researchers examining the effects of intended use (e.g., [41, 42]) have demonstrated that the intended use of the target affects the movement time of the gesture. In physical grasping and pointing, researchers have demonstrated that the a person's perceptions of an object's affordances and the intended use of an object both significantly affect the later phases of a grasping movement (i.e. the deceleration phase of motion). However, research examining the effects of intended use on motion characteristics in virtual interfaces is relatively new. While Mandryk and Lough [41] have shown differences in movement time may exist, they provide little data on how (or even whether) Fitts' Law models movement time for variations in intended use. Assuming that the linear relationship between movement time and ID is preserved for different intended uses, whether we observe changes in ID, IP, or a in the Fitts' Law equation (Equation 2.3) is an open question. In addition, in Mandryk and Lough's original work the deceleration phase of movement encompasses all user action from peak speed until they depress the mouse over the target. The researchers provide little guidance on where, during deceleration, the changes in movement time occur. Is the entire deceleration curve affected? Or does the user simply spend a bit more time hovering over the target after movement stops before pressing the mouse button to begin his or her secondary task?

Given the results demonstrating the effects of target constraint and intended use on movement times, it is imperative to determine how the underlying characteristics change as a result from these factors. A clearer understanding about how the underlying kinematics change will only allow us to create a more accurate predictor.

In the next chapter, we examine each of these open questions as our preliminary work into using motion kinematics for endpoint prediction.

Chapter 3

Modeling Goal-Directed Motion

3.1 Introduction

In this chapter, we address the two preliminary questions proposed at the end of the last chapter. First, is it possible to incorporate the modeling of submovements into techniques that aim at identifying user endpoint using the first 90% of motion? Second, how does target constraint and the intended use of a target affect the motion kinematics of the movement?

We begin this chapter with a discussion on the practicality of modeling the first 90% of motion. In particilar, we examine if it is necessary and/or possible to incorporate the modeling of corrective submovements into techniques that aim to predict user endpoint based on motion kinematics. We begin by examining the importance of the 90% motion distance threshold demonstrated by various HCI researchers [44, 74, 7] both with theoretical data and also by examining motion profiles from real users.

After demonstrating that the the first 90% of motion is primarily balistic, we examine the effects of target constraint and intended use on kinematic profiles. Previous work by Accot and Zhai [2] and Mandryk and Lough [41] have demonstrated that target constraint and how a user plans on using a target once they acquire it affects movement time. However, these researchers have not fully described the underlying effects these constraints have on the characteristics of motion. We assume there are underlying changes in the kinematics and any changes in kinematics must be accounted for in any technique that aims to model pointing motion. In this chapter, we present studies that analyze the effects of amplitude and directional target constraints and intended use on motion kinematics during pointing

tasks. We will show that while kinematic changes are can be detected, they either do not have a significant impact on the effect of the initial ballistic motion or occur at the end of the motion.

Upon the conclusion of this chapter, We will have demonstrated that any for pointing in interfaces, the ballistic phase of motion occupies at least the first 90% of the motion distance. Therefore, any endpoint prediction technique that wishes to predict user motion before 90% of motion must model the ballistic motion.

3.2 Characterizing the First 90% of Pointing Motion

In work on expanding targets, both Zhai et al. [74] and McGuffin and Balakrishnan [44] note that time taken for targeting depends on final target size, not initial size, even if expansion occurs as late as at 90% of the total distance traversed by a gesture. To understand why this late prediction is possible we reexamine the stochastic optimized submovement model and the minimum jerk principle.

As mentioned in Chapter 2, the stochastic optimized-submovement model of Meyer et al. predicts that targeted motion occurs in two stages [46]. A large initial impulse is aimed at the centre of the motion's target. This initial impulse, lasting time T_1, consists of primarily ballistic motion that brings a subject close to the final target. As the subject nears the final target, feedback mechanisms in the neurophysiological system correct the movement, if necessary, with secondary movements lasting time T_2 [46]. Goal directed movement is a stochastic optimization problem, where the increased error rate of higher initial motion amplitudes (with higher probability of secondary impulses) trades off against the shorter time to traverse the distance to the final target. The model corresponds well with Fitts' Law and experimental data, converging on a logarithmic Index of Difficulty term as the number of secondary impulses increases. The optimal motion for a goal-directed movement, as specified by this model and any of the two-component models discussed in Chapter 2, is for a user to hit the target using the initial ballistic motion.

Recall that the kinematics of unconstrained, ballistic motion obey the minimum jerk principle [66] and can be defined by the equation:

$$x(t) = x_0 + (x_0 - x_f)(15\tau^4 - 6\tau^5 - 10\tau^3) \tag{3.1}$$

where $\tau = t/t_{final}$, the normalized period of the gesture. x_0 is the initial hand position coordinate at $t = 0$, and x_{final} is the final hand position coordinates at $t = t_{final}$. By

normalizing the distance traveled by assuming the starting position coordinate is 0 and the total distance traveled as 1, we can rewrite Equation 3.1 as:

$$x(t) = 10t^3 - 15t^4 + 6t^5 \tag{3.2}$$

to produce an equation for distance and time t. We produce an equation for speed by taking the derivative of Equation 3.2 with respect to t, yielding, after some simple algebra:

$$v(t) = 30t^2 \left(t - 1\right)^2 \tag{3.3}$$

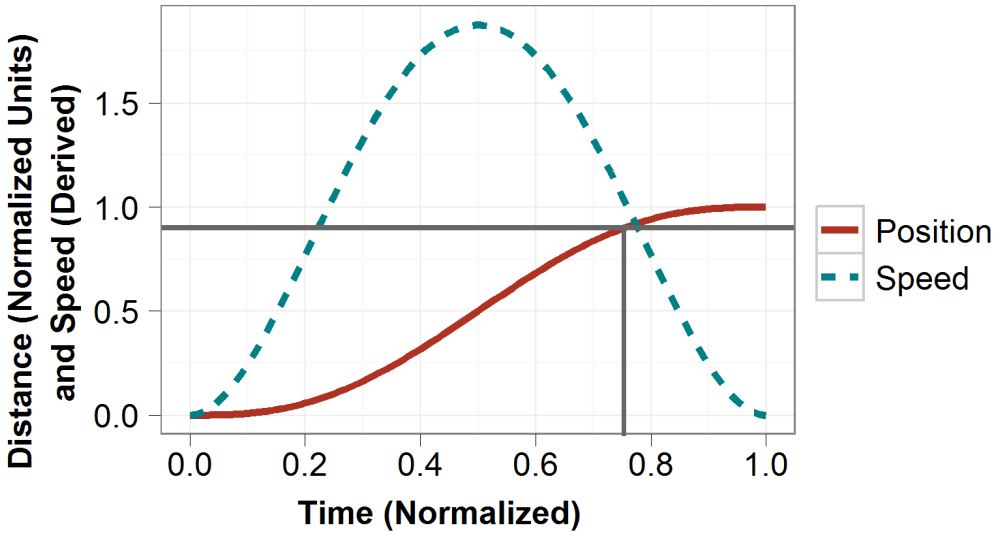


Figure 3.1: Theoretical distance and speed versus time profiles predicted by the Minimum Jerk principle.

The equations for distance and speed are plotted in Figure 3.1. In Figure 3.1, we see that the final 10% of a ballistic (unaimed) gesture's displacement consumes 25% of the total time. Based on the velocity profiles in work of MacKenzie et al. [36] and Graham and MacKenzie [22], the final 10% of aimed gesture displacement may consume as much as 50% of gesture time. It is also informative to look at graphs of distance versus time for experimental participants in our studies (shown in Figure 3.2).

In these graphs, the vertical axis is the distance traveled, normalized from 0-1 as a fraction of movement completed. The horizontal axis is normalized time on a 0-1 scale as

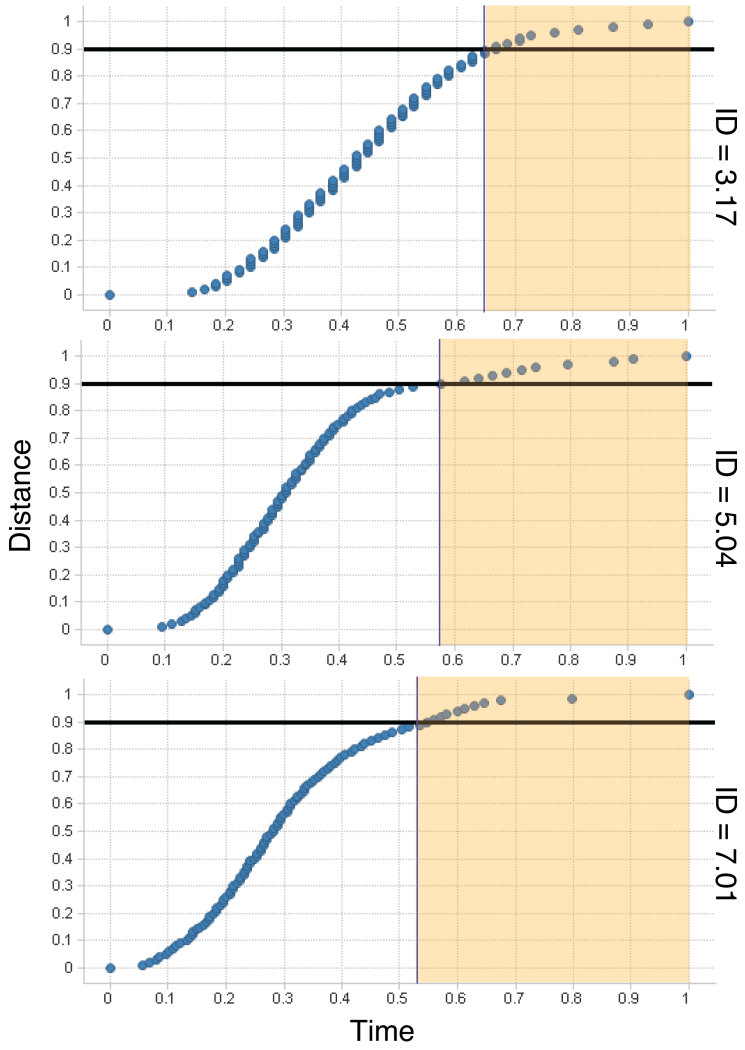


Figure 3.2: Examples of distance vs. time (both normalized) plots from the study for three IDs. The dark horizontal line represents 90% of gesture distance. The shaded region represents the time taken to complete the last 10% of distance.

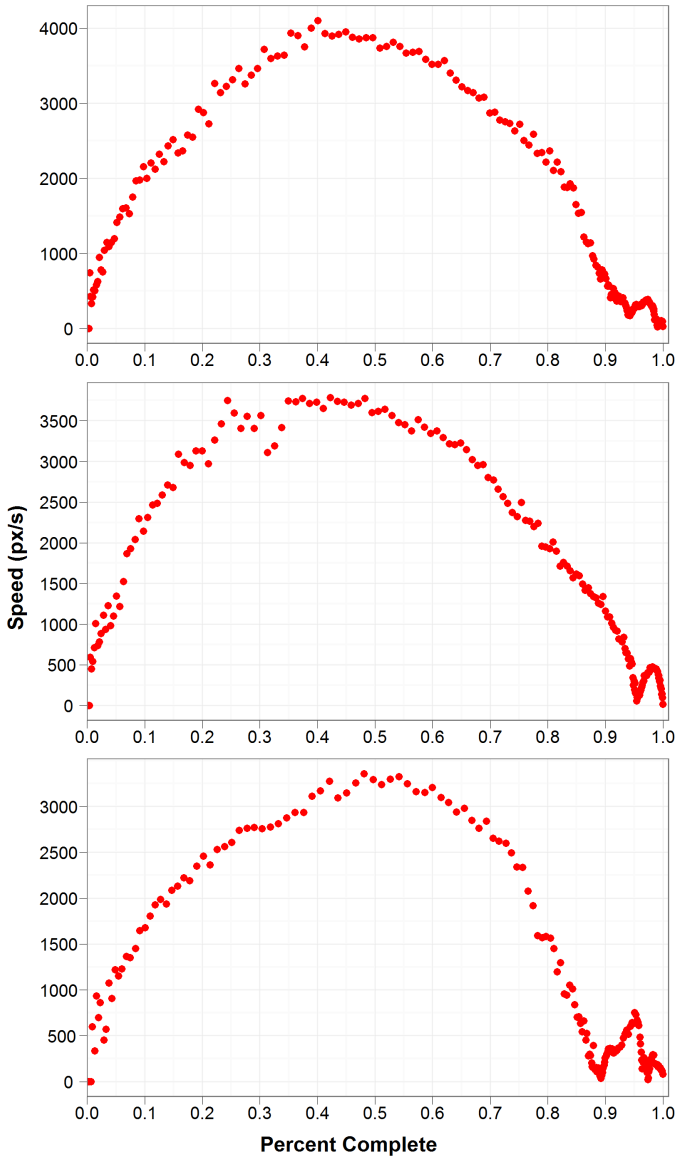


Figure 3.3: Examples of speed versus distance profiles during our study illustrating that corrective submovements tend to occur in the last 10% of movement distance (shown by the increase of speed at the end of the movement).

a fraction of time. Finally, each graph represents different IDs, from "easy" pointing tasks at the top (low-ID) to "difficult" pointing tasks at the bottom (high-ID). There are two salient features to observe in these graphs. First, the last 10% of movement, highlighted by the black horizontal line, consumes between 35% and 47% of movement time (i.e. the shaded region of time). As a result, while 90% of motion might seem very late in the gesture to predict endpoint, it is important to realize that a significant amount of time is consumed during the last 10% of motion in Fitts-style pointing tasks.

The other salient characteristic is the motion profile itself. As shown in the speed versus distance plots illustrated in Figure 3.3, it is apparent that at 90% of motion, the profiles deviate from smooth, ballistic trajectories. This is a known observation, but it merits highlighting. The initial 90% of motion is dominated by ballistic movement, and the last 10% of movement is dominated by corrective submovements. Therefore, the goal of any predictive algorithm should be to identify a target before 90% of motion. This gives sufficient time, about 40% of movement time, for the user to adjust to changes in the display. It also ensures that changes in the target or display do not occur during a period of time when aiming effects are dominant, i.e., when the user is trying to acquire the target.

In conjunction with previous work and findings from data collected for this thesis, we have concluded that the last 10% of motion consumes anywhere from 25% to 50% of total movement time and the first 90% of motion length is exclusively ballistic in nature. We now focus on our second preliminary question into the effect task constraints have on motion kinematics.

3.3 Analyzing the Kinematics of Amplitude and Directional Constraints on Pointing

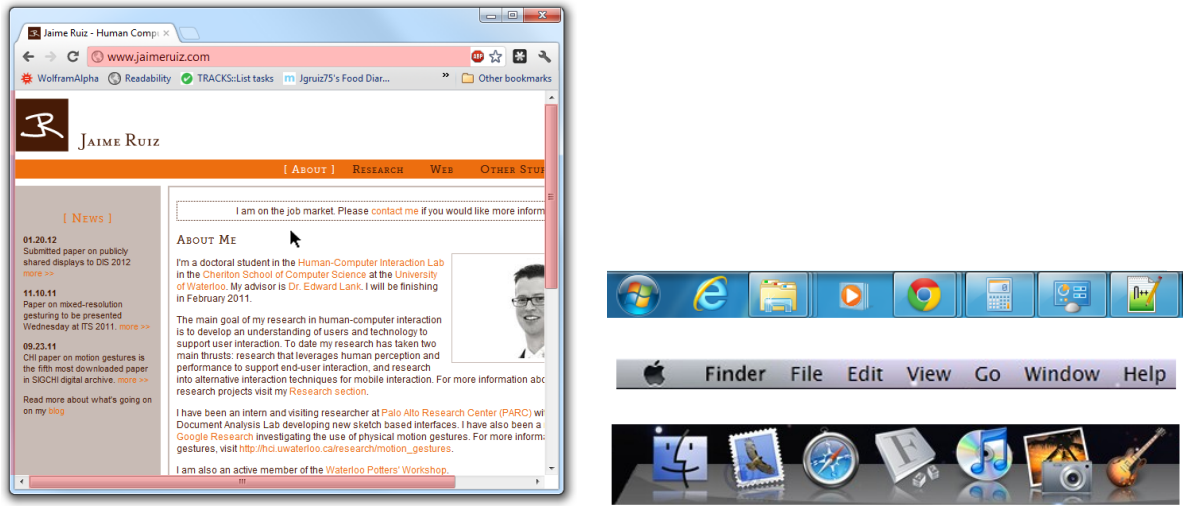
Any button widget in a typical graphical user interface places either an amplitude or directional constraint on a pointing task, depending on its position relative to the cursor. As well, amplitude constraints in interfaces include tasks such as targeting a scrollbar or targeting the edge of a window for resizing, and directional constraints include tasks such as targeting menus in computers running the MacOS operating system. Recall from Chapter 2.1.2, previous work has shown that the two constraints have an unequal impact on movement times for pointing tasks [2, 1]. However, there is not yet an understanding of the impact of amplitude versus directional constraints on the underlying motion characteristics.

In this section, examine the effects an amplitude or directional constraint has on the

kinematic characteristics of motion in pointing tasks. In particular, we are interested in answering several open research questions on the kinematics of bivariate pointing tasks, including:

- Which parameters of movement are affected by amplitude versus directional constraints?
- When is movement first affected by amplitude and directional constraints?
- Given a constrained pointing task, can we determine which constraint generated a motion profile using parameters of the motion?
- How similar is the effect of amplitude versus directional constraints across users?

We found that pointing data is complex, and that the two types of constraint result in significant overlap within the parameter space. To address the overlapping nature of the



(a) Examples of amplitude contraints (highlighted in red).

(b) Examples of directional contraints.

Figure 3.4: Examples of pointing constraints in interfaces. (a)Examples of amplitude constraints (highlighted in red) relative to the cursor occurring in a the Chrome web browser. (b) Toolbars and menus such as the Windows 7 taskbar (top), Mac OS X menubar (middle), and Mac OS X taskbar (bottom) are examples of directional constraints because they are positioned at the edge of the display, thus, allowing the cursor to be stopped by the edge of the display resulting in an infinitely tall target.

data, we applied machine learning techniques, specifically Hidden Markov Models (HMMs), to determine those parameters most affected by amplitude and directional constraints. (For background information on HMMs for Kinematic Analysis see Appendix A.) We show that instantaneous speed, acceleration, and jerk are affected by the type of constraint. We also observed that orthogonal components (i.e, perpendicular to primary direction of movement) tend to be stronger indicators of target constraint condition than directional components. We show that the differing effects on the kinematics of amplitude and directional constraints are observed during the first 70% of the gesture's path. Finally, we will demonstrate that while changes in the kinematics can be observed in the orthogonal components, these changes tend to be masked when examined in conjunction with the collinear profile when examining speed over distance profiles.

We being by describing an experiment designed to gather data on pointing tasks representative of those found GUIs. We then present the results of applying HMMs to understand the impacts of amplitude verus directional constraints on motion and discuss their implications.

3.3.1 Experiment

To determine the kinematic effects of amplitude and directional constraints on pointing motion, we designed a data collection task that captured bivariate pointing data. Below we describe the participants, display task, experiment design, apparatus, and the procedure by which data was collected.

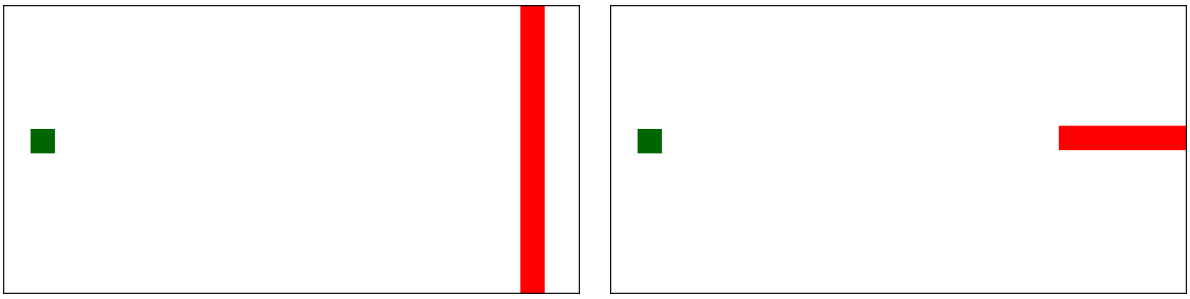
Participants

Eight people, two female and six male, all right-handed, participated in the experiment. All participants were university students.

Task

The task (displayed in Fig. 3.5) was a discrete, one-dimensional pointing task. Initially a green starting rectangle was displayed on the screen (shown on the left in Fig. 3.5). The task began when the participant used the cursor to click within the starting location. At that time, a red target would appear on the opposite side of the display (shown on the right in Fig. 3.5). Participants were required to move the cursor to the red target and use the mouse button to click on the target. A successful target acquisition (i.e., clicking

within the target region) was indicated with the target changing color. Users were told to acquire the target as quickly and accurately as possible, similar to other Fitts' Law tasks.



(a) Amplitude constraint task (b) Directional constraint task

Figure 3.5: The experimental tasks used in our study analyzing the kinematics of amplitude versus directional constraints on pointing motion. The starting target is shown in green and the goal target in red. The black border represents the display boundary.

Design

The experiment consisted of a within-subjects design with repeated measures. The independent variables were target ID and the bivariate constraint of the target (amplitude or directional). The IDs ranged between 3.17 and 7.01, and were the result of the 15 distance/width and 15 distance/height combinations (in pixels): 512/4, 512/8, 512/16, 512/32, 512/64, 1024/8, 1024/16, 1024/32, 1024/64, 1024/128, 1536/12, 1536/24, 1536/48, 1536/96, and 1536/192.

Apparatus

The experiment was conducted on a generic desktop computer (P4, 2.0GHz) with a 23-inch 1920x1200 LCD display running custom software written in C#. Input was collected using a Wacom Intuos3 five button mouse on a 12x19 inch tablet set to a 1:1 control display ratio. The 1:1 control display ratio ensured that motor space and visual space coincided throughout the pointing task. The tablet was used because of its high sampling rate.

Procedure

The experiment consisted of eight blocks: one practice block and seven experimental blocks. Each block consisted of 15 D/W combinations presented twice for each constraint, resulting in 60 tasks per block. The order of presentation of the D/W combinations and constraints was randomized. To minimize fatigue, participants were required to take a five minute break between blocks. The experiment took approximately 60 minutes to complete.

3.3.2 Measures

The custom software captured mouse movements at 200Hz. Movement time, X position, and Y position were captured for each registered mouse movement. Movement time was calculated from when the user clicked the start target to when the user acquired the intended target. To examine the kinematics of motion, we used the position and time information to calculate velocity, acceleration, jerk and curvature for each data point in both the X and Y directions.

3.3.3 Preliminary Data Analysis

Below, we present some initial analysis of the data. The purpose of this initial analysis is to verify that our data agrees with observations of bivariate pointing times by Accot and Zhai [2], and to demonstrate the overlap of the two constraint conditions within individual parameters of motion.

Figure 3.6 plots movement time against Index of Difficulty (ID) for both amplitude and directional constraint. Similar to results reported by Accot and Zhai [2], we see that amplitude constraint (stopping) typically takes longer than directional constraint. Analysis using repeated-measures ANOVA shows a significant effect for constraint ($F_{1,7} = 23.30, p = .002$) and for ID ($F_{4,28} = 44.230, p < .001$). To examine the interaction between constraint and ID, we used univariate ANOVA.[1] The results indicate that there is a significant effect for target constraint*ID ($F_{4,2891} = 3.13, p < .01$).

Our initial hope was that apparent differences would exist in motion profiles for the different constraint conditions. However, our examination of the kinematic characteristics of motion did not clearly display differences by the two types of constraints. Consider, for example, Figures 3.7(a) and 3.7(b), which plot velocity in both the Y and X directions

[1] With RM-ANOVA, only one within-subjects factor can be included in the analysis at a time.

for the two constraints as a function of the percentage of stroke completion. These figures illustrate the degree to which the characteristics of motion overlap regardless of the type of constraint. Plots of other motions characteristics were similar in their lack of observable differences. Likewise, plots with motion characteristics separated by ID showed obvious differences according to ID, however, differences owing to constraint were again difficult to assess based on visual inspection alone.

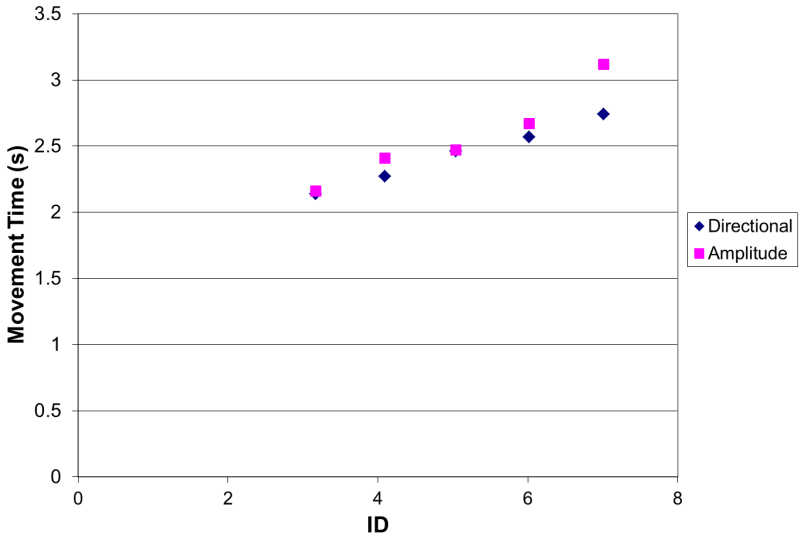
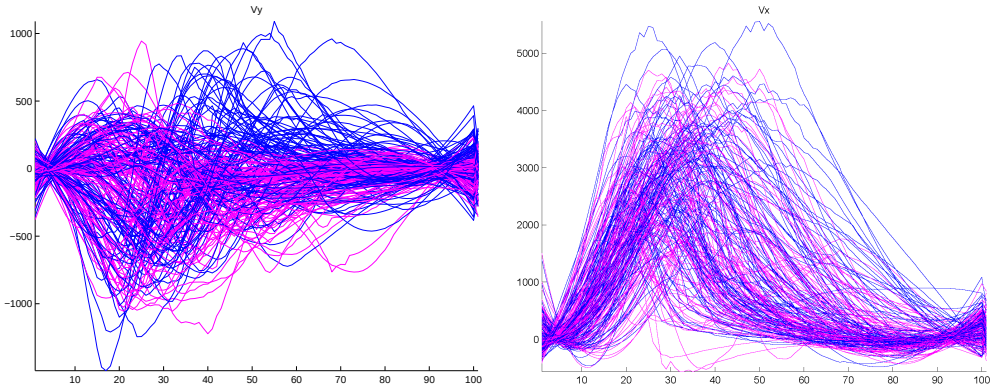


Figure 3.6: Movement times by target constraint.

Despite the lack of readily observable divergences between profiles, the significant effect that constraint has on movement time requires that some variations in motion must be present. To extract from the data those parameters of motion that result in the differences in time for amplitude and directional constraint, we trained HMMs on various combinations of features and used changes in recognition accuracy to determine those parameters affected by changes in target constraint.

3.3.4 Results

In this section we present the results of using HMMs to recognize and characterize the effect of amplitude and directional constraints on the motion of pointing tasks.



(a) Velocity in the Y direction (b) Velocity in the X direction

Figure 3.7: Velocity in the X and Y directions for the target constraints by percentage of stroke completion. Magenta represents amplitude constraints and blue represents directional constraints. (Requires colour viewing)

HMM Training

We trained the HMMs using two strategies.[2] The first strategy resulted in what we refer to as *user-specific HMMs*. This strategy, which provided the classifier with access to data from all users during training, employed an 8-fold cross-validation technique. For each fold, the HMM was trained using 90% of the data from all users and then tested on the remaining 10%. This process was then repeated nine additional times, each time using different training/test sets. The second training strategy resulted in *generic HMMs*. With this strategy, data was withheld from one user during training to test the classifier's ability to generalize to new users. In particular, we used 8-fold cross validation, where, for each fold, the HMM was trained on data from seven users and tested on the remaining user.

For user-specific HMMs to be useful for on-line prediction, the system would have to observe the user perform a number of pointing gestures (and have knowledge on target dimensions) prior to reaching the levels of accuracy described in here. The generic HMMs are useful in understanding potential predictive accuracy when the classifier has no knowledge of a given user.

All HMMs had five states, with one mixture per state. The HMMs were also fully connected, meaning that transitions were possible between all states. Before settling on five states, we experimented with other configurations. We found that for zero to five

[2]See appendix A for more information on the construction of our HMM models.

states, accuracy increased as the number of states increased. After five states, there was there was a very slight increase in accuracy. For the purpose of our experiments, however, this increase was not enough to warrant the additional training time.

HMM Inputs

The inputs to the HMMs are all derived from the position and timing information of the pointing motion. We identify three categories of parameters: instantaneous parameters, path-based parameters, and cumulative parameters. The inputs for each these three parameter categories are summarized in table 3.1.

Input	Description
	Instantaneous Components (Vectors)
V_x	Velocity w.r.t. X-axis
V_y	Velocity w.r.t. Y-axis
A_x	Acceleration w.r.t. X-axis
A_y	Acceleration w.r.t. Y-axis
J_x	Jerk w.r.t. X-axis
J_y	Jerk w.r.t. Y-axis
	Path-Based Components
κ	Local curvature of trajectory
	Cumulative Components (Scalars)
V_{c_x}	Cumulative speed w.r.t X-axis
V_{c_y}	Cumulative speed w.r.t Y-axis
A_{c_x}	Cumulative acceleration w.r.t X-axis
A_{c_y}	Cumulative acceleration w.r.t. Y-axis
J_{c_x}	Cumulative jerk w.r.t. X-axis
J_{c_y}	Cumulative jerk w.r.t. Y-axis

Table 3.1: The inputs combinations used to train the HMMs.

Our rationale for these three categories of parameters is an observation that target constraint can affect motion in three ways. First, the act of either steering (directional constraint) or stopping (amplitude constraint) can cause changes in instantaneous components of movement during motion, a result of trying to control one or more instantaneous parameters of movement. As an example, the y-component of jerk might vary more abruptly during motion to keep the trajectory aligned with the target.

Second, a directional constraint could cause the path to bend more than a stopping constraint, a result of trying to steer back to a target. We used local curvature to measure variations in the "straightness" of the motion path. Curvature was calculated using the standard curvature formula:

$$\kappa = \frac{|v_x a_y - v_y a_x|}{(v_x^2 + v_y^2)^{3/2}} \tag{3.4}$$

Finally, different constraints could result in variations in the overall components of motion. For example, a directional constraint might result in a higher peak speed than an amplitude constraint for a given distance. Cumulative parameters encapsulate information on the entire trajectory. We computed cumulative parameters as the sum of the absolute values of velocity, acceleration and jerk components, allowing us to analyze whether changes in overall motion occurred as a result of amplitude and directional constraints.

Analysis

HMMs were trained using different combinations of the inputs in Table 3.1 to determine which inputs contained the most differentiating information. We considered eighteen different combinations of features covering both parameters within the three categories and mixtures of parameters from across the different categories. We selected the eighteen feature sets based on what we felt would be the most informative and also provide a good breadth of coverage of the parameter space. While it would be possible to experiment with additional feature combinations, we did not feel that they would provide the same insight, and both the time necessary to train the HMMs (using the 10-fold cross validation) and the tractability of the analysis were limiting factors.

Our first goal was to determine where, along motion path, observable differences caused by target constraint become apparent. Figure 3.8 depicts a typical graph of recognition accuracy given partial results. This figure is representative of the results for both the user-specific and the generic HMMs. After having observed only 10 % of the task, as expected, the recognition accuracy is about 52%, or near chance. However, once 70% of the task is observed, accuracy rises to about 73% and remains stable for the remainder of the sequence. This implies that once 70% of the task is observed we can make as accurate a guess as if the task was completed. Consequently, in the remainder of our analysis we consider only the first 70% of the observations. During the final 30% of the motion, no new constraint-specific information can be gleaned from the motion.

Of the three categories of features, we found that instantaneous components of motion were most affected by variations in target constraint. In contrast, target constraint had

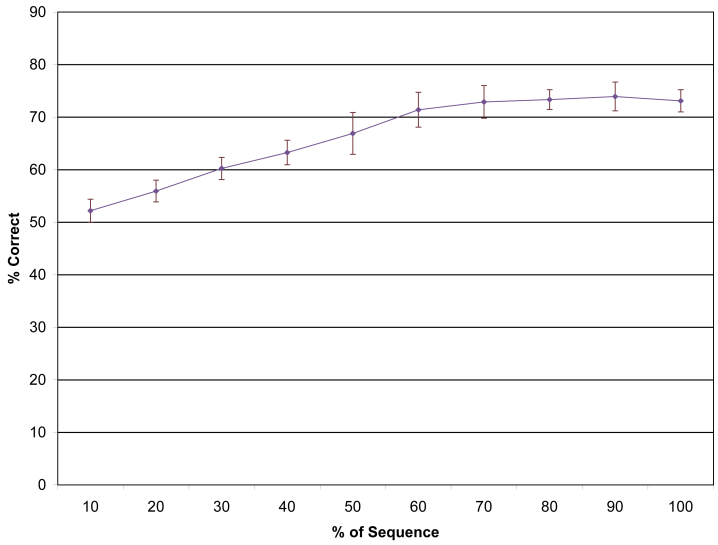


Figure 3.8: Accuracy given partial observations for a user-specific HMM using instantaneous x- and y-components of motion.

much less of an impact on the overall components of movement and the path-based parameters. As instantaneous parameters were most affected by bivariate constraint, we present six different combinations of these features. Table 3.2 indicates the specific cases that will be presented for analysis in the following section. For comparison purposes, we also present representative results from the other two categories (Case 7 and Case 8) and an additional parameter combination that incorporates combinations of instantaneous, path-based, and cumulative features (Case 9). In the eighteen cases that we examined, a number of cases combined features from the different categories, however, such combinations resulted in no improvement in recognition. In Case 9, we use all available parameters from all categories.

For the combinations listed in Table 3.2, we now explore classifier accuracy. As noted in Section 3.3.4, we used two different strategies for training HMMs. The first was to create user-specific HMMs, which are trained on data from a specific user. The second was to create generic HMMs, which analyze information given no prior data on a specific user. Figure 3.9 presents the results for user-specific HMMs, and Figure 3.10 presents recognition accuracy for generic HMMs. Errors bars in the graphs denote the standard deviations from the cross-validation. As noted in section 3.3.4, user-specific HMMs (Figure 3.9) were evaluated using 10-fold cross-validation, and generic HMMs (Figure 3.10) were evaluated using a leave-one-out strategy.

Case	Inputs
	Instantaneous Parameters
1	$V_x, V_y, A_x, A_y, J_x, J_y$
2	V_x, V_y
3	V_x, A_x, J_x
4	V_y, A_y, J_y
5	V_x
6	V_y
	Path-Based Parameters
7	κ
	Cumulative Parameters
8	V_{c_x}, V_{c_y}
	Multi-Category Parameters
9	$V_x, V_y, A_x, A_y, J_x, J_y, \kappa, V_{c_x}, V_{c_y}, A_{c_x}, A_{c_y}, J_{c_x}, J_{c_y}$

Table 3.2: Description of cases

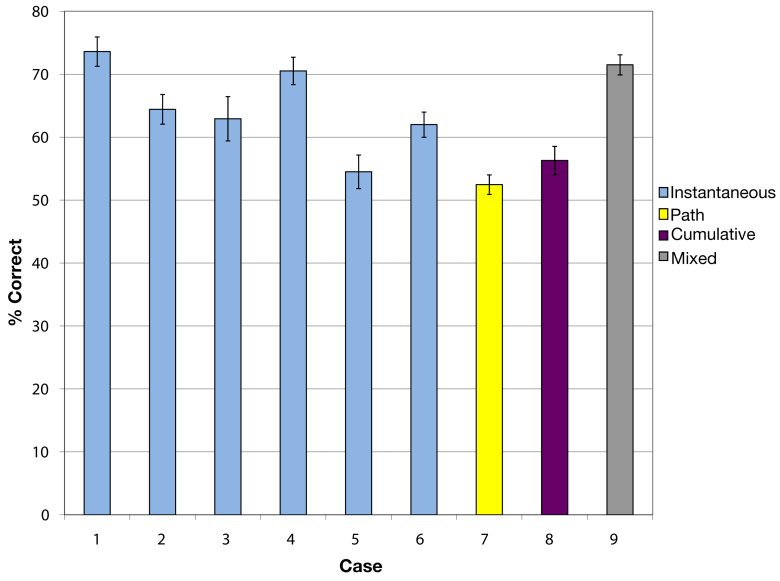


Figure 3.9: User-Specific HMM Results

In our results, we use HMMs to distinguish those features of motion that are most affected by target constraint. We do this by providing selected features from an unlabeled

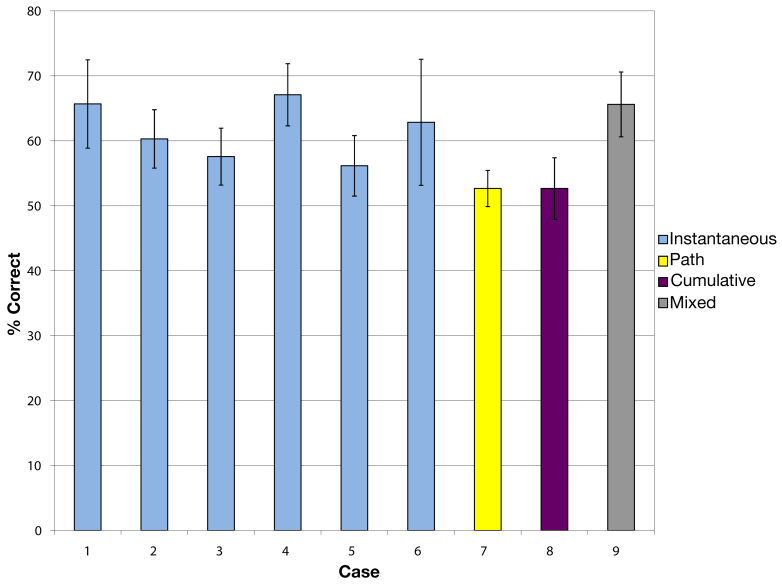


Figure 3.10: Generic HMM Results

instance of a constrained pointing task to a trained HMM and asking it to classify whether the sample was generated by an amplitude or directional constraint. As we add additional features, we expect to see the HMM's accuracy increase if the features it uses are affected by target constraint. Accuracy should remain constant (or decrease slightly[3]) if the features are not affected.

Our first result is that the best possible HMM we observed was a user-specific HMM that had access to all instantaneous kinematic components of motion. Under this condition, the HMM can classify target constraint with 73.6% accuracy. This is represented as Case 1 in Figure 3.9. These results indicate that there are detectable differences between the pointing motions with amplitude constraints and with directional constraints. Furthermore, it is possible to detect this difference on an individual observation (given training data) in real-time with much better-than-chance accuracy.

Figure 3.10 illustrates the results of training the HMMs under the second scenario, where each HMM is trained on seven users' data, then tested on the remaining user. This represents the more difficult recognition task of creating a user-independent model. As

[3]Using spurious features frequently causes Machine Learning (ML) techniques to perform slightly worse unless a sufficiently large training corpus is available. With enough training, ML techniques can be trained to ignore spurious features. However, for reasonable training sets, the observation of poorer performance with additional features is sufficient to conclude that the new features contain no useful information.

expected the recognition rates decreased, with the best case averaging 67.1%, and the standard deviations tending to be larger. Although these recognition rates are lower, they are again much better than chance. The generic results also correlate well with the results from the user-specific HMMs. Furthermore, this result indicates that the difference between amplitude-constrained pointing motions and directionally constrained pointing motions are consistent across multiple users.

As mentioned above, Case 1 represents the best-case scenario in our results. To isolate which inputs provide the most useful information, we experimented with several different combinations of the features in Case 1, shown in Figures 3.9 and 3.10 as Cases 2 - 6. As indicated in Table 3.2, Case 2 is x and y components of speed, Case 3 is instantaneous x-components, Case 4 is instantaneous y components, Case 5 is the x-component of velocity and Case 6 is the y-component of velocity. The single most important input was velocity with respect to the y-axis (Case 6). Adding acceleration and jerk with respect to the y-axis (Case 4) produced nearly as accurate recognition, 70.5%, as Case 1, indicating that most discriminating information is contained in motion perpendicular to the direction of the target.

Other categories of features were less affected by target constraint. Little discriminating information exists in curvature (52% accuracy, only slightly above chance, shown in Case 7). Cumulative components do perform better than chance (Case 8). However, cumulative components include no new information, as shown in Case 9, where when all components are added to the HMM we observe a slight drop in recognition. If any of the additional components added useful information, we would expect to see an increase in performance for the mixed HMM.

In summary, we have shown that the difference between amplitude- and directionally constrained pointing motions is detectable. We have shown the most discriminating information is encoded in the motion perpendicular to the direction of the target. We have presented evidence that the differences between pointing motions are detectable once only 70% of the motion has been observed, and that these observations are user independent.

3.3.5 Discussion

At the begining of this section we introduced four questions on the kinematic properties of amplitude versus directional constraints on pointing:

- Which parameters of movement are affected by amplitude versus directional constraints?

- When is movement first affected by amplitude and directional constraints?
- Given a constrained pointing task, can we determine which constraint generated a motion profile using parameters of the motion?
- How similar is the effect of amplitude versus directional constraints across users?

Our results provide answers to each of these questions.

First, we analyzed various parameters of movement at different positions along the trajectory. We note that after 10% of the trajectory, amplitude and directional constraints have little effect on motion. However, by 20% of movement, we are able to predict constraint with an accuracy that exceeds chance, and this accuracy increases until 70% of trajectory is observed. This indicates that corrective components of motion can occur early in motion, and that they become more pronounced through to 70% of movement. During the last 30% of the trajectory, corrective movement is undoubtedly occurring, but there exists little qualitative difference between instantaneous, path-based, or cumulative characteristics of movement.

Second, based on Figures 3.9 and 3.10 we can conclude that most effects of target constraint occur in the instantaneous y-component of movement. Furthermore, we found that path-based and cumulative components of motion are not affected by target constraint.

Third, we note that we can determine the target constraint from parameters of motion 73.6% of the time with user-specific Gaussian HMMs – a result that is much better than chance. Since our focus in this thesis is on understanding the underlying kinematics, we have not yet explored ways to improve classifier recognition. Therefore, it is likely that with further attention paid to improving recognition that this accuracy rate will rise. Whether or not such a classifier could achieve an accuracy rate that acceptable from the user's perspective, given an interaction technique designed to leverage these predictions, remains an open question.

Finally, based on the accuracy of the user-specific and generic HMMs, the y-component effects of target constraint generalize across our subjects. However, Case 1 in Figure 3.9 shows that a small improvement in recognition occurs when x-components of motion are considered in user-specific HMMs. This improvement is not apparent in Figure 3.10, indicating that the variability in the x-component does not generalize across users.

Effects of Bivariate Constraints on Ballistic Motion

In this section, we have shown that the difference between amplitude and directionally constrained pointing motions is detectable and that the most discriminating information is

encoded in the motion perpendicular to the direction of the target. Given these results we focus our attention on the overall effects of bivariate constraints on the initial ballistic motion. As demonstrated by our results, the differences in orthogonal characteristics appears to be mainly concentrated in the initial 70% of gesture motion. However, as we see in Figure 3.7 the magnitude of orthogonal velocity is an order of magnitudes lower than that of collinear motion. Therefore, the differences in orthogonal characteristics are masked due to the dominating collinear movement. Specifically, we examine the speed versus distance kinematic profiles by target constraint, we observe no difference between motion profiles while pointing to a collinear or orthogonal constrained target. Therefore, we conclude that while differences in orthogonal constraints can be detected they are mitigated by the magnitiude of the collinear characteristics.

However, there are significant benefits to the observation that we can classify pointing motion based on whether the endpoint of the movement is constrained either orthogonal to or collinear with the direction of movement. For example, imagine that the goal of an endpoint predictor is to identify a specific target on the display. If this is the case, then the endpoint predictor and the constraint on the endpoint could be used as two independent data to increase the accuracy of a target prediction.

3.4 Effects of Intended Use on Motion Kinematics

As mentioned in Chapter 2, Mandryk and Lough [41] examined whether the intended use of a target in graphical interfaces also affects the movement time and kinematic profiles of end-users who seek to acquire that target. They examined four common intended uses of on-screen objects (illustrated in Figure 3.11). These were: single targeting, where the user presses and releases the mouse button over a target; dual targeting, where the user first clicks on one target and then moves to and clicks on a second target; flicking, where the user presses the mouse button down on a target and quickly and imprecisely directs the target to another position on the screen; and docking, where the user presses the mouse button down on a target and carefully repositions it within a tightly constrained region on the display.

Mandryk and Lough analyzed the initial targeting task, i.e. the task of moving from a start position to the target location before performing the intended use task. We call this first target the user's primary target. Mandryk and Lough demonstrated that the movement time of Fitts-style pointing tasks varies based on intended use of the primary target, i.e. that the time taken to depress a mouse button over the primary target varies depending on the subsequent task to be performed. They also expressed concern that their

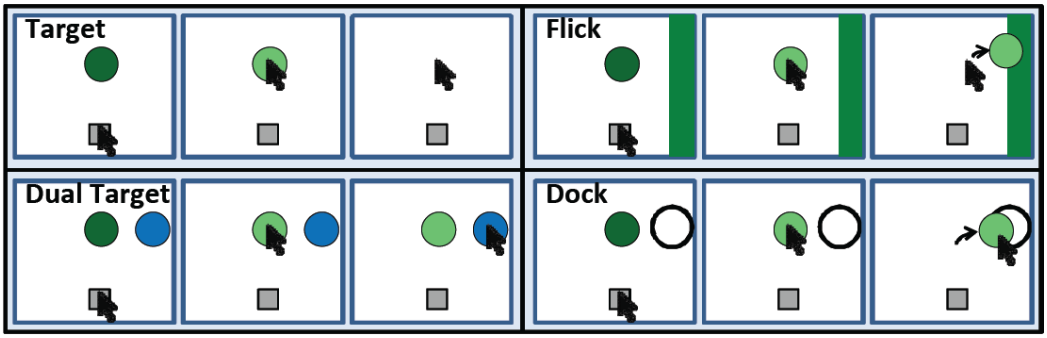


Figure 3.11: The four tasks tested by Mandryk and Lough. Frame 1 illustrates selecting the start square. Frame 2 demonstrates the initial targeting movement. Frame 3 illustrates the action to be taken by the user with the acquired target. Adapted from [41].

results may have a significant impact on researchers who seek to model movement time based on kinematic profiles.

While differences may exist in time, Mandryk and Lough provide little data on how (or even whether) Fitts' Law models movement time for variations in intended use. Assuming that the linear relationship between movement time and ID is preserved for different intended uses, whether we observe changes in ID, IP, or A in the Fitts' Law equation (Equation 2.3) is an open question.

Finally, the deceleration phase of movement encompasses all user action from peak speed until they depress the mouse over the target. Mandryk and Lough provide little guidance on where, during deceleration, the changes in movement time occur. Is the entire deceleration curve affected? Or does the user simply spend a bit more time over the target after movement stops before pressing the mouse button to begin his or her secondary task?

As a result of these open questions, we now present a replication study that analyzes the kinematic and temporal characteristics of the intended uses originally analyzed by Mandryk and Lough.

3.4.1 Method

Apparatus

The experiment was conducted on a generic desktop computer (Core 2 Duo, 3.0GHz) with a 17-inch 1280x1024 LCD display (mimicking Mandryk and Lough) running custom software

written in C#. Input was collected using a Microsoft Sidewinder X3 mouse with cursor acceleration set to the Windows operating system default level.

Tasks

The tasks conditions were the same as described by Mandryk and Lough and required participants to perform an initial aiming movement (primary task) in addition to a secondary subtask. Our primary task differs from Mandryk and Lough in that we opted for a modified version of the ISO 9421-9 [28] targeting task (shown in Figure 3.12(a)) to vary the direction of movement. Eight circular targets were arranged in a circle with a radius of D. Our ISO 9421-9 targets differ from the standard ISO setup in that we only displayed the starting target (represented by the color blue) and the final target (represented in red). After completing the full task (primary and secondary), the task would continue with the previous primary target becoming the new start target. This sequence would continue until all eight targets were traversed (resulting in 9-targeting tasks per arrangement).

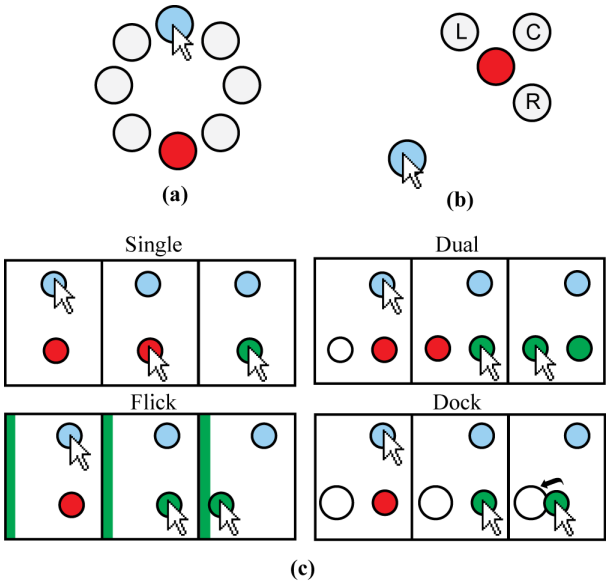


Figure 3.12: (a) The standard ISO 9421-9 targeting task. (b) Our modified task and the possible secondary target locations. (c) Task conditions for the study. Frame 1 - Start location; Frame 2 - the primary task; Frame 3 - the sub-task to be performed.

At the onset of the trial, the primary task target was colored gray. The task began when the participant moved the cursor into the blue colored starting area and hovered for

approximately one second. At that time, a red target would appear on the display. As in previous Fitts' tasks, participants were required to move the cursor the red target as quickly and accurately as possible.

The secondary tasks were replicated from Mandryk and Lough and included:

- Single Targeting Task: For the targeting task condition participants were not required to perform a secondary task. Therefore, only the primary task was performed. This task condition replicates the task normally performed in Fitts' style studies.
- Dual Targeting Task: In the dual targeting task, the user is presented with two targets, the primary target and the secondary target. Once the user completes the primary task by clicking on the primary target, the secondary target turns red. The user is then required to click on the secondary target.
- Flick Task: For the flick task, the participant is presented with the primary target and a 35-pixel green border on an edge of the screen. The participant was told to move to the primary target (primary task) and flick the target in the direction of the green-border. Mandryk and Lough created the flick task to echo the throwing task of Marteniuk et al.[42]
- Docking Task: In the docking task the user is presented with the primary target and a docking region, represented by a white disk 20 pixels larger than the primary target. The participant was required to move to the primary target and drag the target into the docking region. Mandryk and Lough created the docking task to echo the fitting task of Marteniuk et al.[42]

Location of secondary targets was randomized to one of three locations (collinear, left, or right) in relation to the direction of motion as shown in Figure 3.12(b).

3.4.2 Design and Procedure

The studies consisted of a 4 (task) by 3(target width) by 3 (target distance) within-subjects design with repeated measures. For the primary task, target widths (W) of 30, 60, and 120 pixels were each shown at a distance (D) of 150, 300 and 600 pixels. The resulting D/W combinations provided Indices of Difficulty (ID) between 1.17 and 4.39, echoing Mandryk and Lough's IDs. The study consisted of six blocks, two blocks for each task condition. The first block was a training block and was omitted during our analysis. Within each

block, participants were presented each D/W combination in random order resulting in 81 trials per task condition (324 trials per participant). Ordering of the task condition was counter-balanced using a 4x4 Latin Square.

Dependent Measures

As in Mandryk and Lough, all dependent measures focus on the initial task of acquiring the primary target and were calculated using the logs generated from our custom software. For each trial, we interpolated movement to create time-equidistant points along the gesture. Using the equidistant points we calculated speed and position at each point. Speed was smoothed using an interpolating degree 2 polynomial. This degree 2 polynomial naturally smooths the curve without the dampening effects on peak speed that occur when using a sliding window.

For consistency, our dependent measures were the same as Mandryk and Lough and included:

- Movement Time (MT): Movement time (MT) is defined by the temporal interval between the first detected mouse movement and the mouse down event on the primary target.
- Overshooting errors: Overshooting errors, referred to by Mandryk and Lough as exit errors, are errors in which the participant exited and re-entered the primary target prior to mouse-down.
- Peak speed (S_{max}): Peak speed is simply the maximum speed reached during the primary task.
- Time to peak speed (tS_{max}) and Percent after peak speed ($\%afterS_{max}$): Time to peak speed is the temporal measure taken to reach peak speed and represents the acceleration phase of the motion. Percent after peak speed is the amount of time that occurs after peak speed is reached as a percentage of total movement time and represents the deceleration phase of the motion.
- Click Speed: Click speed is defined as the mean speed over the 33ms prior to selecting the primary target.

Participants

Twenty participants (6 female) aged between 21-35 (mean 27.4) participated in the study. All participants were right-handed mouse users and affiliated with a local university. Participants were compensated $10 for participating. The study took approximately 60 minutes.

3.4.3 Results

Of the 6480 tasks recorded, 2.5% resulted in the user not correctly hitting the primary target and were removed form analysis. There was no statistical difference in error rate between task conditions.

Comparison to Mandryk and Lough

In this section, we briefly outline the similarities and differences between our results and those presented by Mandryk and Lough. In the next section, we provider a more detailed analysis of kinematic profiles to diagnose exactly why the observed differences occur.

We conducted a repeated measures MANOVA on MT, overshot errors, S_{max}, tS_{max}, $\%afterS_{max}$, and click speed with ID and task condition (i.e. intended use) as factors. Post-hoc analysis was also performed for dependent measures using Bonferroni correction.

Effects of ID Similar to Mandryke and Lough we observe significant effects of ID on all measures ($p < .001$ in all cases, $.48 \leq \eta^2 \geq .98$). Bonferroni corrected pairwise comparisons show:

1. Significant differences exist for MT and peak speed for all IDs ($p < .001$).
2. For time to peak speed (tS_{max}), significant differences exist between the lowest ID (1.17) and all other IDs ($p < .005$) and between the highest ID (4.39) and all other IDs ($p < 0.005$).
3. Percent after peak speed shows a significant difference between all IDs demonstrating that the deceleration phase of the movement increases as the index of difficulty increases.

4. Analysis of click velocity shows click velocity to be fastest for the lowest ID (1.17) compared to all other IDs and the second lowest ID (1.81) to be significantly faster than the highest ID (4.39) ($p < .01$ in all cases).

5. Significantly fewer overshot errors occurred for the two lowest IDs (1.17 and 1.81) than the higher IDs ($p < .001$ in all cases).

There were no significant ID*task condition interaction.

Effects of Task Condition Means and 95% confidence intervals for our dependent measures by task condtion are shown in Figure 3.13. The primary difference between our results and Mandryk and Lough involves the flick task. In particular:

1. Similar to Mandryk and Lough, analysis of variance shows a significant effect of task condition on movement time ($F_{3,342} = 8.08, p < .001, \eta^2 = .02$). However, pairwise comparisons show only the dock task to be significantly different than other task conditions ($p < .01$). Unlike Mandryk and Lough, we found no significant difference between movement times for flick and single target and dual target.

2. Similar to Mandryk and Lough, analysis of variance shows a significant effect of task condition on fraction of time after peak speed, $\%afterS_{max}$ ($F_{3,342} = 90.59, p < .005, \eta^2 = .04$). Post-hoc analysis using Bonforroni correction shows $\%afterS_{max}$ to be significantly higher for the dock task condition compared to the single targeting task ($p < .05$). Unlike Mandryk and Lough, we did not see a significant difference between flick and single target and dual target for $\%afterS_{max}$.

3. Unlike Mandryk and Lough, we see no significant effect of task condition on peak speed ($p > .07$). Mandryk and Lough found that flick had significantly higher peak speed than other task conditions. Qualitatively, we note that, in our experiments, the flick task condition actually had the lowest peak speed.

4. Similar to Mandryk and Lough, we see a significant effect of task condition on click speed ($F_{3,342} = 23.58, p < .001, \eta^2 = .40$). However, Mandryk and Lough found that click speed for flick and dock to be slower than for single target and dual target. In contrast, we found click speed for flick to be significantly faster than for all other conditions ($p < .001$ for all conditions). No other significant differences were found.

As with Mandryk and Lough, no significant effect of task condition was observed on overshooting errors or time to peak speed.

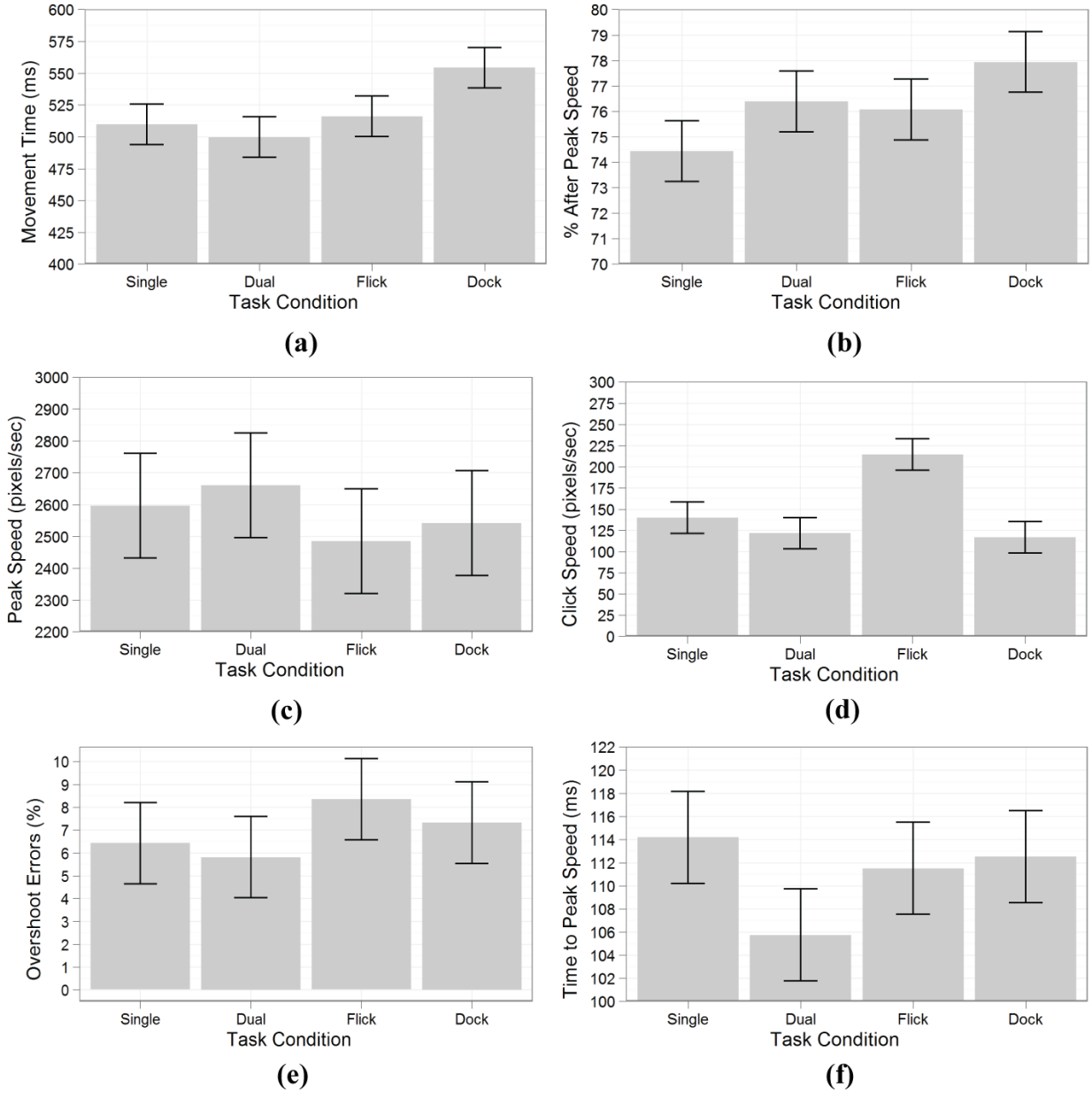


Figure 3.13: Dependent measures by task condition (error bars 95% CI). (a) Movement time. (b) Percent of gesture after peak speed. (c) Peak Speed. (d) Click speed. (e) Percentage of Overshoot Errors. (f) Time to peak speed.

Synthesis with Mandryk and Lough In general, our results support the observations of Mandryk and Lough that intended use has an effect on movement time. Specifically, if the secondary task is to dock the primary target, i.e. to drag the primary target to a restricted screen location, then movement time increases. As well, we also see that the increase in time is observed specifically in the deceleration phase of movement, i.e. the phase of movement after peak speed until the user clicks on the primary target. The one minor point of contrast is in our data for the flick task. However, Mandryk and Lough also note that the design of their flick task may have been more challenging for users than they anticipated. We attribute the differences between our results and Mandryk and Lough to the potential confound of a more difficult flick task.

Mandryk and Lough's primary concern with disparities in movement time based on intended use is that the effect may represent a significant alteration of the kinematic profile. The two potential areas of concern are the kinematics of the dock task as compared to other intended uses of a user's target during pointing motion because of the increased time taken during the deceleration phase. As well, the higher click speed of the Flick task that we observe may also cause problems if the overall kinematic profiles of motion are altered significantly.

Kinematic Analysis Based Upon Intended Use

It is important to note that many aspects of user movement may affect movement time and time spent during the deceleration phase of movement. Beyond variations in the kinematics of deceleration, variations may exist in the effective target width, and variations may exist in the amount of time a user spends motionless over a target before pressing the mouse button. The first question we must answer is where, exactly, during deceleration do variations in the kinematics of movement occur.

To answer this question, we first examined the average normalized speed versus distance profiles of end-user motion for each of the task conditions. When we examine these normalized profiles, shown in Figure 3.14, we see that all deviations in profiles are concentrated near the end of movement.

Because of the similarities in the normalized and averaged kinematic profiles of each task, we became interested in exactly why the discrepancies in movement time between dock and the other intended use tasks were observed. To address this specific question, we identified four possible hypotheses that could cause an increase in average movement time for the dock task. Our four hypotheses are:

H1 The Index of Performance (IP) could be lower for the dock task;

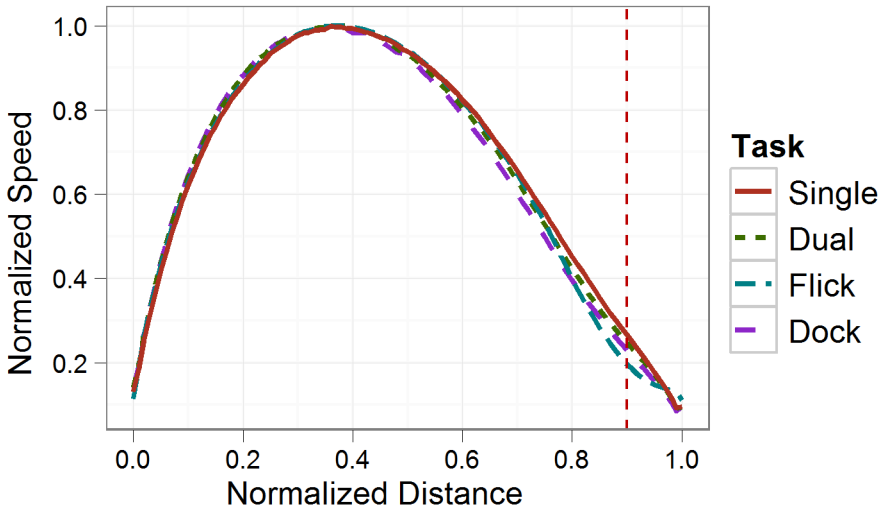


Figure 3.14: Normalized kinematic profiles by task condition.

H2 The nature of the dock task could cause a change in the effective target width for that task;

H3 Despite the lack of effect on the endpoint prediction algorithm, it is still possible that the deceleration phase from peak speed until the cursor stops over the target could be longer for the dock task.

H4 The user may spend more time hovering after movement is complete over the primary target before pressing the mouse button to begin the dock task.

Any or all of these factors could be affected by intended use. We address each of these possible explanations in turn.

First, consider a possible change in the slope of the movement time versus ID graph, i.e. a change in the Index of Performance of the targeting task. To explore whether this is, indeed the case, we plotted movement time versus index of difficulty for each task (Figure 3.15).

For all task we observe a high correlation ($R^2 > .99$) between movement times and ID as described by Fitts' Law, so Fitts' Law applies to each of the targeting tasks, regardless of intended use. We can also clearly observe the consistently higher movement times of the dock task (diamond line). However, the slope of dock's movement time vs ID line is

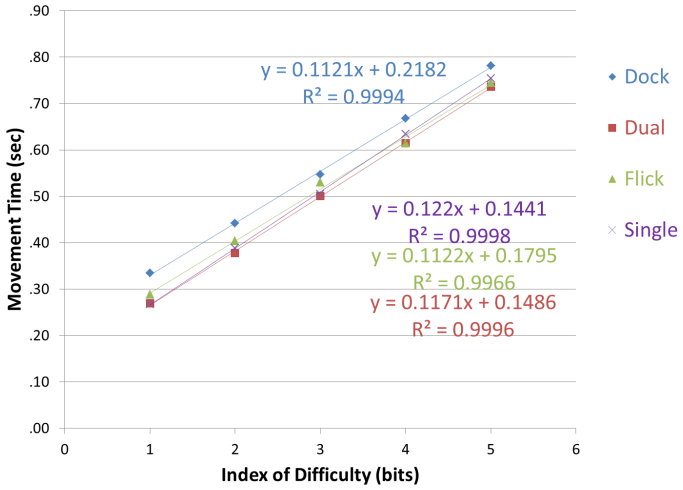


Figure 3.15: Movement time by ID for each task condition.

virtually identical to the slopes of the other lines. There is no significant difference between the average slopes of the various lines in Figure 4 ($p > .10$). We therefore reject H1.

Our second hypothesis to explain the increase in movement time observed in the docking task condition is an increase in ID resulting from a reduction of effective width [37] of the primary target. To understand why this might be the case, consider the nature of the docking task.

During docking, after acquiring the primary target participants must carefully centre the primary target over a restricted region (see Figure 3.12). When the participant performs this task, they may find it easier to accurately position the primary target within the docking region if the mouse cursor is centred over the primary target. If the mouse cursor is centred, the participant's task becomes a dual targeting task-first move the mouse to the centre of the primary target, then drag the mouse to the centre of the secondary target before releasing the mouse button. In other words, we hypothesize that the docking task may be simplified for the participant if the point of contact with the primary target is as close as possible to the centre of the target. This, in turn, requires that the user "shrinks" the effective size of the primary target to an effective size that is much smaller than the primary targets on-screen size.

To test this hypothesis, we used the error rate for each distance and width combination to calculate the effective width for each task condition as outlined by MacKenzie [37]. Analysis of variance showed no effect of task condition on effective target width ($F_{3,60} = 1.07, ns$). Therefore, we reject our second hypothesis and conclude the observed increase

of movement time does not the result from the decrease of effective target size.

Our final two hypotheses are that the user may spend more time either moving during the deceleration phase (H3) or hovering over the primary target after deceleration is complete (H4). To test these hypothesis, we redefine the two phases of movement time (MT) as a combination of the time to move from the initial location to the primary target (T_{move}), followed by the time a participant hovers over the final target (T_{hover}), i.e.:

$$MT = T_{move} + T_{hover} \tag{3.5}$$

To determine if the increase in movement time was due to an increase in T_{move}, T_{hover}, or both, we calculated T_{move}, and T_{hover} for each trial. The resulting means by task conditions are shown in Figure 3.16.

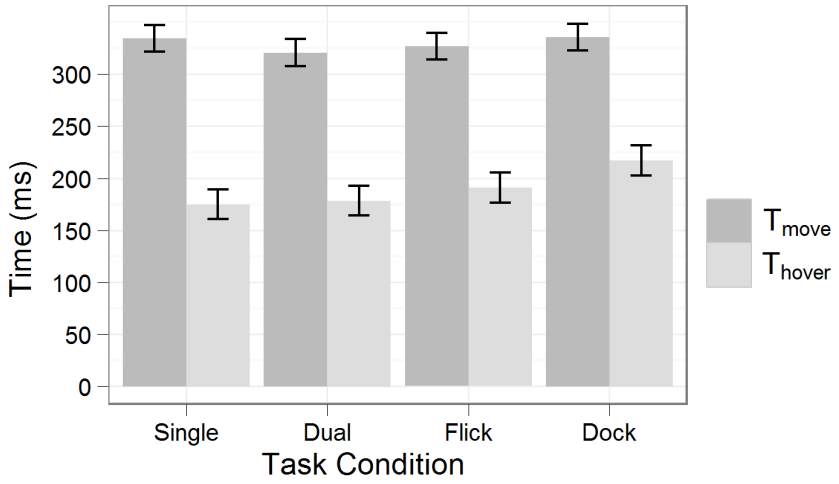


Figure 3.16: Means for T_{move} and T_{hover} by task condition.

We performed a RM-ANOVA on T_{move}, and T_{hover} with ID and task condition as factors. As expected, we find that ID has a significant effect on movement time, T_{move}, ($F_{4,342} = 817.95, p < .001, \eta^2 = .99$) and T_{hover} (F4,342=11.93, p ¡ .01,?2=.24). We also observed a significant effect of task condition on T_{hover} ($F_{3,342} = 5.37, p < .01, \eta^2 = .68$) but not on T_{move} ($p > .11$). Post-hoc analysis using Bonferroni correction shows the dock task condition to have significantly longer hover times than all other tasks ($p < .01$ in all conditions). As a result, we reject H3, and claim that movement time, T_{move}, is not affected by intended use. Furthermore, we are now able to reject the null hypothesis associated with H4. We claim that our data supports the hypothesis that the increase in movement

time observed is a result of an increase in the hover time over the primary target after motion stops. As well, our observations support the premise that the increase in hover time over the primary target is entirely responsible for the observed increase in movement time from an initial position to that primary target.

In summary, given the time to move to the target (T_{move}) is the same across all task conditions and no observed differences exist in the time required to reach peak velocity from the initial position, we can conclude that any difference in the $\%afterS_{max}$ measure and in movement time is the result of an increase in time spent motionless over the primary target after movement is completed.

3.4.4 Discussion

In Mandryk and Lough's research on intended use, the primary concern that they express in their paper is that their results may indicate a variation in the kinematic profile of user motion in Fitts-style targeting tasks depending on intended use of the primary target. While our research lays those concerns to rest, the observation that intended use affects movement time is both accurate and significant to modeling movement time with Fitts' Law.

One significant implication of these observations involves the mental processes that underlie compound tasks that involve targeting an on-screen object and then acting on that object. Our results suggest that it is possible that participants may view the mouse-down action over the target as the beginning of the drag task for our dock condition, as opposed to the end of the primary targeting task. In participants' mental model of target acquisition, it seems possible that positioning the cursor over the target, rather than clicking on the target, may signify completion of the targeting task. Other user interface tasks may also involve subtle aspects of serialization that must be teased out prior to advocating serial or parallel task assembly [6,9,20].

Furthermore, the validation of Mandryk and Lough's results for the dock task speaks to the relative cognitive cost of planning a dock versus flick, single or dual target task. In psychology research, the temporal cost of initiating an action is frequently used as a proxy of the relative cognitive cost of planning that action, i.e. if it takes longer to initiate a task, then the planning of the task must demand more cognitive load[56]. Mandryk and Lough's observation of the increase in time before beginning the dock task is an indication of the relative increase in cognitive load caused by docking versus the other tasks evaluated.

Our results have also clarified an inaccurate interpretation that may arise from Mandryk and Lough.Their experiments were motivated by the observation that kinematic profiles

differ when approaching objects in the real world based on intended use and prior experience. While in physical pointing, the affordance of an object may lead to a participant approaching the target at different speeds, virtual targets do not have the same properties as physical objects. Virtual objects have no friction to slow the hand, so users are unlikely to increase speed and rely on friction. As well, the user does not need to modify the orientation of their hand to drag versus select a target (i.e., using a mouse to control the cursor is the same whether the intended use involves a click or a drag). We should note that it may be the case that certain intended uses do cause modifications to the kinematic profiles of targeting tasks. However, the dock task studies by us and by Mandryk and Lough is not one such task.

3.5 Summary

In this chapter, we addressed the preliminary open questions related to modeling pointing motion in user interfaces. First, we demonstrated that it is not practical to incorporate the modeling of submovements into techniques that aim at identifying user endpoint by showing submovements often occur in the last 10% of motion distance. Next, we showed that while differences in the kinematic profiles can be detected between amplitude- and directionally constrained pointing motions, the differences are limited to the orthoginal properties of motion and are mitigated by the magnitude of the collinear motion. Lastly, we demonstrated that differences in movement time caused by the intended use of a target is the result from the user hovering over the final target and is not the result from any discernable kinematic changes in the intial balistic motion.

Given these findings, in the next chapter we present an endpoint prediction technique that models the initial ballistic impulse phase of pointing motion to predict gesture distance.

Chapter 4

Kinematic Endpoint Prediction

4.1 Introduction

In the previous chapter, we demonstrated that. to predict endpoint in the first 90% of motion using the underlying motion kinematic characteristics, we should focus on modeling ballistic movement. In this chapter, we begin by describing a taxonomy for endpoint predictors. Next, we present Kinematic Endpoint Predictor (KEP), a endpoint prediction technqiue that uses the minimium jerk principle [20, 66] to model the initial balistic motion to predict user endpoint. We begin by presenting the original technique proposed in collaboration with Dr. Edward Lank and a short description of the intial validation of the technique. This is followed by a revision of the technique to enable real-time identification of user endpoint.

4.2 Designing Endpoint Predictors for Pointing Facilitation

In this section, we revisit the previous work in pointing facilitation in order to describe a taxonomy for techniques that aim to predict gesture endpoint. We begin by classifying pointing facilitation into two categories, those that act on a widget (or target) and those that act on a pixel level. Next, we describe design characteristics an endpoint predictor should provide to support each of these categories. The resulting taxonomy presented in this section informs interaction designers on the requirements endpoint predictors must fulfill to provide support for pointing facilitation techniques.

4.2.1 Pointing Facilitation Techniques Categories

In Chapter 2.4, we introduced several proposed techniques to facilitate pointing in user interfaces (see Table 2.1). We are particularly interested in the techniques that may support pointing facilitation in modern interfaces and, therefore, focus our classification on those pointing facilitation techniques that have we have identified that may be able to support tiled widget arrangements[1]. The resulting pointing facilitation techniques can be classified into two categories, those that are target-based and those that are pixel-based. We define *target-based* techniques as those techniques that alter a target. For example, expanding widgets is a target-based technique in that the target is doubled in size on the display. In contrast, a *pixel-based* technique acts at the pixel level. Techniques that can be classified as pixel-based can often also be modified to be target-based. Semantic pointing [7] is one example of a technique that can be either pixel- or target-based, depending on whether device movement is altered using a continuous function over all pixels on the display (making it pixel-based) or in discrete steps over targets along the pointer's path (resulting in a target-based predictor). As we will see below, how we classify a pointing facilitation technique affects the requirements of a predictor.

4.2.2 A Taxonomy for Endpoint Predictors

Defining a taxonomy for endpoint predictors is not as straight forward as classifying pointing facilitation techniques mainly because very few predictors exists in the literature. Therefore, instead of creating a taxonomy based on current techniques, we must speculate on possible techniques in order to provide a complete taxonomy. The resulting taxonomy, shown in Table 4.1, consist of four dimensions; temporal, complexity, scope and execution.

The *temporal* dimension describes how often the prediction is performed during the motion gesture and is separated into two categories, *continuous* and *single-shot*. The continuous category describes a predictor that is continuously applied throughout the gesture. The single-shot category describes a predictor that is applied once during a gesture. For example, if a pointing facilitation technique uses a predictor to alter the display (for example, by expanding the predicted widget), a revision to the prediction is not possible.

The *complexity* dimension describes how many inputs are used by the predictor to predict gesture endpoint. A single-input predictor uses a single input. For example, the

[1]While other techniques listed in Table 2.1 can be classified using our categories, for simplicity we only include techniques of interest in this discussion

Taxonomy of Endpoint Predictors		
Temporal	Continuous	Prediction constantly is performed throughout the gesture.
	Single Shot	Prediction occurs once per motion gesture.
Complexity	Single Input	Uses a single input (e.g., a mouse)
	Multimodal	Uses multimodal input (e.g., a mouse and eye-tracker)
Scope	Learned	Prediction is based on learned behaviour.
	Derived	Prediction is deterministic and not based on learned behaviour.
Execution	Off-line	Prediction occurs in the future.
	Delayed	Prediction occurs shortly after the motion gesture has been completed.
	Real-Time	Prediction occurs during gesture motion.

Table 4.1: Taxonomy dimensions for endpoint predictors for graphical user interfaces.

predictor described in this thesis uses mouse input. A multimodal predictor uses more than one input to predict gesture input. For example, one can imagine using mouse motion and an eye tracker to predict gesture input.

The *scope* dimension describes the information needed by the predictor to predict endpoint. The *learned* category describes a predictor that leverages prior knowledge. An example of a learned predictor is the predictor we used in Chapter 3.3 to predict target constraint which used a trained HMM. In contrast, a *derived* predictor does not use prior knowledge but instead uses current information to predict endpoint. The predictor presented later in this chapter is an example of a dynamic predictor.

Finally, the *execution* dimension describes when prediction is performed and consists of three categories. The *off-line* category describes the situation where recognition occurs in the distant future, long after the motion gesture has been completed. An example of an off-line predictor is the HMM predictor we presented in Chapter 3.3 where prediction was done after all data was collected. A *delayed* category describes prediction that occurs slightly after the motion gesture is complete. An example of a delayed predictor is Schwarz et al.'s probabilistic toolkit [61]. The probabilistic toolkit examines subsequent actions in order to infer the intent of earlier actions. For example, imagine a user depressed the left mouse button over an on-screen widget but near a resize handle on a window. If the user releases the mouse button, then the user probably intended to target the widget. If, instead, the user begins to drag the mouse, then the user probably intended to drag the resize handle. Therefore, prediction of what a user clicked on is predicted based on the action shortly after the motion gesture has been completed. Finally, a *real-time* predictor performs prediction during gesture motion. As described by the temporal dimension, predictions can occur

once (single-shot) or throughout the motion gesture (continuous).

4.3 Kinematic Endpoint Prediction

Our goal is the prediction of the endpoint of single gesture movement. While path motion tends to move in a straight line directly toward the target [20], the length of the path, peak speed, and time varies. If we can predict the final length of a gesture at the beginning, midpoint, or some other sufficiently early intermediate point on the gesture, endpoint location in screen coordinates can be inferred by projecting along the direction of motion to the appropriate length.

When predicting endpoint, our goal is to extrapolate using a function fit to a partial gesture. Various approximation techniques can be used to determine values for gesture endpoint based on the above equations [52]. The most convenient representation would be an instantaneous speed versus distance graph, as we could fit a $(distance, speed)$ function and then calculate total distance directly from the function, thus eliminating any noise contribution from gesture time. To understand how instantaneous speed varies over distance, the minimum jerk model must be transposed from a speed signature over time to a speed signature over distance. Figure 4.1 depicts this relationship by plotting $(distance, speed)$ points calculated using Equations 3.2 and 3.3 over time interval $[0, 1]$. Also shown are three polynomial fitting functions: at the top, a quadratic polynomial (x^2), in the middle a quartic polynomial (x^4), and at the bottom a degree six polynomial, each with a least-squares fit polynomial and the polynomial's correlation.

4.3.1 Predicting Gesture Length

Endpoint prediction involves extrapolation of a best-fit polynomial to determine gesture length. This best-fit polynomial will describe the variation in speed over distance traveled. Neurophysiologists have also analyzed the path taken during motion, and note that the end-effector, in this case the pen or mouse, will typically follow a straight line [20] [49]. We can predict endpoint by extrapolating a distance, as we know that aimed motion follows a straight line path from starting point to ending point.

Extrapolation, particularly extrapolation of distant points, is a numerically unstable process [52]. As a result, it is desirable to use the lowest degree polynomial possible to extrapolate. While a polynomial of degree six has best fit, shown in Figure 4.1, there is a risk of over-fitting the data, which can affect extrapolation. Overfitting effects are

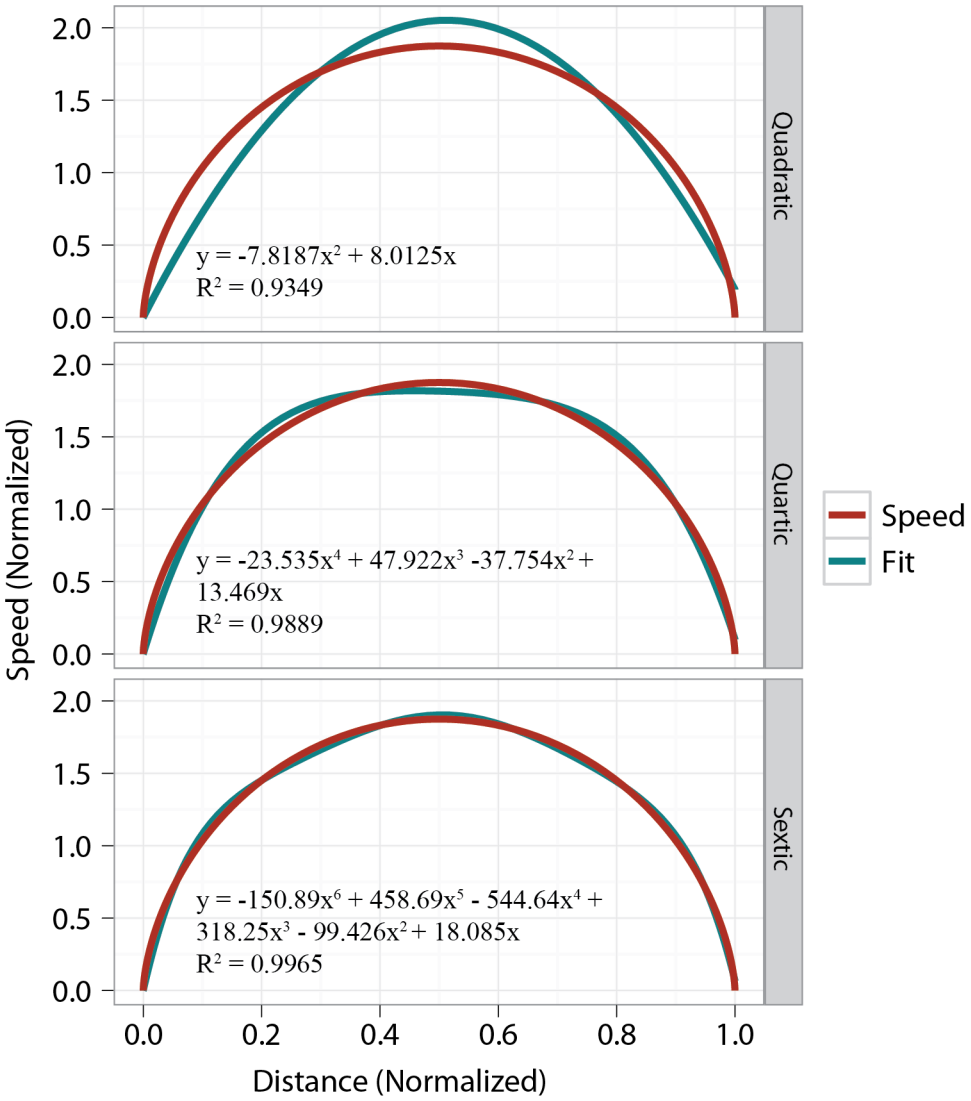


Figure 4.1: Theoretical speed versus distance profile predicted by the Minimum Jerk Law.

present in both the degree six and quartic polynomial. Figure 4.2 demonstrates the use of a quartic polynomial to extrapolate. The quartic function oscillates until reaching the last data point, and, rather than continuing smoothly, instead bends abruptly toward the x-axis.

Using least squares fitting on data points, we can calculate a quadratic polynomial that

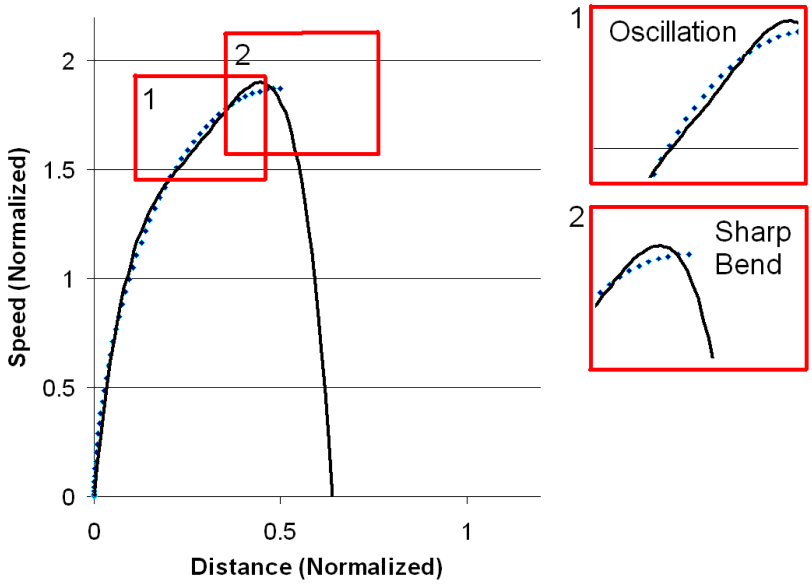


Figure 4.2: Fitting issues with a quartic polynomial include undesirable oscillation (1) and sharp bends rather than smooth continuity (2).

behaves regularly and predictably and use this polynomial to extrapolate. However, one challenge with degree two polynomial fits is that the velocity versus distance profile is not a perfect parabolic function. When we examine the fit of a degree two polynomial for the theoretical velocity versus distance plots in Figure 4.3 taken at 30%, 50%, 80%, and 90% of stroke length, the polynomial underestimates prior to 80%, is accurate at 80%, and then overestimates at 90%.

Two sub-optimal solutions present themselves. The first, attempting to fit a higher order polynomial, results in an inability to effectively extrapolate due to the need to predict a data point $(x, v(x))$ that is distant from our sampled data points during a partial gesture [52]. The second, fitting a lower-order, well-behaved polynomial is the established approach for performing distant extrapolation on data, but results in measurable inaccuracies in our predicted endpoint, even on theoretical data. While the standard rule of thumb for extrapolation is to use the lowest degree polynomial possible, what is needed in this case is some technique to correct measurable theoretical errors in extrapolated values. In this section, we describe an extrapolate-then-correct process for endpoint estimation. This technique is based upon the theorem that any polynomial function can be expressed as the product of degree-one and degree-two polynomials. While outside the scope of this thesis,

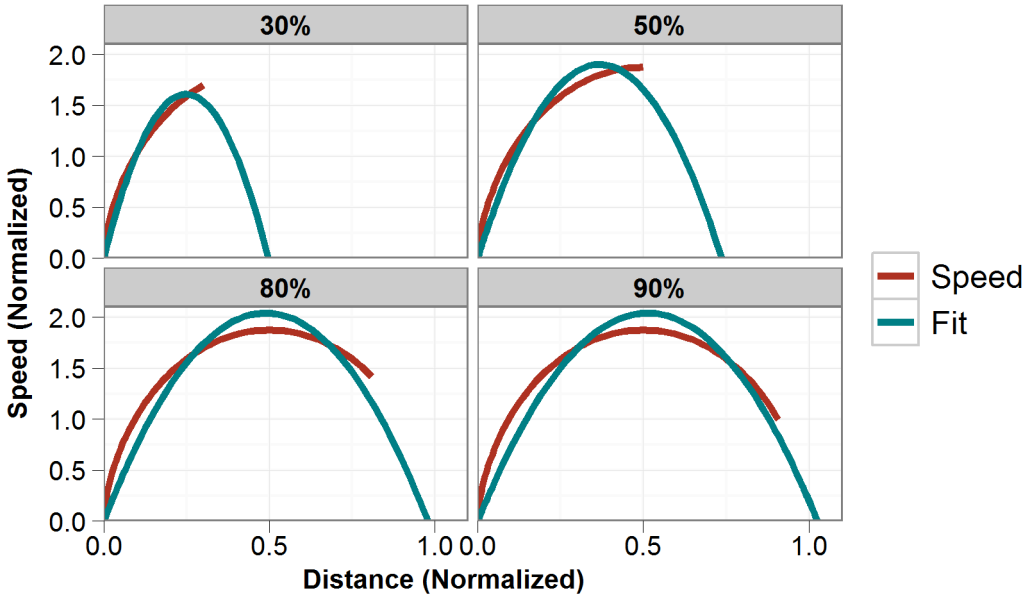


Figure 4.3: Fitting inaccuracies at 30%, 50%, 80% and 90% of gesture. At 80% of gesture, polynomial x-intercept and actual endpoint correspond perfectly.

the details of this argument can be found in Lank et al. [35].

Our solution to correctly estimate endpoint uses a set of coefficients calculated from the theoretical data produced by Equations 3.2 and 3.3 to correct the quadratic polynomial when fit to real data. The coefficients are determined by comparing the extrapolated value produced by a quadratic polynomial fit to theoretical data to the known endpoint. Table 4.2 depicts the coefficient obtained by dividing actual endpoint of our theoretical curves ($x = 1$) by the x-intercept of a quadratic polynomial fit to the first s_i fraction of data points in the stroke. The assumption is that real human motion will be represented sufficiently accurately by the laws that were used to create this data that it, too, will produce polynomial fits that exhibit similar inaccuracies, and that the same coefficients will apply. We can generate coefficients at an arbitrary density along the gesture extents by simply calculating endpoint, fitting, and tabulating the reciprocal of the value. In our implemented prediction algorithm, we currently tabulate coefficients at 1000 equally spaced points along the theoretical gesture. The coefficients, calculated only using Equations 3.2 and 3.3, are independent of actual subject motion.

Given these coefficients, predicting gesture length for real user motion is a two-step process. Given a partial gesture drawn by the user, we fit a quadratic polynomial to a

Stroke Index (s_i)	**Coefficient**
30%	2.01
40%	1.58
50%	1.36
60%	1.20
70%	1.09
80%	1.02
90%	0.97

Table 4.2: Coefficients to correct for predicted endpoint, as calculated on theoretical data. We use the values to correct estimation in actual gestures.

partial gesture's $(x, v(x))$ data points. One x-intercept occurs at $(0, 0)$; the other occurs at some location along the x-axis, x_{calc}. We seek a prediction for endpoint, x_{actual}. To determine x_{actual}, we must determine the coefficient by which x_{calc} must be multiplied. However, to determine the coefficient, we must also determine portion of gesture we have completed, s_i, which is also unknown. We determine coefficient and fraction of gesture numerically as follows.

Let us assume that a user has begun a gesture and has traversed distance d of the total intended gesture length, L. We wish to determine an estimate of L which we call x_{actual}. Given the user's partial gesture of length $d < L$, we can fit a quadratic polynomial to the $(x, v(x))$ data points of the partial gesture, giving us two pieces of data: the distance drawn from the starting point to the current point along the gesture, d, calculated based on Euclidean geometry (we know where the user started and their current location); and the x-intercept calculated from a quadratic polynomial fit to the data, x_{calc}, using least squares fitting of a quadratic polynomial to the entire set of points from beginning of the gesture to current location. Unknown are the coefficient, c_r, and the actual endpoint predicted by our formula, x_{actual}. Note that the coefficient, c_r, is a function of the fraction of the gesture that has been completed, s_i, tabulated above in Table 4.2. Two equations present themselves:

$$x_{actual} = c_r x_{calc} \tag{4.1}$$

$$d = s_i x_{actual} \tag{4.2}$$

Equation 4.1 is the mechanism we use for calculating our predicted endpoint, x_{actual}, multiplying the x-intercept by a specific coefficient, c_r. Equation 4.2 describes d, the distance traversed, as a function of fractional distance s_i from estimated endpoint x_{actual}.

Substituting x_{actual} in Equation 4.2 using Equation 4.1, we find that:

$$d = s_i c_r x_{calc} \tag{4.3}$$

Because c_r is a function of s_i based on tabulated values, we can numerically determine the values for s_i and c_r based on Table 4.2 that satisfy the equality in Equation 4.3. Once c_r has been determined, we can predict endpoint location x_{actual} using Equation 4.1. We determine c_r via exhaustive search, a process that takes about 1ms.

To summarize, we use the following real-time process to predict endpoint of a partially completed gesture:

1. Given a partial gesture of length $d < L$, L = total gesture length, we fit a quadratic equation to the data points $(x, v(x))$ along the partial gesture.
2. One x-intercept occurs at point $(0, 0)$, the other at a more distant point, x_{calc} along the x-axis. We determine x_{calc} by solving the quadratic polynomial for its roots.
3. Given x_{calc}, we use Equation 4.3 and Table 4.2 to determine a value for c_r.
4. We multiply c_r by x_{calc} to determine x_{actual}, an estimate of actual gesture length L.

In the following section, we analyze the accuracy of x_{actual} as a predictor of L, the actual gesture length. Following an analysis of the predictive ability of our model, we more fully analyze the theoretical underpinnings of our prediction process.

4.3.2 Initial Validation using Sylus-based Motion

Method

To validate our model, we asked ten subjects to draw 100 stylus targeting gestures on a 14 inch tablet computer with 1024x768 screen resolution. The length of the targeting gestures varied from 200 to 600 pixels by 100-pixel increments. Using circular targets, we varied target diameter between 15 and 75 pixels by 15-pixel increments. Subjects saw five different gesture-length/target-size pairs, counterbalanced using a 5x5 Latin square. Subjects drew 20 gestures for each of the five length/size combinations they were assigned. The order of presentation of individual length/size combination was randomized, as was the direction of the gesture. During a single gesture, subjects were presented with a start location. They depressed the stylus inside the start location. After a 1.5 second time-out,

a target was presented. Subjects drew a gesture to the target and lifted their pen inside the target region. Similar to directives in typical Fitts' Law experiments, subjects were asked to draw "as quickly as possible and as accurately as possible." The software captured location information in tablet ink coordinates and time in ticks to maximize data resolution for analysis. 35 of the gestures drawn were target misses, a 3.5% error rate which agrees well with ideal performance in Fitts' Law pointing tasks. We eliminated these gestures when doing analysis.

To analyze our data, we calculated speed and position for points along the gesture. Speed data was smoothed using an interpolating degree 2 polynomial. We fit portions of the gesture, specifically the first 15% 20%, 25%, 30%, etc. of (distance, speed) points in 5% increments of gesture length to generate endpoint predictions at locations along the gesture.

At each portion of the gesture, we compared our predicted endpoint with ground truth for the current gesture. Two candidates present themselves as possible ground truth values: the centre of the target and the observed endpoint of the gesture we are analyzing. Both produce similar error measurements. We chose target centre for two reasons. First, based on established laws of motion, a subject should aim toward the centre of the target, and the actual endpoint of their gesture should be normally distributed around that centre. The actual endpoint of any gesture is a result of an initial submovement and, potentially, secondary, corrective movements that occur after initial submovement. Those gestures requiring unpredictable secondary submovements would increase our error rate, while those without would reduce the error rate. Depending on the frequency of secondary submovements, prediction error might be biased either for or against our algorithm. Second, if we use gesture endpoint for the same target presented to the same subject twice, then each gesture produces its own endpoint and ground-truth is a gesture-specific measure. Repeatability of measurements does not exist, and analyses of the distribution of predictions are gesture-specific rather than condition-specific. Measuring accuracy by condition allows us to determine whether our prediction will be useful as an enabling technology, or whether it simply constitutes an intellectual exercise which, while still of value, has little practical relevance.

Results

Figure 4.4 shows the accuracy distributions for our endpoint predictions using a box and whisker plot. For each target size, endpoint prediction is plotted from 15% to 90% of gesture and is reported as distance from ground truth. Boxes contain 50% of the values; whiskers contain all non-outlier data values. Our best predictive power seems to occur

at approximately 80% of gesture length. This corresponds to 67% of initial submovement time, based on the equations for distance and speed in time, Equations 3.2 and 3.3, plotted in Figure 3.1. Based on work on expanding targets, 80% seems a convenient percentage of gesture length from which to predict endpoint location. At this point, 42.4% of target predictions fall within $\pm 0.5W$ of target centre, i.e. within the target, and an additional 39% of target values fall within $\pm 1.5W$, i.e. within the adjacent target (as shown in Figure 4.5).

Examining fitting accuracy at 80% of the gesture, we would expect target size to affect accuracy of prediction, as larger target size allows more tolerance for endpoint. Target effects can be observed in Figure 4.4. ANOVA of prediction accuracy (pixel error) for target size and distance shows a significant effect for target size, $F_{4,961} = 29.167$, distance, $F_{4,961} = 7.230$, and target*distance interaction, $F_{16,949} = 6.082$, $p < 0.01$ in all cases. Post-hoc pairwise tests (Tukey's) indicate significant ($p < 0.05$) differences between all targets except 15 and 30 and 45 and 60. Only distance of 600 pixels differs significantly, in its case from all other distances.

4.4 Real-Time Kinematic Endpoint Prediction

In our original work developing KEP, we use an extrapolate-then-correct procedure, where we fit a quadratic (degree-2) polynomial to speed versus distance profiles. This underestimates endpoint, so we use a set of tabulated coefficients to adjust the endpoint. The coefficients are calculated from theoretical data and converge to 1.0 at 80% of movement.

To simplify the predictor we combine two observations: prediction between 80% and 90% of movement is sufficiently early to allow a user to react [44]; and coefficients converge to 1.0 at 80% of movement and stabilize at that value. Since the coefficient approaches 1.0 as the user approaches 80% of gesture length, and we are more interested in realistic predictions than early predictions, we simplify KEP by eliminating the use of coefficients. Calculating speed and collecting speed-distance points is an easy task. Fitting a simple quadratic equation to this profile and calculating 0-speed points (i.e. distance-intercepts) is also computationally trivial. This simplifies, significantly, the implementation of the KEP algorithm.

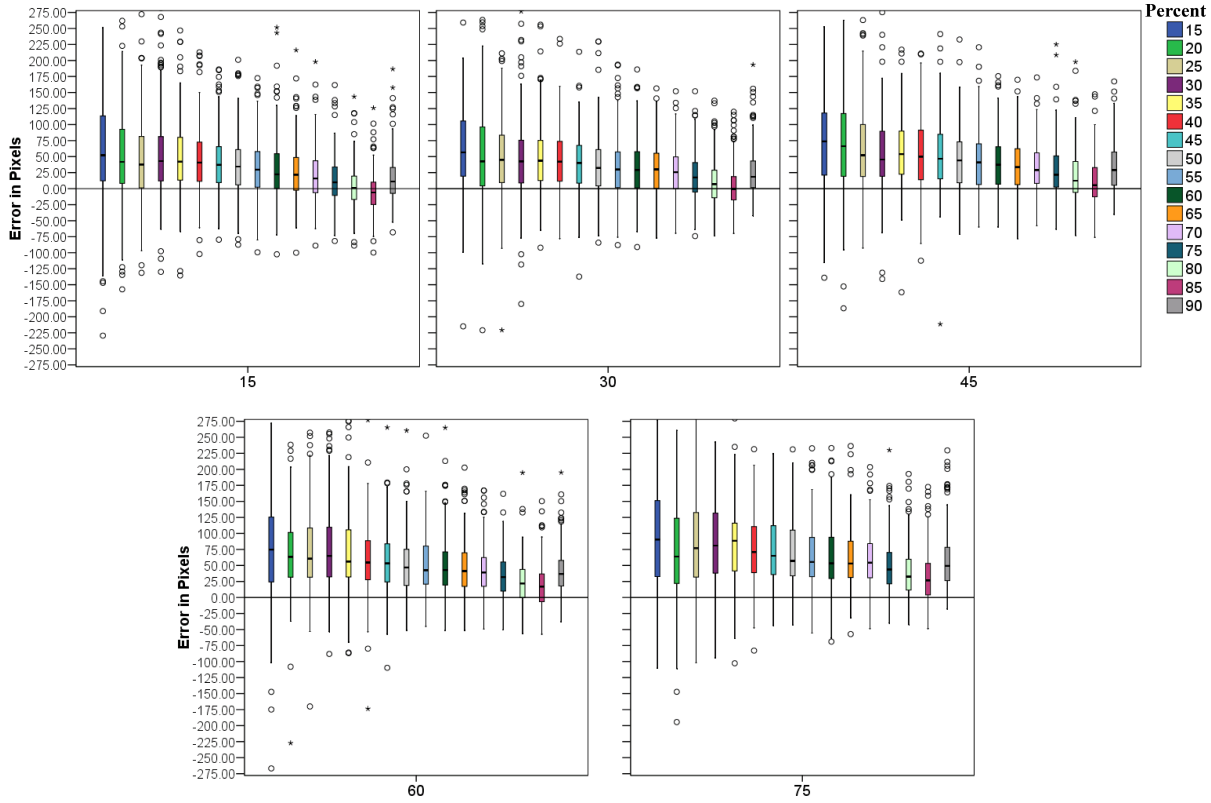


Figure 4.4: Predictive accuracy of KEP at locations along gesture path for stylus-based motion. Gesture path percentage is estimated based on $\frac{distance\_traveled}{predicted\_endpoint}$.

Managing Numerical Instabilities

One problem with endpoint prediction is that it uses curve fitting and extrapolation. Extrapolation, particularly of distant points, is known to be numerically unstable [35]. Numeric instability is particularly evident early in movement which is demonstrated by the predicted endpoints having significantly higher deviations than they do later in movement.

One problem with this numerical instability is that the algorithm frequently predicts that we have arrived at 90% of movement even when the endpoint is distant due to significant fluctuations in the predicted endpoint early in movement. One way to address this is to develop a measure of the predictor's stability.

In our revised technique, we capitalize on numerical instability to identify when an

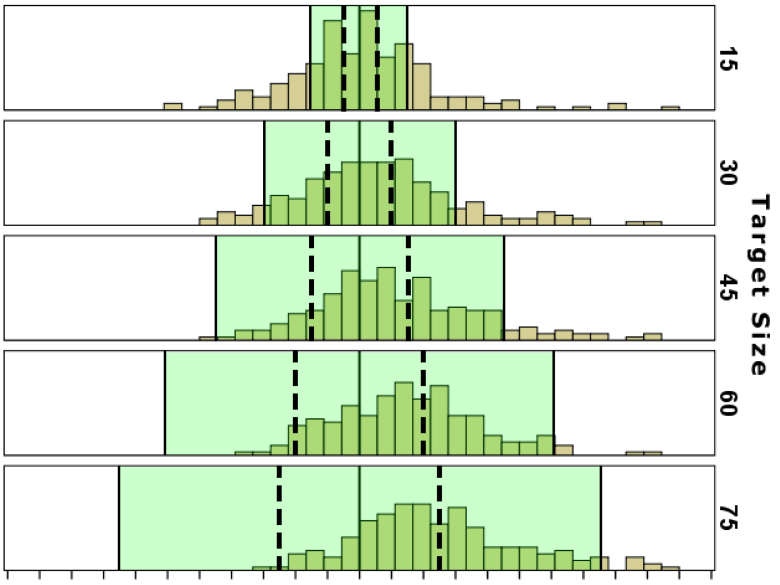


Figure 4.5: Histograms of endpoint predictions. Dashed lines representing ±0.5 target size and shaded regions representing ±1.5 target size are superimposed on the image. 42.4% of predictions fall within the dashed regions, i.e. within the target, while 81.4% of predictions fall within the shaded regions, i.e. ± one target, assuming tiled, collinear widgets of identical size.

accurate prediction is unlikely. During pilot studies, we observed that over the movement, our modified KEP stabilizes such that:

$$\frac{l_n - l_{n-1}}{l_n} \to 0 \tag{4.4}$$

Where l_n is current predicted length of the gesture and l_{n-1} is the previous predicted length for the current gesture. We call this value the stability of the prediction and find that values less than .02 represent that the predictor has stabilized, i.e. that the stabilization point has been reached.

A Refined KEP predictor for Real-Time Predictions

Using the above results, i.e. the elimination of the coefficients and the measure of numerical stability, we create a new, simplified KEP algorithm. First, we select a point along our

gesture where we would like prediction to occur, for example, at 85% of movement. Then, taking advantage of Equation 4.4 and the algorithm's predictive power later in motion, we use the following modified algorithm to calculate a single prediction for each movement collected:

1. Given a partial gesture of length d, a quadratic equation is used to fit the data points $(d, s(d))$ along the partial gesture.
2. Using the equation determined by step 1, calculate the roots of the equation. One root occurs at point $(0, 0)$ the other at a more distant point $(x_{calc}, 0)$.
3. Calculate the stability of the current prediction using Equation 4.4. If the prediction is determined to be unstable, i.e. Equation 4.4 returns a value greater than .02, return a value indicating an accurate prediction is not possible at this point in time.
4. Calculate the predicted percentage of gesture length completed by dividing the current distance traveled by x_{calc}. If the percentage is greater than the set target distance threshold, i.e. greater than 0.85 in our above example, return x_{calc} as our prediction; otherwise return a value indicating a prediction is not possible at this time.

4.5 Validation Study using Mouse-based Input

Since KEP was initially evaluated using an electronic stylus for input, a number of reservations have been expressed about the use of KEP for mouse-based input. These concerns revolve around the nature of the mouse for input, i.e. whether the kinematics of mouse motion are sufficiently similar to a stylus on tablet computer to support endpoint prediction with a mouse. The goal of the study presented in this section was to replicate our original validation study (presented in Section 4.3.2) using mouse-based motion. Since KEP is based on models of motion, we hypothesized that accuracies for mouse input would match the prior results using a stylus on a tablet computer.

Method

Our mouse-based validation study was conducted on a generic desktop computer (Core 2 Duo, 3.0GHz) with a 24-inch 1920x1200 LCD display running custom software written in C#. Input was collected using a Wacom Intuos3 five-button mouse on a 12x19 inch tablet

set to a 1:1 control display ratio. The 1:1 control display ratio ensured that motor space and visual space coincided throughout the pointing task as well as maintaining experimental validity and consistency with other studies [44, 2]. The Wacom tablet was used because of its high sampling rate. The custom software captured mouse movements at 200Hz.

The experimental task was a discrete, one-dimensional pointing task. As our goal is to contrast mouse pointing to stylus-based predictions, a one-dimensional pointing task preserves internal validity.

Initially a green starting rectangle was displayed on the screen. The task began when the participant used the cursor to hover over the starting location. After a period of 1 second, a red target would appear on the display. Participants were required to move the cursor to the red target and use the mouse button to click on the target. A successful target acquisition (i.e., clicking within the target region) was indicated by the target changing color. Users were told to acquire the target as quickly and accurately as possible, similar to other Fitts' Law experiments (e.g. [44, 2, 13]).

The study consisted of a within-subjects design with repeated measures. As in the original study, target distances (D) varied between 200-600 pixels in 100-pixel increments while target size (W) varied from 15-75 pixels in 15-pixel increments resulting in 25 D/W combinations.

The study consisted of two blocks: a practice block and an experimental block. Each block consisted of the 25 D/W combinations presented five times for each constraint, resulting in 125 tasks per block. The order of presentation of the D/W combinations was randomized. To minimize fatigue, participants were encouraged to take a five minute break between the practice and experimental blocks. The experiment took approximately 40 minutes to complete.

Eight male graduate students participated in the study. All participants were right handed. Of the 1,000 tasks recorded, 3.1% resulted in the user not hitting the target. These tasks were removed from our analysis.

4.5.1 Results

As shown in Table 4.5.1, we observe target accuracy rates similar to those for stylus motion. While we observed target accuracies of approximately 40% using stylus motion, for mouse motion we are typically predicting the correct target with almost 50% accuracy.

	Correct	Off-by-One
80% Gesture Length	49.3%	34.1%
85% Gesture Length	51.0%	35.8%
90% Gesture Length	51.4%	36.4%
Stylus Motion	42.4%	39.0%

Table 4.3: Observed target frequencies by percentage of gesture completed.

4.6 Discussion

In this Chapter, we presented our technique to predict motion endpoint using motion kinematics as well as a technique to measure the reliability of a given prediction by measuring the technique's "stability" at any given time. We demonstrated that for stylus-based motion we predict the user's intended target or an adjacent target 81.4% of the time. For mouse-based motion our prediction accuracy increases to 87.8%. As a result, our Kinematic Endpoint Predictor (KEP) is significantly more accurate than previous techniques [3, 44]. However, our techinque may have some limitations in its predictive power.

Table 4.4 describes the target accuracies for our user studies using stylus-based motion (Table 4.3(a)) and mouse-based motion (Table 4.3(b)) by target size and target distance. The highlighted cells represent target size and target distance combinations where accuracy of predicting the correct target fell bellow 41% (the average for stylus-based motion). As shown in the table, for the smallest target size (15px) and as distance increases, our ability to predict the user's intended target decreases dramatically for both stylus and mouse-based. These results suggest that there may exist limitations in our ability to predict small and/or distant targets. Given that a goal of this dissertation is to be able to predict goal directed motion endpoint on modern desktop displays, which contain small targets at large distances, further evaluation of the KEP predictor is needed in order to gain an understanding of the role target size and target distance plays in the ability to predict motion endpoint.

4.7 Summary

We began this chapter by exploring the design space of endpoint predictors and describing a taxonomy for endpoint predictors. Next, we presented KEP, our endpoint prediction technique and demonstrated that the technique can effectively predict endpoint for both

(a) Stylus-based Motion

	Target Distance				
Target Width	**200**	**300**	**400**	**500**	**600**
15	53.0%	10.2%	24.0%	12.5%	10.0%
30	47.4%	55.3%	43.6%	25.0%	35.9%
45	60.0%	42.5%	50.0%	62.5%	35.9%
60	53.1%	75.0%	42.1%	62.5%	65.0%
75	71.0%	43.2%	82.5%	72.5%	47.5%

(b) Mouse-based Motion

	Target Distance				
Target Width	**200**	**300**	**400**	**500**	**600**
15	44.1%	27.8%	23.7%	23.1%	19.5%
30	48.7%	51.3%	30.8%	46.2%	36.0%
45	67.6%	56.8%	53.8%	35.0%	47.4%
60	67.5%	50.0%	68.4%	68.4%	68.3%
75	80.6%	73.7%	67.6%	80.5%	50.0%

Table 4.4: Observed target accuracies for (a)stylus-based and (b)mouse-based motion by target width and target distance at 90% gesture length. Shaded regions indicate when target accuracies are below 40%.

stylus and mouse-based motion in user interfaces. We also presented a technique to measure stability of the predictor to enable real-time prediction.

Given our results suggest that their may be a limitation to the predictive power of the KEP technique (i.e., lower accuracies for small targets), the next chapter will examine the effects of target width and target distance on prediction accuracy.

Chapter 5

Performance and Evaluation of Kinematic Endpoint Predictor

Results presented in the previous chapter demonstrated that the kinematic endpoint technique (KEP) can successfully predict gesture endpoint for stylus and mouse-based motion. However, results from our two studies suggest that there may be an effect of target size and distance on prediction accuracy. In addition, our initial studies only examined one-dimensional (1D) targets, whereas, modern user interfaces consist of two-dimensional targets. Therefore, it is unclear if the nature of the targeting task, specifically the presence of two-dimensional (2D) targets and cursor acceleration for mouse-based input, may influence the ability of the algorithm to accurately identify motion endpoint. As a result, this chapter addresses the following three questions:

- How does the wider range of target Index of Difficulties (IDs) typical of desktop displays affect KEP accuracy?
- How accurate is KEP for predicting 2D targets?
- How does cursor acceleration affect the performance of KEP?

In this chapter, we expand the ID range of the validation study and show that the distance of target has a significant effect on predictor accuracy. Predictions at longer distances are less accurate than predictions at shorter distances. As well, the accuracy of KEP has an inverse linear relationship with distance traveled during pointing movement. We also demonstrate that KEP can be easily modified to handle 2D targets and that

predicting 2D targets has no significant effect on KEP's accuracy. Finally, we use our understanding of the relationship between movement distance and predictor accuracy to infer real-time probability distributions on target sets within an interface and demonstrate how this probability distribution can be leveraged to identify a small subset of high probability targets in the interface.

This chapter is organized as follows. First, we provide description of the analysis and measurements used throughout the chapter to describe the accuracy of the KEP. Next, we describe three experiments that analyze the accuracy of the KEP for mouse base input. We begin by examining the performance of KEP on 1D targets for a wide range of IDs typical of desktop displays. Using the result from this study, we show how KEP can be used to create probability distributions of likely targets in modern user interfaces. We then focus on 2D targets with and without the presence of cursor acceleration to determine if the results from the 1D study are generalizable. We conclude the chapter with a discussion of how the results from our studies can be leveraged to refine the initial target predictions produced by KEP.

5.1 Analysis and Measurements

We use two different prediction strategies to analyze the behaviour and performance of the Kinematic Endpoint Prediction (KEP) technique, *continuous* and *single-shot*. These two strategies correspond to the temporal dimension of the endpoint prediction taxonomy described in the previous chapter. The *continuous prediction strategy* consists of making a prediction at regular intervals throughout a motion gesture and describes the behaviour of the predictor throughout the motion gesture. The *single-shot prediction strategy* provides only one prediction per pointing gesture and represents the accuracy of the predictor when the system will act upon a prediction provided by KEP (for example, by expanding the predicted widget). In this section we begin by describing in further detail our prediction strategies. This is followed by a description of the measurements used to describe the accuracy and behaviour of the KEP technique for each of the prediction strategies.

5.1.1 Continuous Prediction

As stated above, continuous prediction examines the behaviour of the KEP technique throughout (i.e, from the beginning to end) the motion gesture. For each pointing gesture, we calculate speed and position for points along the gesture. To minimize noise caused by

pixelization, speed data is smoothed using an interpolating degree 2 polynomial. Using these speed points, we normalize distance and interpolate points at 1% increments of gesture length along the length of the gesture. Next, using the equidistance points we fit portions of the gesture, specifically the first 15% 20%, 25%, 30%, etc. of (distance, speed) points in 5% increments of gesture length to generate endpoint predictions at locations along the gesture as outlined by Section 4.3. It is important to note that with the continuous prediction strategy the predicted endpoint is multiplied by a coefficent due to our need to be able to generate a prediction throughout the gesture.

5.1.2 One-shot Prediction

The second analysis we perform uses a single-shot prediction strategy which more accurately represents how the predictor would be used in practice. As mentioned above, in real world situations, the KEP algorithm would be used to predict a target widget and something would happen to that target widget; for example, perhaps the widget would expand on the display so it is easier to acquire [44]. Using the KEP technique, we obtain a single prediction for each collected gesture by taking the first prediction returned by the algorithm. While it is possible that later predictions are more accurate, we assume that the first prediction is acted upon (for example, by expanding the predicted widget) and a revision to the prediction is not possible. One-shot prediction strategy relies on the Real-Time Predictor presented in Chapter 4.4.

5.1.3 Measurements

There are two measurements which are necessary when evaluating any endpoint predictor. The first is how accurate the algorithm is at predicting the user's distance, the prediction accuracy. The second is where during movement - at 50%, 80%, 90%, etc. - an accurate prediction occurs, i.e. the prediction location.

Whether we are using continuous or single-shot prediction, we evaluate the prediction accuracy of the predictor in two ways: *pixel accuracy* and *target accuracy*. Pixel accuracy (also referred to as pixel error) is the measurement from the predicted endpoint to the centre of the user's target in pixels (0 pixels is best). Thus, it is independent of target size. We choose target center (opposed to a gesture's actual endpoint) for two reasons. First, based on established laws of motion, a subject should aim toward the centre of the target, and the actual endpoint of their gesture should be normally distributed around that centre. The actual endpoint of any gesture is a result of an initial submovement and,

potentially, secondary, corrective movements that occur after initial submovement. Those gestures requiring unpredictable secondary submovements would increase our error rate, while those without would reduce the error rate. Depending on the frequency of secondary submovements, prediction error might be biased either for or against our algorithm. Second, if we use gesture endpoint for the same target presented to the same subject twice, then each gesture produces its own endpoint and ground-truth is a gesture-specific measure. Repeatability of measurements does not exist, and analyses of the distribution of predictions are gesture-specific rather than condition-specific.

Target accuracy is how accurately the predictor is able to correctly identify the user's target. We classify target accuracy into one of five groups:

- Correct: The predicted target is the user's intended target.
- Off-By-One: The predicted or following target is the preceding target.
- Negative/Positive Off: The category representing instances when the predicted target is a distant (more than one) target.

The second necessary measurement is prediction location, i.e. when (at what point during motion) predictions can reliably be made. As illustrated in Figure 3.2, the last 10% of gesture distance consumes over 40% of the total movement time. Therefore, any manipulation to a target before the last 10% of motion will result in the user having time to react. However, if prediction occurs too late - after 90% of the gesture, for example - when the user is trying to acquire the target rather than trying to cover the distance to the target, then the usefulness of the technique is questionable.

In the remaining of the chapter, we present several user trials that explore the effectiveness of the Kinematic Endpoint Predictor (KEP). For each trial, we will describe the accuracy of the KEP predictor in both target and pixel accuracy as described above. We will also examine the performance of the predictor both as a continuous and a single-shot predictor. We begin by examining the performance of KEP using target/distance combinations typical of modern desktop computer interfaces for both one-dimensional targets. Next, we examine if KEP can be adapted to two-dimensional targets and the effect cursor-acceleration and two-dimensional targets.

5.2 Distance and Target Effects on Kinematic Endpoint Prediction Accuracy

The goal of this study was to determine whether KEP can effectively predict user endpoint using target/distance combinations typical of modern desktop computer interfaces. Given our initial validation study suggests that that a limit may exist to predictor accuracy (i.e. small, distant targets had poorer accuracy), we also wish to determine whether distance, target width, or ID affect endpoint accuracy. Specifically:

- What is predictor accuracy for a broader range of target/distance combinations, i.e. for a broader ID range?
- What effect does distance have on KEP accuracy?
- What effect does target size have on KEP accuracy?
- Does interaction between target size and distance (i.e. does ID) affect KEP accuracy?

5.2.1 Method

The study was conducted on a generic desktop computer (Core 2 Duo, 3.0GHz) with a 24-inch 1920x1200 LCD display running custom software written in C#. Input was collected using a Wacom Intuos3 five-button mouse on a 12x19 inch tablet set to a 1:1 control display ratio. The 1:1 control display ratio ensured that motor space and visual space coincided throughout the pointing task as well as maintaining experimental validity and consistency with other studies [44, 2]. The Wacom tablet was used because of its high sampling rate. The custom software captured mouse movements at 200Hz.

The experimental task was a discrete, one-dimensional pointing task similar to both our stylus and mouse initial validation studies in order to preserve internal validity.

The task was identical to the task described in Chapter 4.5. Initially a green starting rectangle was displayed on the screen. The task began when the participant used the cursor to hover over the starting location. After a period of 1 second, a red target would appear on the display. Participants were required to move the cursor to the red target and use the mouse button to click on the target. A successful target acquisition (i.e., clicking within the target region) was indicated by the target changing colour. As in our validation studies, users were told to acquire the target as quickly and accurately as possible.

The study was within-subjects with repeated measures. The independent variables were distance and target width. Target widths of 4, 8, 16, 32, 64, 128, and 192 were each shown at distances of 512, 1024, and 1536 pixels. The resulting D/W combinations provided Indices of Difficulty (ID) between 1.87 and 8.59. The study consisted of a practice and experimental block separated by a 5 minute break. Each block consisted of the 21 D/W combinations presented to the user 10 times in random order. The experiment took approximately 50 minutes to complete.

Eight subjects participated in the study, 7 males and 1 female. All participants were affiliated with the local university and were right-handed.

5.2.2 Results

Of the 1920 tasks recorded, 4.1% resulted in the user not correctly hitting the target and were removed from our analysis.

Continuous Prediction

Overall target accuracies for predicting the user's target are shown in Table 5.1. As in our validation study, the highest target accuracy occurs at 90% of gesture length and not at 80% as observed with stylus-based motion. McGuffin and Balakrishnan [44] claim that prediction is useful up to 90% of movement distance.

	Correct	**Off-by-One**
80% Gesture Length	28.1%	23.4%
85% Gesture Length	31.1%	23.9%
90% Gesture Length	34.8%	24.7%

Table 5.1: Observed target accuracy frequencies by percentage of gesture completed.

At 90% of motion, KEP predicts the correct target approximately 35% of the time and is off-by-one 25% of the time. Examining effects of distance and target width on target accuracy indicates an effect for both distance and target width. While we expected target accuracy rates to be low for smaller targets (it is very hard to accurately predict a 4-pixel target), we also see a decrease in target accuracy as distance increases. As shown in Table 5.2, this is especially evident for large target widths. Analysis of variance on the accuracy of identifying the user's intended target with distance and target width as factors demonstrated there was a significant effect of target width ($F_{10,368} = 119.0$,

$p < .001$, partial $\eta^2 = .83$) and distance ($F_{6,368} = 9.61$, $p < .001$, partial $\eta^2 = .12$) on accuracy. However, there was no significant interaction between distance*target width on the frequency of predicting the correct target.

Post-hoc analysis using Bonferroni correction for percent correct showed significant differences ($p < .01$) between the shorter distances (512 and 1024-pixel) and the longest distance (1536-pixels). Post-hoc analysis also showed all target sizes to be significantly different ($p < .01$ in all cases) except for the smaller target sizes (4, 8, 16-pixels), where no significant difference was observed.

	Target Distance					
	512		**1024**		**1536**	
Target Width	**Correct**	**Off-by-One**	**Correct**	**Off-by-One**	**Correct**	**Off-by-One**
4	2.7%	13.3%	4.0%	4.0%	2.5%	7.8%
8	5.0%	10.0%	3.8%	12.7%	3.8%	8.8%
16	8.9%	24.1%	12.8%	16.7%	8.9%	10.1%
32	21.5%	49.4%	22.8%	31.6%	13.8%	10.1%
64	52.5%	40.0%	33.8%	48.8%	24.1%	75.0%
128	75.0%	23.8%	71.3%	28.8%	58.8%	40.0%
192	93.7%	6.3%	87.5%	12.5%	69.6%	30.4%

Table 5.2: Target accuracies of the KEP predictor by target width and target distance at 90% of gesture length.

The decrease in target accuracy as distance increases is a result of the distribution of pixel accuracy (pixel error) increasing with motion distance (shown in Figure 5.1). Analysis of variance of pixel accuracy by target width, and distance shows a significant effect for distance ($F_{2,270} = 4.21$, $p < .05$, $\eta^2 = .62$) but not for target width ($F_{6,270} = 1.91, p > .06$) or for a distance and target width interaction ($F_{12,270} = 0.46$, $p > .93$). Since we did not observe a distance and target width interaction, we conclude that ID does not have a significant effect on pixel accuracy. Instead target size and distance affect accuracy independently. Post-hoc analysis using Bonferroni correction shows a significant difference between all distances ($p < .05$).

Post-hoc analysis using Bonferroni correction shows a significant difference between the 1536-pixel distance and both the 512 and 1024-pixel distances ($p < .05$ in both cases). Post-hoc analysis for target width shows the 8-pixel target to be significantly different than the 128 and 192-pixel targets. No other distance or target size combinations differ significantly.

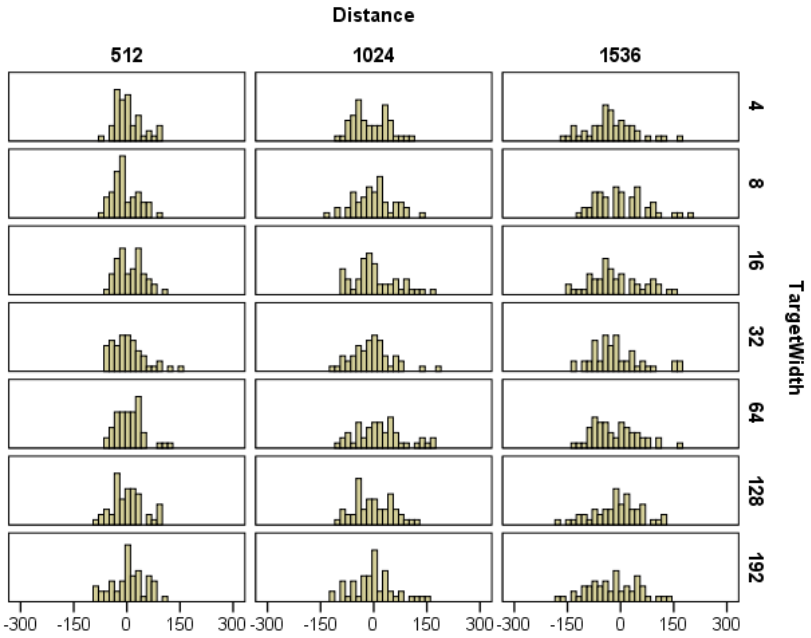


Figure 5.1: Distribution of pixel accuracy by distance and target width for continuous prediction.

Single-shot Prediction

We now focus our attention on our single-shot prediction analysis which represents how KEP would be used in practice. As stated above, to determine the point at which a prediction is made for the current movement, we examine if the algorithm has stabilized, and estimate the percentage of gesture distance traveled. Only when the predicted percentage of the movement distance reaches a set threshold do we log the endpoint. Since only one prediction is logged per movement, it is essential that the chosen threshold results in the highest possible accuracy.

While our first analysis showed that 90% of the distance of the movement resulted in the highest predictor, these results do not necessarily translate to our single-shot algorithm. To determine the appropriate threshold, we ran our analysis using thresholds of 80%, 85% and 90% of gesture distance. In addition, we also examine prediction location (i.e., at what point during motion prediction occurs). Prediction location is praticulary important to techniques that manipulate a target, for example, expanding widgets [44].

Table 5.3 lists target accuracies for our single-shot predictor for the three distance thresholds by prediction location. Although a 90% gesture length threshold results in the highest overall accuracy, this is a direct result of predictions occurring after 90% of actual gesture length. Eliminating predictions occurring after 90% of actual gesture length, a 90% threshold results in the lowest target prediction accuracy only identifying the correct target 16.9% of the time, whereas a threshold of 85% of gesture length results in the highest accuracy identifying the correct target 22.5% of the time and the adjacent target an additional 16.7% of the time. Using our single-shot predictor with 85% gesture length threshold, the user's target was predicted 22.5% of the time and an adjacent target an additional 21.0% of the time.

	Actual Gesture Length			
	Before 90%		**After 90%**	
Threshold	**Correct**	**Off-by-One**	**Correct**	**Off-by-One**
80%	21.3%	21.2%	0.2%	0.2%
85%	22.5%	21.0%	2.1%	2.2%
90%	16.9%	16.7%	12.4%	6.8%

Table 5.3: Observed target frequencies using single-shot prediction for varying thresholds categorized by percentage of gesture completed.

Examining effects of distance and target width on target accuracy using the 85% threshold (shown in Table 5.4), indicates an effect for both distance and target width. Similar to our observations with the continous predictor, we see a decrease in target accuracy as target width decreases and target distance increases. This is especially evident between large target widths at the 512 and 1536-pixel distances.

	Target Distance					
	512		**1024**		**1536**	
Target Width	**Correct**	**Off-by-One**	**Correct**	**Off-by-One**	**Correct**	**Off-by-One**
4	4.3%	2.2%	1.3%	1.3%	4.2%	2.1%
8	6.0%	4.0%	6.1%	6.1%	2.0%	2.0%
16	10.2%	14.3%	8.0%	12.0%	4.0%	4.0%
32	18.4%	32.6%	8.2%	25.7%	6.0%	26.0%
64	52.0%	44.0%	20.0%	28.0%	14.0%	38.0%
128	72.0%	28.0%	44.0%	52.0%	40.0%	48.0%
192	90.0%	10.0%	75.5%	24.5%	59.2%	38.8%

Table 5.4: Target accuracies of our single-shot predictor by target width and target distance using an 85% gesture length threshold.

Similar to the continuous predictor, the decrease in target accuracy as distance increases with the single-shot predictor is a result of the distribution of pixel accuracy (pixel error) increasing with motion distance (shown in Figure 5.2). However, distance seems to have a greater effect on the single-shot predictor than with the continuous predictor. As illustrated in the figure, the distribution of pixel error flattens as distance increases, more so than with our continuous predictor, and this significant flattening resulted in poorer target accuracy.

Analysis of variance for pixel accuracy at a 85% distance threshold shows a significant effect for distance ($F_{2,1840} = 18.10$, $p < .001$). Post-hoc analysis using Bonferroni correction shows a significant difference between all distances ($p < .001$ in all cases). We did not observe an effect for target width on pixel accuracy for our one-shot predictor.

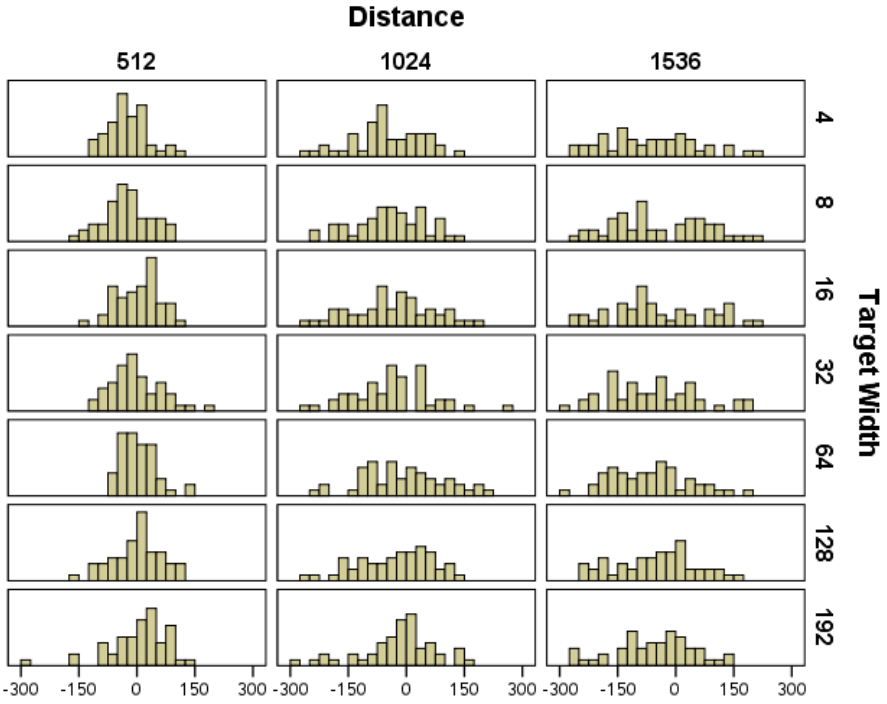
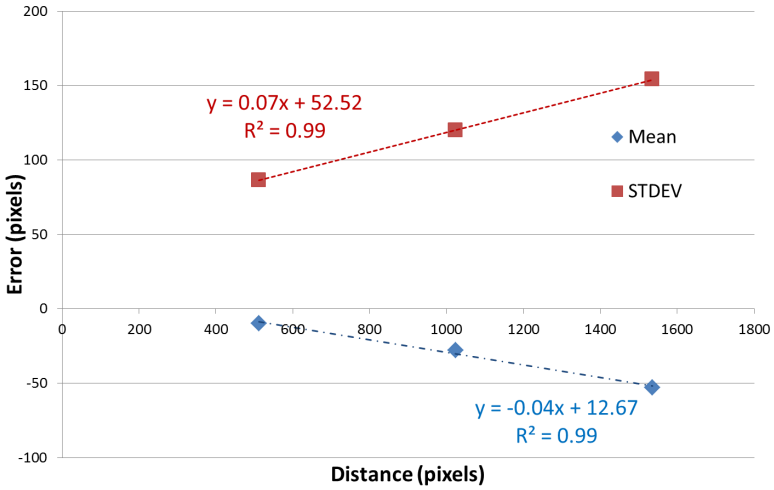
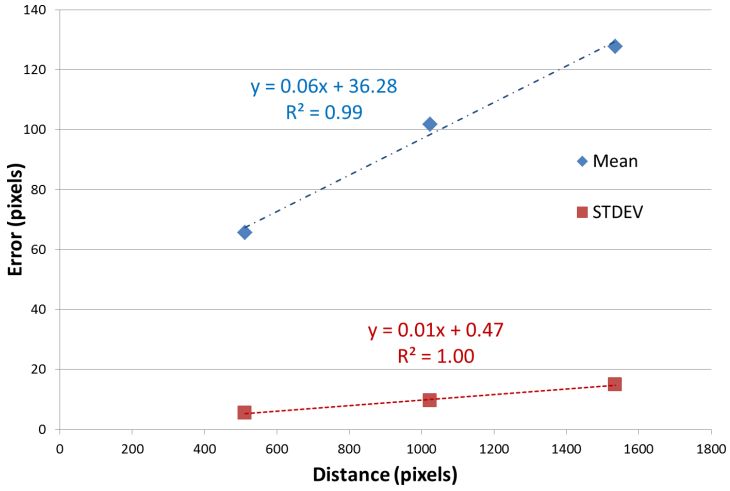


Figure 5.2: Distribution of pixel accuracy by distance and target width for single-shot prediction using an 85% gesture length threshold.



(a) Continuous Prediction Strategy



(b) Single-shot Prediction Strategy

Figure 5.3: Mean and standard deviation (STDEV) for pixel accuracy by distance of the KEP predictor for the continuous prediction strategy (a) and single-shot prediction strategy (b).

Discussion

A major goal of this study was to identify how the Kinematic Endpoint Prediction (KEP) algorithm performs using mouse input with a wide range of target sizes and distances, and to determine how to adapt the algorithm to enable pointing techniques that require target identification.

To examine the performance of the KEP algorithm, we used two prediction strategies, continuous and single-shot. The continuous predictor allowed us to examine the accuracy of the algorithm throughout a gesture, whereas the single-shot predictor allowed us to simulate real-time prediction to support pointing facilitation techniques that manipulate a target. For each experiment conducted, we analyzed each predictor on how accurately it was able to identify the user's intended target (target accuracy) and its distance from the center of the target (pixel accuracy). Using the continuous predictor, we demonstrated that, for mouse input, the highest target accuracy occurred at 90% of motion. We have also shown that there exists a relationship between distance and prediction accuracy.

In our results, we take a pessimistic approach to prediction accuracy that post-90% predictions may be unusable. As a result, we may be giving a mistaken impression that the KEP algorithm we developed has poor accuracy, but this is definitely not the case. When we consider continuous prediction for a broad range of target size/distance pairs at 90% of movement (Table 5.1), we see that we have 35% accuracy for predicting target, and that we predict an adjacent target an additional 25% of the time. In other words, 60% of the time, we can predict a small subset of targets of interest on the display, even if we assume that every single location on the display is a salient target, i.e. that targets are not separated by white space.

Interaction techniques could be designed that allow continuous prediction instead of requiring single-shot prediction, and these would enjoy our best-case performance. As well, setting our distance threshold to 0.90 resulted in many predictions occurring after 90% of motion, but these predictions may be occurring just after 90% of motion: Distance thresholds of 0.88 and 0.89 may preserve the high accuracy of 90% single-shot prediction while increasing distance from the user's final target. In summary, our predictor shows promise if new pointing facilitation techniques can be designed that take its behavior into account.

However, as we note, many interaction techniques - expanding widgets, being a prime example - may require single-shot prediction, and our KEP may have too low an accuracy to support these techniques. How, then, to boost the accuracy of a prediction? It is with this in mind that we focus our attention on the link between distance and pixel accuracy (i.e. the pixel-spread of endpoints).

Figure 5.3(a) illustrates the mean and standard deviation of pixel error by target distance for our continuous predictor at 90% gesture length. As shown in the figure, a strong linear correlation exists between distance and both mean and standard deviation. In the case of mean, $R^2 = 0.99$ ($p < .05$), indicating that we underestimate the center of the target, and that this underestimate is linearly correlated with how far a user moved. Furthermore, $R^2 = 0.99$ ($p < .01$) for standard deviation showing that standard deviation is directly proportional to distance of motion.

We also observe a strong correlation for mean ($R^2 = 0.99$, $p < .05$) and standard deviation ($R^2 = 0.99$, $p < .01$) for our singe-shot predictor (shown in Figure 5.3(b)). Similar to our continuous predictor, our one-shot predictor underestimates the center of the target, and this underestimate is linearly correlated with distance traveled. These numbers also hold for our validation study, where we see correlations of $R^2 = 0.95$ ($p < .05$) for mean and $R^2 = 0.99$ ($p < .001$) for standard deviation on 5 data points.

Given the strong linear relationship between distance and the mean and standard deviation of our predictions, we can make use of this knowledge to define a centered normal distribution on a target region. This normal distribution would be expected to be centered on a slightly adjusted predicted endpoint. The adjusted endpoint is calculated using the linear equation for the mean of pixel accuracy from Figure 5.3(a), where y becomes the offset to subtract from the predicted distance (x) to obtain the new adjusted prediction. Figure 5.4 compares the original and adjusted distributions which are now centered around 0 as a result of the shift. Given the distribution is now centered, we are able to define a likelihood region around the predicted endpoint. For example, at a distance of 1536 pixels we can claim that the user's target will be enclosed by a ± 153 pixel (i.e., $\pm$ one standard deviation) region from the predicted endpoint 68.2% of the time. In summary, because of the linear relationship between distance and pixel accuracy we can calculate a probability distribution around our predicted endpoint.

Summary

In this section, we examined an extensive range of IDs (from 1.87 to 8.59) to determine whether distance, target size, or ID has an effect on prediction accuracy. Our results demonstrate that target size, distance and a target width and distance interaction has an effect on target accuracy (i.e. correct or off-by-one accuracies) for both the continuous and single-shot predictors. We also demonstrate that target size and distance has an effect of pixel accuracy. However, we did not observe a target size by target distance interaction for pixel accuracy. Therefore, we conclude that ID does not have an effect on pixel accuracy.

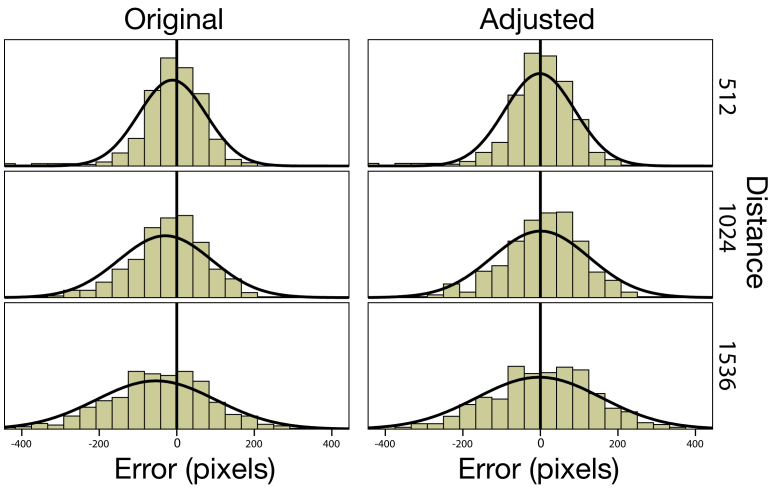


Figure 5.4: Distributions of pixel accuracy for our continuous predictor before and after applying the offset calculated using the linear correlation between distance and prediction accuracy mean.

We also demonstrate that their exists a strong correlation between the mean and standard deviation of pixel accuracy and distance. Leveraging this correction, we demonstrate that KEP (using either the continuous or single-shot predictor) can be used to identify the probability of any target (or pixel) on the display being the user's intended target.

In the next section we explore the performance of the KEP technique to predict two-dimensional targets with and with-out pointer acceleration.

5.3 Distance and Target Effects on Kinematic Endpoint Prediction Accuracy with Two-Dimensional Targets

In our previous study we examined one-dimensional targets for a broad range of target size/distance pairs. While these are significant contributions to understanding the performance of the KEP technique for mouse-based pointing on desktop sized displays, several questions still remain when considering predicting two-dimensional (2D) targets.

Endpoint prediction for 2D targets requires predicting both distance and path. Errors

can occur as a result of mispredicting distance, mispredicting the intended path of the user, or from mispredicting both distance and path. In this section, we evaluate 2D KEP by performing two experiments. In the first experiment, we continue to use a 1:1 control display ratio to maintain experimental validity and consistency with the first two studies and prior work in this area [44, 2, 74, 23]. Next we evaluate 2D KEP with pointer acceleration to determine the effectiveness of the algorithm in more realistic interface configurations.

5.3.1 Two-Dimensional Endpoint Prediction

To support 2D target prediction, the KEP algorithm must be revised to predict both distance and path. As in our previous studies, we first predict gesture distance. Using the predicted distance, we calculate the remaining distance by subtracting the predicted distance from the current gesture length. Next, we use linear least squares fitting (LLSF) on the previous 10% of the movement to predict the movement path. The predicted endpoint is calculated by using the line determined by our LLSF and finding a point on that line where the distance from the current point to the predicted point is equal to the predicted remaining distance. This process is illustrated in Figure 5.5.

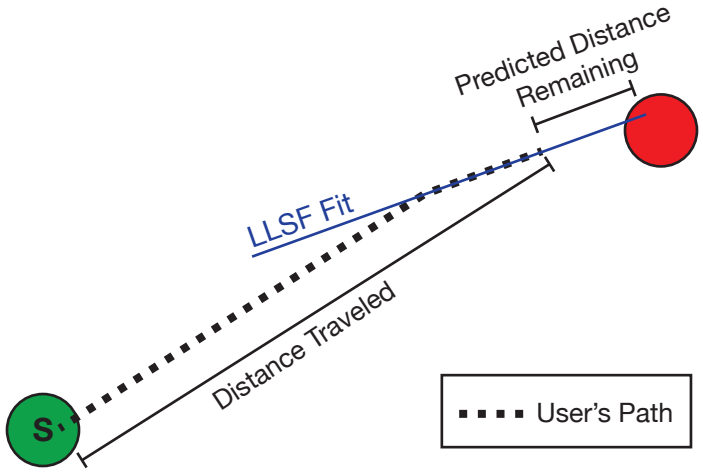


Figure 5.5: An illustration of predicting endpoint in 2D using a gesture collected from our study. The line determined by our linear least squares fitting (LLSF) (shown in blue) is used to determine the endpoint location using the remaining distance predicted by the KEP technique.

5.3.2 User Trial: 2D Targets Without Cursor Acceleration

The goal of our first 2D target experiment is to compare the accuracy of the KEP technique to our one-dimensional target studies.

Method

The studies were conducted on the same system used in our one-dimensional (1D) targeting studies. The experimental task was a discrete two-dimensional pointing task based on the ISO 9421-9 standard [28] (shown in Figure 5.6). Eight circular targets were arranged in a circle with a radius of D. At the onset of the trial, the targets were colored gray except for a blue starting target. The task began when the participant moved the cursor into the starting area and hovered for approximately one second. At that time, a red target would appear on the display. As in our previous Fitts' tasks, participants were required to move the cursor to the red target and use the mouse button to click on the target as quickly and accurately as possible. A successful target acquisition (i.e., clicking within the target region) was indicated by the target changing color. After clicking, the task would continue with the previous target becoming the new start target. This sequence would continue until all eight targets were traversed (resulting in 9-targeting tasks per arrangement). Input was collected using the same Wacom Intuos3 five-button mouse used in our prior studies. For internal validity with our prior study, the mouse was set to a 1:1 control display ratio.

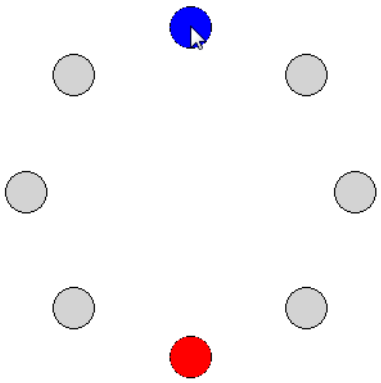


Figure 5.6: Screenshot of the task used for our 2D targeting studies.

The study consisted of a within-subjects design with repeated measures. The independent variables were distance and target width. Target widths of 8, 16, 32, 64, and 96 were each shown arranged in a circle with a radius of 256, 512, 768 and 1024 pixels.

The resulting D/W combinations provided Indices of Difficulty (ID) between 1.59 and 7.01. The experiment consisted of a practice and three experimental blocks separated by a five minute break. Each block consisted of the 20 D/W target arrangements presented to the user and resulted in 540 pointing tasks collected for each participant.

Target accuracy was computed in the same manner as in our $1D$ target experiments described above. Since $2D$ targets require us to predict path direction, we report two accuracies. The first is observed accuracy, where a target is correct if both distance and path are predicted correctly. The second is perfect path prediction accuracy, where we assume path is correct and examine distance accuracy. Reporting both values allows us to assess the effect of path inaccuracies on KEP.

Eight subjects participated in the experiment, five males and three females. All participants where affiliated with the local university and were right-handed.

Results

Of the 4320 tasks recorded, 2.0% resulted in the user not correctly hitting the target and were removed for analysis.

Continuous Prediction Overall accuracies for predicting the user's target are shown in Tables 5.5 and 5.6. As in the 1D targeting studies, the highest target accuracy occurs at 90% of gesture length and not at 80% as observed with stylus-based motion. Analysis of variance on percent correct with distance and target width as factors shows a significant effect for distance ($F_{3,320} = 92.2, p < .001, \eta^2 = .25$) and target width ($F_{4,160} = 303.9$, $p < .001$, $\eta^2 = .56$). We also observed a significant interaction between distance and target width ($F_{12,368} = 32.6$, $p < .001$, $\eta^2 = .18$).

	Observed		**Perfect Path Prediction**	
	Correct	**Off-by-One**	**Correct**	**Off-by-One**
80% Gesture Length	19.5%	32.9%	21.7%	31.3%
85% Gesture Length	22.6%	33.5%	24.6%	31.7%
90% Gesture Length	26.6%	35.3%	27.6%	33.6%

Table 5.5: The observed target accuracies and target accuracies assuming perfect path prediction for our continuous predictor.

Post-hoc analysis using Bonferroni correction showed a significant difference in target accuracy between all distances ($p < .001$ in all cases) with the accuracy decreasing as

	Target Distance			
Target Width	**256**	**512**	**768**	**1024**
8	3.1% (21.3%)	1.3% (10.6%)	1.2% (6.0%)	1.3% (2.6%)
16	12.6% (56.9%)	3.7% (25.8%)	3.6% (22.8%)	0.6% (10.1%)
32	32.9% (57.3%)	11.4% (63.3%)	11.0% (48.5%)	4.9% (39.6%)
64	72.5% (25.7%)	43.6% (50.3%)	28.3% (63.3%)	18.8% (65.5)
96	89.8% (9.0%)	65.7% (33.1%)	49.1% (49.1%)	39.8% (35.6%)

Table 5.6: Observed correct and off-by-one (in parentheses) target accuracies of our continuous predictor by target width and target distance at 90% gesture length.

distance increased. Post-hoc analysis also showed accuracy for all but the 8 and 16-pixel targets to be significantly different ($p < .01$ in all cases).

To determine what effect needing to predict motion path had on our observed accuracies, we performed a paired t-test and found prediction technique (with or without path prediction) did not have a significant effect on our ability to predict the participant's intended target ($p > .20$).

To measure the pixel accuracy for 2D targets, pixel error is separated into two dimensions: *orthogonal error* and *collinear error*. Orthogonal error represents the distance from the predicted endpoint to the target center that is perpendicular to the path of motion. Collinear error represents the amount of pixel error in the path of motion. In the 1D target experiment where we did not need to predict direction, all pixel error was calculated as collinear error.

As shown in Table 5.3.2, analysis of variance of collinear error and orthogonal error by target width and distance shows a significant effect for distance on collinear and orthogonal error. We do not observe any significant effects for target width on orthogonal or collinear error. Post-hoc analysis using Bonferroni correction for both collinear and orthogonal error shows a significant difference between all distances ($p < .05$).

Pixel Accuracy	**Distance**	**Width**	**Distance*Width**
Collinear	$F_{3,78} = 39.18$, $p < .001$, $\eta^2 = .90$	$F_{4,78} = 1.03, ns$	$F_{12,78} = 0.75, ns$
Orthogonal	$F_{3,78} = 48.7$, $p < .001$, $\eta^2 = .95$	$F_{4,78} = .93, ns$	$F_{12,78} = 1.03, ns$

Table 5.7: F and p-values for within-subjects ANOVA on pixel accuracy by error type. Shaded cells show no significance ($p > .05$).

Our study using 1D targets demonstrated that a major benefit of the KEP technique is the ability to define a probability region around a set of targets. To determine if the same

benefit can be observed with 2D targets, we examine the distributions of pixel accuracy in the collinear and orthogonal directions. Figure 5.7 illustrates the distribution of the collinear and orthogonal pixel error by distance. Similar to our findings with 1D targets, the pixel accuracy is normally distributed.

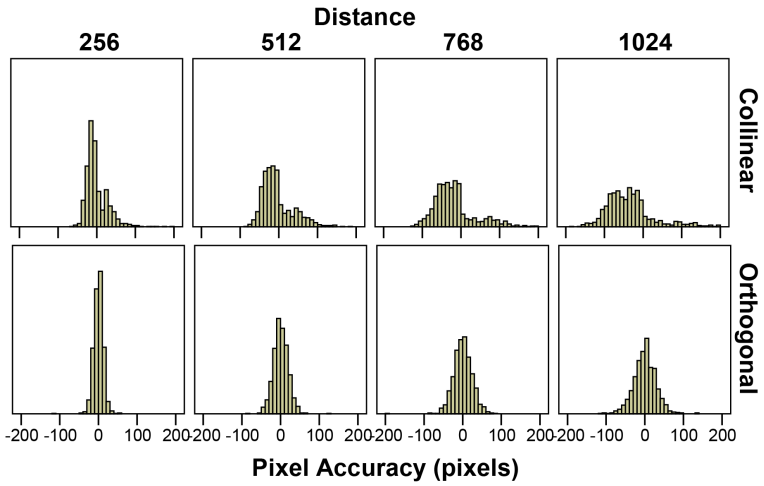
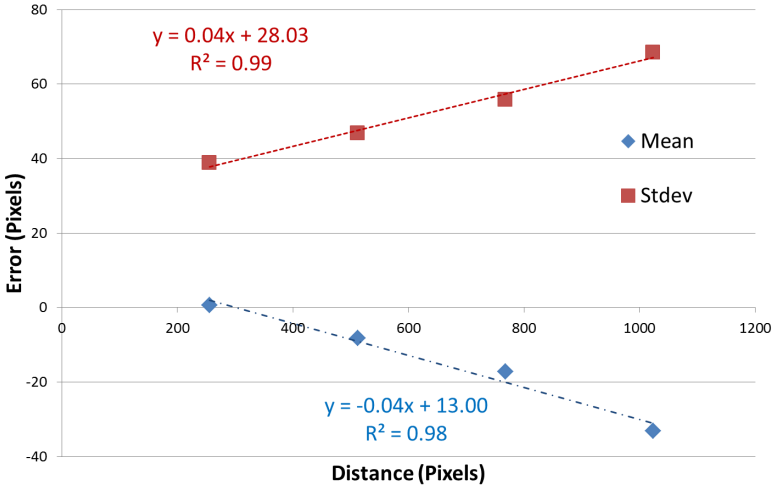


Figure 5.7: Distributions of collinear (top) and orthogonal (bottom) pixel accuracy by distance for the continuous predictor.

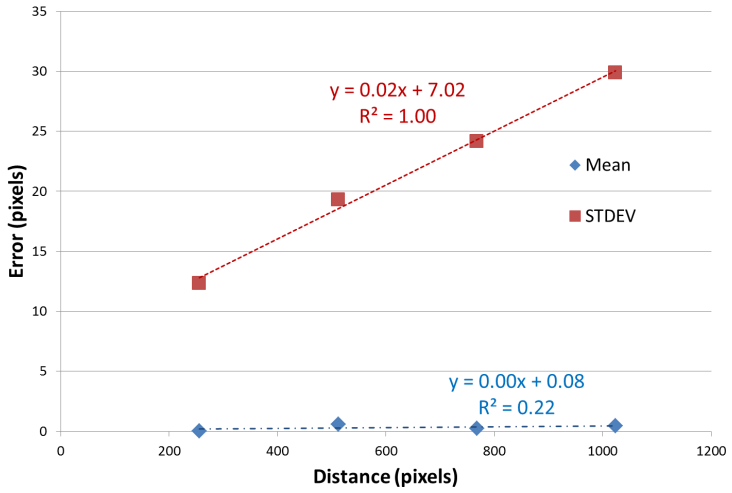
In addition, we observe a strong linear correlation between distance and the standard deviation of both collinear and orthogonal error ($R^2 = 0.99, p < 0.01$ and $R^2 = 0.99, p < 0.01$ respectively). A strong correlation of $R^2 = 0.98$ ($p < 0.05$) is also observed between distance and the collinear mean. Therefore, the distributions for the pixel errors can be shifted similar to Figure 5.4. However, as shown in Figure 5.8, we did not observe a correlation between distance and the mean of the orthogonal error.Given the standard deviations are less than two pixels, it is unnecessary to shift the observed distribution. Therefore, the lack of correlation for mean has no effect on our ability to identify likely targets.

We now examine the prediction accuracy of KEP using a single-shot prediction strategy.

Single-Shot Prediction Similar to our one-dimensional study described in Section 5.2.2, we examine various thresholds to determine prediction accuracy. We also examine what effect of needing to predict motion path had on our observed accuracies. A summary of the results are shown in Table 5.7(a). As we saw in the previous study, the single-shot predictor target accuracy is less accurate than our continuous prediction strategy, and



(a) Collinear Pixel Accuracy



(b) Orthogonal Pixel Accuracy

Figure 5.8: Mean and standard deviation (STDEV) of pixel accuracy by distance for the continuous prediction (a) and single-shot prediction (b) strategies.

higher thresholds result in predictions occurring beyond 90% of gesture length. Therefore, we concentrate our analysis in the remaining of this section using a threshold of 85%.

Analysis of variance on percent correct with target width and distance as factors shows a significant effect for distance ($F_{3_{78}} = 165.3$, $p < .001$, $\eta^2 = .15$) and target width ($F_{4_{78}} = 693.3$, $p < .001$, $\eta^2 = .81$). We also observe a significant distance and target width

interaction ($F_{12_78} = 11.4$, $p < .001$, $\eta^2 = .04$). These effects can clearly be seen in Table 5.9.

(a)

	Actual Gesture Length			
	Before 90%		After 90%	
Threshold	**Correct**	**Off-by-One**	**Correct**	**Off-by-One**
80%	12.4%	26.5%	0.6%	0.4%
85%	13.8%	27.0%	1.6%	2.1%
90%	12.0%	25.0%	6.5%	5.1%

(b)

	Actual Gesture Length			
	Before 90%		After 90%	
Threshold	**Correct**	**Off-by-One**	**Correct**	**Off-by-One**
80%	13.1%	24.7%	0.7%	0.4%
85%	14.0%	25.7%	2.4%	1.7%
90%	7.9%	5.3%	11.8%	23.5%

Table 5.8: The observed target accuracies for our single shot predictor by actual gesture length. (a) Observed target frequencies. (b) Target frequencies given perfect path prediction.

	Target Distance			
Target Width	**256**	**512**	**768**	**1024**
8	1.5% (14.1%)	0.5% (6.2%)	2.4% (6.1%)	1.5% (1.5%)
16	5.6% (22.9%)	6.2% (14.2%)	1.4% (9.5%)	2.9% (5.8%)
32	15.8% (48.8%)	12.7% (33.3%)	6.6% (22.7%)	3.3% (15.2%)
64	41.0% (57.5)	29.6% (50.2%)	22.0% (45.3%)	10.3% (30.8)
96	55.9% (44.1%)	39.4% (51.2%)	29.8% (49.8%)	17.3% (50.0%)

Table 5.9: Observed correct and off-by-one (in parentheses) target accuracies of our single-shot predictor by target width and target distance using an 85% threshold.

Similar to the pixel-accuracy analysis of the continuous predictor presented above, pixel error is separated into orthogonal and collinear error. As shown in Table 5.3.2, analysis of variance of collinear and orthogonal error shows a significant effect for distance on collinear

error. We did not see any effect for target width on collinear error or any effect for distance or target width for orthogonal error. Post-hoc analysis using Bonferroni correction for collinear error shows a significant difference between all distances ($p < .05$).

Pixel Accuracy	Distance	Width	Distance*Width
Collinear	$F_{3,78} = 48.17$, $p < .001$, $\eta^2 = .72$	$F_{4,78} = 0.95, ns$	$F_{12,78} = 0.43, ns$
Orthogonal	$F_{3,78} = 0.57, ns$	$F_{4,78} = 0.26, ns$	$F_{12,78} = 0.70, ns$

Table 5.10: F and p-values for within-subjects ANOVA on pixel accuracy by error type. Shaded cells show no significance ($p > .05$).

Examination of the mean and standard deviation of pixel error show results similar with the continuous predictor. As shown in Figure 5.9, we only observe a correlation for collinear mean ($R^2 = 0.96$, $p < .01$) and standard deviation ($R^2 = 0.99$, $p < .01$). We also see a strong correlation for standard deviation of orthogonal error and distance ($R^2 = .99$, $p < .01$). Again, we observe very low mean values for orthogonal error, suggesting orthogonal error distributions are already centered around 0 and does not need to be corrected.

5.3.3 User Trial: 2D Targets With Cursor Acceleration

A goal of this dissertation is to determine if the KEP technique is applicable to desktop computing. Since cursor acceleration is the default behavior on modern operating systems, it is important to understand the effects cursor acceleration may have on KEP accuracy.

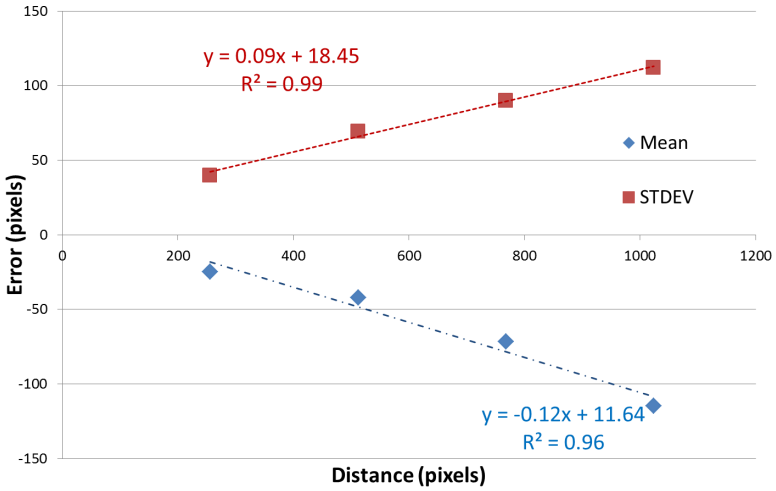
Method

The task and experimental setup is identical to our initial 2D targeting study except for the input device. In this study, input was collected using a Microsoft Sidewinder 3 mouse set to 600 dpi. Cursor acceleration was set to the default level set by the Windows 7 operating system.

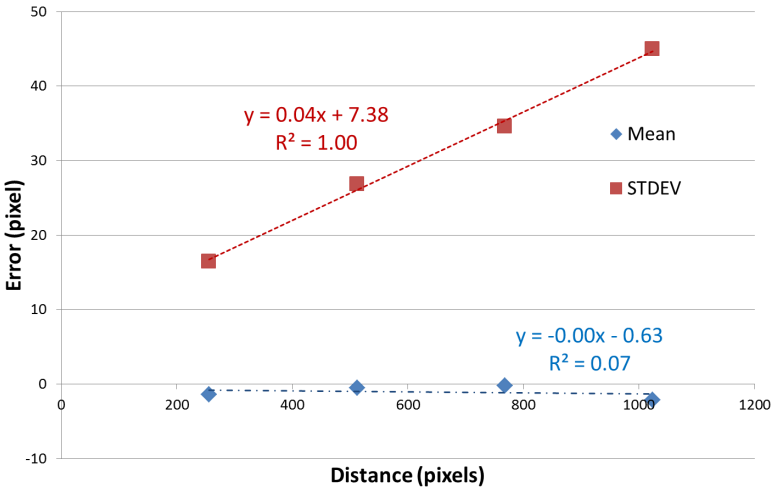
Sixteen participants, ten males and six females participated in the study. All participants were affiliated with the local university and four were left-handed.

Results

Of the 7680 tasks recorded during the study, 7.8% resulted in the user not correctly hitting the target. A single user accounted for 51.5% of all errors recorded. This user was removed



(a) Collinear Pixel Accuracy



(b) Orthogonal Pixel Accuracy

Figure 5.9: Mean and standard deviation (STDEV) for pixel accuracy by distance of the KEP predictor for the single-shot prediction strategy.

from the study resulting in an overall error rate of 4.6% for the remaining 15 participants. A majority of the remaining errors (56.0%) were the result of users having difficulty acquiring the 8 pixel target. As in our previous studies, errors were removed from our analysis.

Continuous Prediction Overall accuracies for predicting the user's intended target for our continuous predictor is shown in Table 5.11. The users' correct target was identified 22.3% of the time. Analysis of variance on percent correct with distance and width as factors shows a significant effect for distance ($F_{3,260} = 155.7$, $p < .001$, $\eta^2 = .27$) and target width ($F_{4,260} = 386.6$, $p < .001$, $\eta^2 = .55$) similar to the previous study without cursor acceleration. A distance*target width interaction was also observed ($F_{12,260} = 61.1$, $p < .001$, $\eta^2 = .18$).

Post-hoc analysis using Bonferroni correction shows a significant difference for all distances ($p < .001$ in all cases) and all target widths with the exception of the two smallest target widths ($p < .001$). Again, these effects are similar to those we observed in the 2D study without cursor acceleration.

A paired t-test shows that the need to determine the user's path does not significantly affect KEP's accuracy ($p > .68$). Given that in both our 2D targeting studies the need to infer motion path did not significantly differ from the perfect prediction condition, we conclude that our method of determining the user's intended path is sufficient.

	Observed		**Perfect Path Prediction**	
	Correct	**Off-by-One**	**Correct**	**Off-by-One**
80% Gesture Length	12.8%	28.2%	16.0%	24.4%
85% Gesture Length	15.3%	33.2%	17.9%	27.5%
90% Gesture Length	22.3%	38.1%	22.9%	27.5%

Table 5.11: The observed target accuracies and target accuracies assuming perfect path prediction for our continuous predictor.

	Target Distance			
Target Width	**256**	**512**	**768**	**1024**
8	3.7% (17.9%)	1.6% (13.4%)	0.7% (2.6%)	0.4% (2.1%)
16	13.7% (54.2%)	3.6% (28.0%)	1.8% (16.5%)	0.4% (10.9%)
32	33.5% (56.9%)	11.7% (67.4%)	5.3% (46.6%)	4.3% (32.6%)
64	69.2% (27.7%)	40.1% (54.7%)	20.8% (70.8%)	9.0% (82.3)
96	88.6% (10.0%)	72.4% (27.2%)	41.0% (56.6%)	18.0% (79.9%)

Table 5.12: Observed correct and off-by-one (in parentheses) target accuracies of our continuous predictor by target width and target distance at 90% gesture length.

Similar to our previous study, analysis of variance of collinear and orthogonal pixel accuracy by target width and distance shows a significant effect for distance (shown in

	Distance	Width	Distance*Width
Collinear	$F_{3,99} = 108.87$, $p < .001$, $\eta^2 = .90$	$F_{4,99} = 6.88, p > .001, \eta^2 = .06$	$F_{12,99} = 2.34, p < .05, \eta^2 = .04$
Orthogonal	$F_{3,99} = 58.2$, $p < .001$, $\eta^2 = .87$	$F_{4,99} = 5.01, ns$	$F_{12,99} = 1.40, ns$

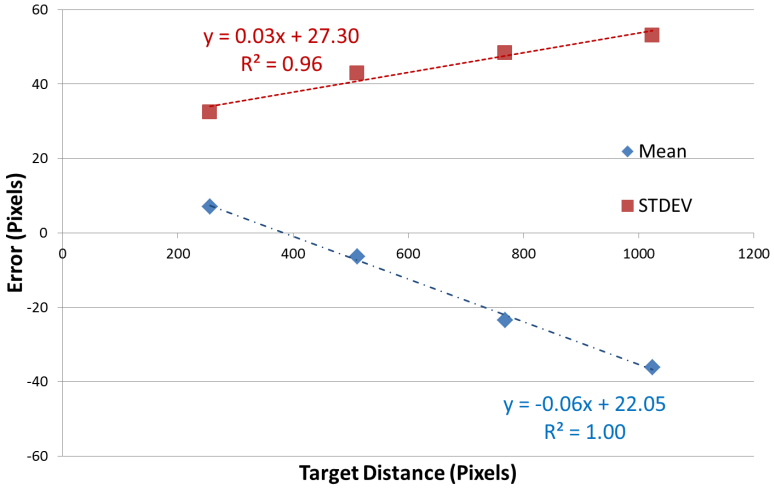
Table 5.13: F and p-values for within-subjects ANOVA on pixel accuracy by study. Shaded cells show no significance ($p > .05$).

Table 5.13). We also observed a significant effect for target width on both collinear and orthogonal accuracy. A target width and distance interaction on collinear accuracy was also observed.

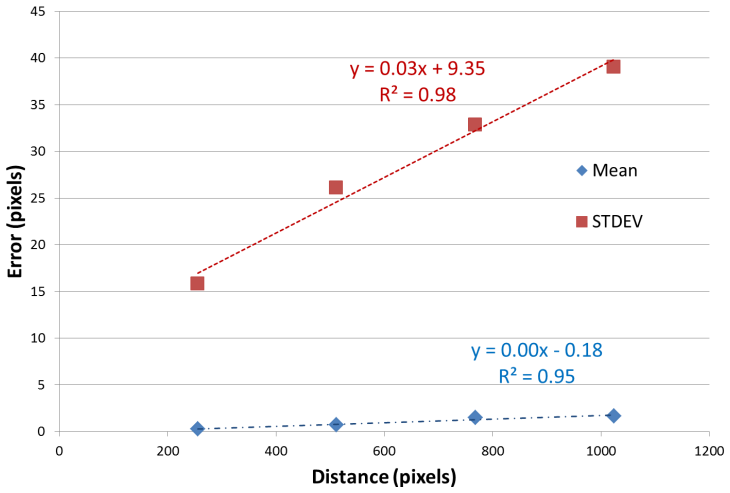
Post-hoc analysis using Bonferroni correction for both collinear and orthogonal error shows a significant difference between all distances ($p < .05$). We also observe a significant difference for collinear pixel accuracy between the 92-pixel target and all other targets ($p < .01$). Post-hoc analysis did not show any significant differences in orthogonal error between target sizes.

As in our previous studies, distance had a significant effect on the distribution of collinear and orthogonal accuracy. Examining the mean and standard deviation for pixel error (Figure 5.10), we again observe strong linear correlations for the means of collinear and orthogonal accuracy by distance ($R^2 > .99$, $p < 0.01$ and $R^2 = 0.95$, $p < 0.01$). We also observe strong linear correlations for the standard deviations for collinear ($R^2 = 0.96$, $p < 0.05$) and orthogonal ($R^2 = 0.98$, $p < 0.01$) error by distance. The strong correlations allow us to conclude that cursor acceleration does not hinder our ability to use the KEP technique to create a probability distribution around a relatively small subset of targets within the interface when using the continuous prediction strategy. Below we examine the effects of cursor acceleration using the single-shot prediction strategy.

Single-Shot Prediction In this chapter we have shown that for one-dimensional targeting tasks, the target accuracy of the KEP predictor decreases as distance increases, and the decrease in target accuracy is due to an increase in the distribution of pixel accuracy (or error). We also demonstrated that there exist a strong correlation between mean and standard deviation, and this releationship can be leveraged to identify likely targets on the display. We have shown that this remains true for two-dimensional targeting without the presence of cursor acceleration. Above, we have shown when using KEP with the continuous prediction strategy, these findings are also observed in the presence of cursor acceleration. Our final goal of this chapter is to examine the single-shot prediction strategy in presence with cursor acceleration. If we observe similar findings to our previous results, we will have shown that regardless of target dimensions (i.e., one or two dimensional)



(a) Collinear Pixel Accuracy



(b) Orthogonal Pixel Accuracy

Figure 5.10: Mean and standard deviation (STDEV) for pixel accuracy by distance of the KEP predictor for the continuous prediction strategy with cursor acceleration activated.

and the presence of cursor acceleration (i.e, with or without cursor acceleration activated) KEP can be used to create a probability distribution around likely targets on the computer display.

Table 5.14 illustrates the target accuracies of the single shot prediction strategy with cursor acceleration activated. Again, the single-shot predictor target accuracy is shown

to be less accurate than when using the continuous prediction strategy, and that higher thresholds result in predictions occurring beyond 90% of gesture length. As in the previous study, we concentrate our analysis in the remaining of this section using a threshold of 85%.

Table 5.15 summarizes the target accuracy for the single shot prediction strategy by distance and target width. Analysis of variance on percent correct with target width and distance as factors shows a significant effect for distance ($F_{3,99} = 132.5$, $p < .001$, $\eta^2 = .14$) and target width ($F_{4,99} = 581.1$, $p < .001$, $\eta^2 = .82$). We also observe a significant distance and target width interaction ($F_{12,99} = 8.69$, $p < .001$, $\eta^2 = .04$).

(a)

	Actual Gesture Length			
	Before 90%		**After 90%**	
Threshold	**Correct**	**Off-by-One**	**Correct**	**Off-by-One**
80%	10.9%	19.8%	0.1%	0.2%
85%	14.2%	26.9%	1.7%	2.3%
90%	9.1%	21.7%	1.9%	1.2%

(b)

	Actual Gesture Length			
	Before 90%		**After 90%**	
Threshold	**Correct**	**Off-by-One**	**Correct**	**Off-by-One**
80%	10.7%	4.0%	0.1%	0.2%
85%	14.4%	25.8%	2.7%	1.9%
90%	11.7%	20.3%	2.2%	1.2%

Table 5.14: The observed target accuracies for our single shot predictor by actual gesture length. (a) Observed target frequencies. (b) Target frequencies given perfect path prediction.

As in our previous analysis of pixel accuracy with 2D targets, we separate pixel accuracy into orthogonal and collinear error. As shown in Table 5.16, analysis of variance of collinear and orthogonal error shows a significant effect for distance on collinear error. As with the one-shot predictor without the presence of cursor acceleration, we did not see any effect for target width on collinear error or any effect for distance or target width for orthogonal error. Post-hoc analysis using Bonferroni correction for collinear error shows a significant difference between all distances ($p < .05$).

Examination of the mean and standard deviation of pixel error show deviations from

	Target Distance			
Target Width	**256**	**512**	**768**	**1024**
8	1.7% (15.0%)	0.5% (6.6%)	2.7% (7.0%)	1.7% (1.7%)
16	5.9% (23.5%)	5.9% (14.6%)	1.6% (10.3%)	2.7% (6.6%)
32	17.6% (47.3%)	13.4% (33.9%)	7.0% (23.7%)	3.8% (16.8%)
64	40.5% (58.4)	31.0% (47.6%)	23.4% (44.7%)	11.2% (31.9)
96	55.4% (44.6%)	40.3% (48.9%)	29.8% (49.5%)	19.8% (49.2%)

Table 5.15: Observed correct and off-by-one (in parentheses) target accuracies of our single-shot predictor by target width and target distance using an 85% threshold.

Pixel Accuracy	**Distance**	**Width**	**Distance*Width**
Collinear	$F_{3,99} = 36.86$, $p < .001$, $\eta^2 = .72$	$F_{4,99} = 0.98, ns$	$F_{12,99} = 0.81, ns$
Orthogonal	$F_{3,99} = 0.54, ns$	$F_{4,99} = 0.73, ns$	$F_{12,99} = 0.57, ns$

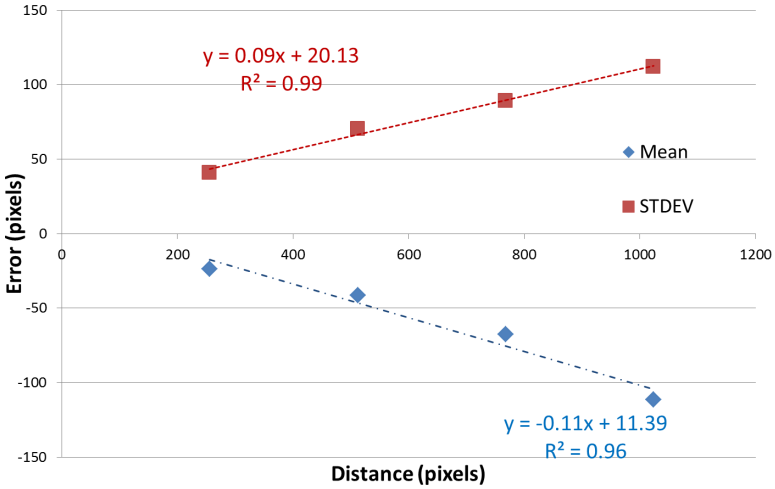
Table 5.16: F and p-values for within-subjects ANOVA on pixel accuracy by error type. Shaded cells show no significance ($p > .05$).

the continuous predictor, but results similar to the one-shot predictor in the previous 2D targeting study. As shown in Figure 5.11, we only observe a correlation for collinear mean ($R^2 = 0.96$, $p < .05$), collinear standard deviation ($R^2 = 0.99$, $p < .01$), and orthogonal standard deviation ($R^2 = 0.99$, $p < .01$). We do not observe a correlation for orthogonal mean. However, as shown in the figure the mean for orthogonal error is quite small (within ± 2). Therefore, similar to the one-shot prediction strategy results without cursor acceleration, the lack of correlation for orthogonal error does not prohibit us from determining the likelihood of a given target on the display in the same manner as with the continuous predictor.

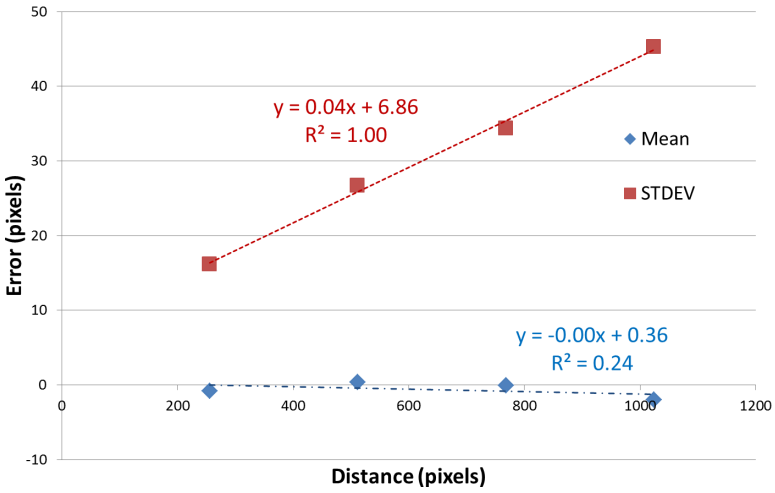
5.3.4 Comparing Prediction Accuracies With and Without Cursor Acceleration

To examine the effects of cursor acceleration on target accuracy, we performed a between subjects ANOVA using the data collected from our third and fourth study. We observed cursor acceleration to have a significant effect on target accuracy ($t(24) = 4.18$, $p < 0.05$) with our continuous predictor. However, the effect was very small ($\eta^2 = .01$). We did not observe an effect for target accuracy for our single-shot predictor.

Given we observed a small effect of cursor acceleration on target accuracy for our continuous predictor, we would expect to see a similar effect on pixel accuracy. However, we



(a) Collinear Pixel Accuracy



(b) Orthogonal Pixel Accuracy

Figure 5.11: Mean and standard deviation (STDEV) for pixel accuracy by distance of the KEP predictor for the single-shot prediction strategy with cursor acceleration activated.

do not observe any effect of cursor acceleration on pixel accuracy for either the continuous or single-shot predictor (see Table 5.17 for results from the paired t-test) . As shown in Figure 5.12, the distributions of pixel accuracy by distance are nearly identical regardless if cursor acceleration was activated.

Pixel Accuracy	Continuous Prediction	Single-shot Prediction
Collinear	$t(24) = 0.53, ns$	$t(24) = 0.23, ns$
Orthogonal	$t(24) = -1.26, ns$	$t(24) = 0.46, ns$

Table 5.17: T-test values comparing pixel accuracy for the continuous and single-shot predictors with and without cursor acceleration.

5.3.5 Summary

In this section we presented two user studies that examined the prediction accuracy of the KEP technique for 2D targets. The first experiment used a 1:1 control to display (CD) ratio allowing us to keep internal validity with our one-dimensional studies and other Fitts-style pointing studies. In the second experiment, we examined the accuracy of KEP in the presence of cursor acceleration, thus, providing external validity of the performance of KEP in modern desktop systems.

In both our experiments we found that distance has a significant effect on KEP's ability to identify a target. Through the examination of pixel error, we found that the decrease in target accuracy was the result of an increase in the distribution of pixel accuracy as distance increased, and this error is mainly concentrated in the path of motion (i.e., collinear error). We then showed that the relationship between pixel error and distance can be leveraged to create a probability distribution around likely targets similar to our one-dimensional targeting study.

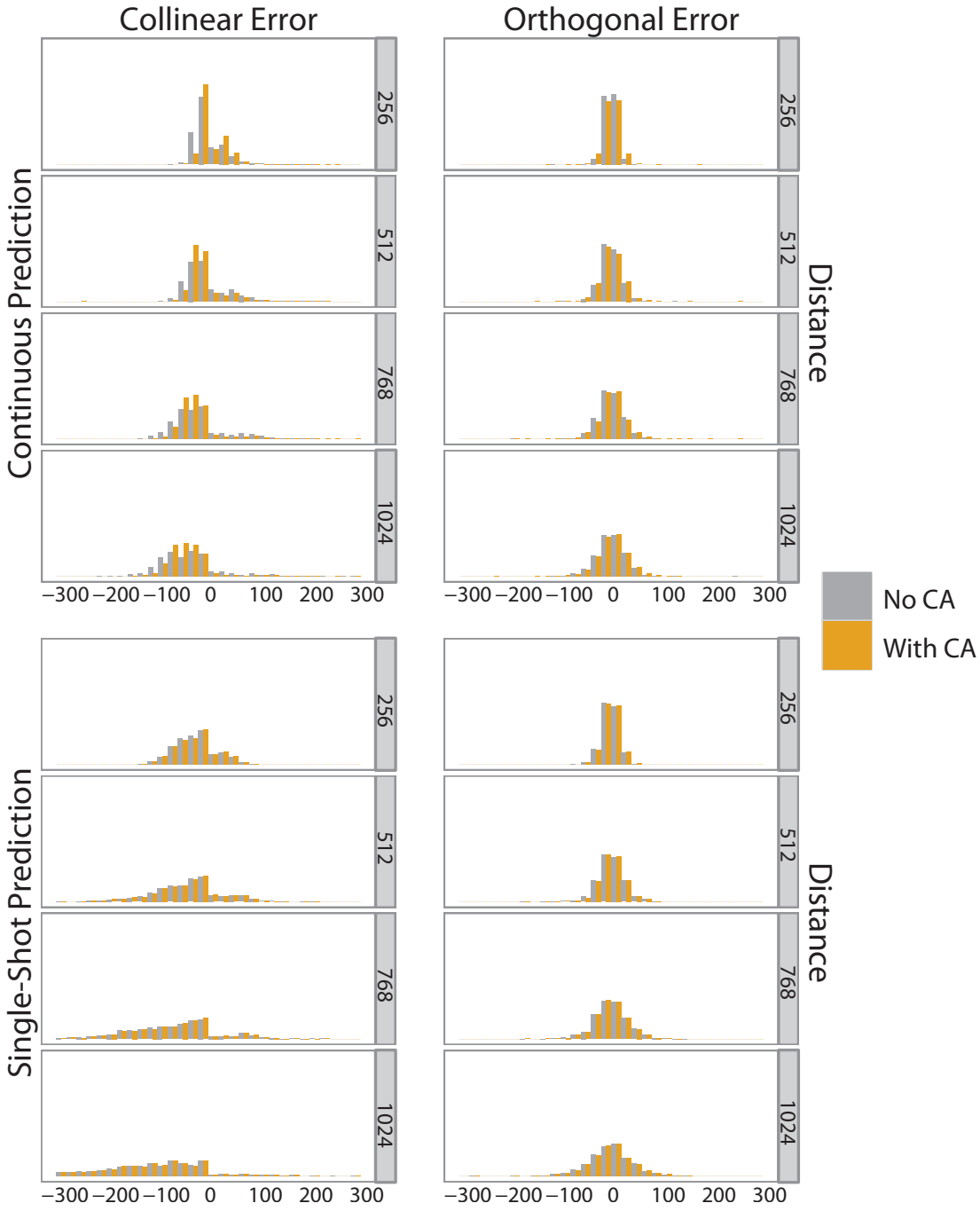


Figure 5.12: Distributions comparing pixel accuracy of collinear(left) and orthogonal error (right) for the continuous (top) and single-shot (bottom) predictors by whether or not cursor acceleration (CA) was activated. (Colour required.)

5.4 Re-examining Target Accuracy

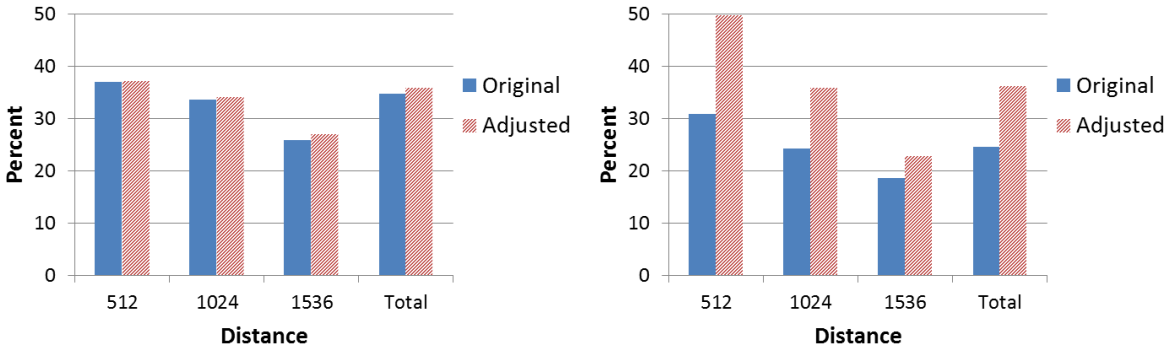
While the discussion has mainly focused on how the relationship between distance and target accuracy can be leveraged to identify likely targets on the computer display, this relationship can also be used to correct the initial target prediction provided by KEP. The process is the same used to center the pixel accuracy distributions. Using the linear equation generated from the correlation of the mean of the pixel accuracies and distance, we calculate an offset to subtract from the original predicted distance to obtain the new adjusted prediction. This new prediction is then used to identify the user's intended target.

To demonstrate the benefits of readjusting initial predictions, we apply the calculated corrections to the predictions of the three user studies presented in this chapter. As shown in Figure 5.13, target accuracy is improved for both the continuous and single-shot prediction strategies. While there was little improvement for the continuous prediction strategy in our 1D study (Figure 5.13(c)), the single-shot prediction strategy target accuracies showed significant improvement.

5.5 Summary

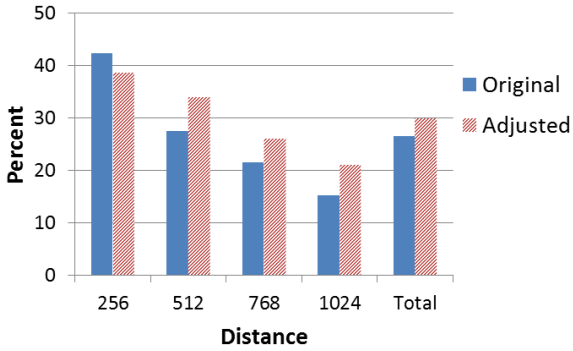
In this chapter we explored the effects of target width and target distance on prediction accuracy for one and two-dimensional targets. Results from the three user studies presented in this chapter demonstrates that there exists a strong linear correlation between accuracy and target distance. Leveraging this relationship, we demonstrated that the KEP predictor can be used to identify the likelihood of any target or pixel on the display being the user's intended target. In addition, this relationship can also be used to improve target accuracy by providing a corrective measure to initial predictions.

In the next chapter we explore the implications of these results (and the results from Chapter 3) to research on motor control and endpoint predictors which aim at predicting endpoint by modeling motion characteristics.

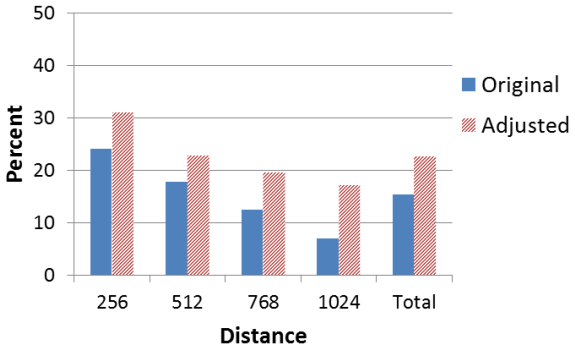


(a) 1D, Continuous Prediction

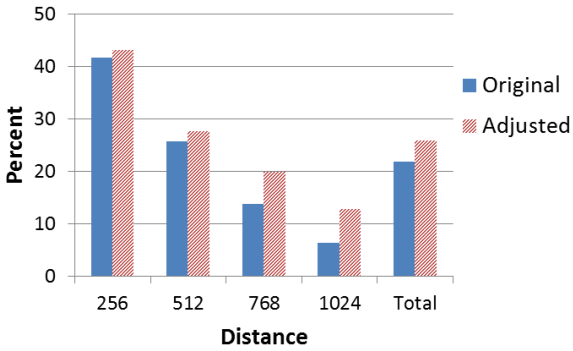
(b) 1D, Single-shot Prediction



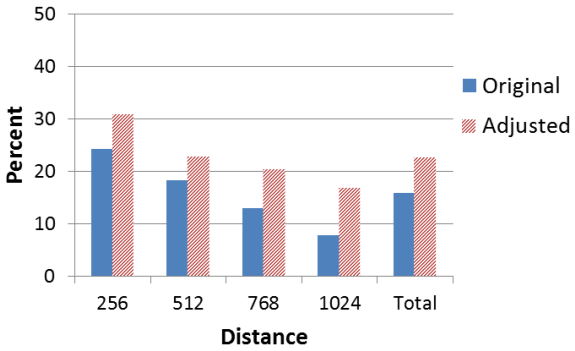
(c) 2D without CA, Continuous Prediction



(d) 2D without CA, Single-shot Prediction



(e) 2D with CA, Continuous Prediction



(f) 2D with CA, Single-shot Prediction

Figure 5.13: Accuracies of identifying the user's intended target before (in blue) and after correction is applied (in red) for each of the three studies presented in this chapter. Target accuracies for the continuous prediction strategy are shown on the left and the single-shot prediction strategy on the right. Accuracies for the one-dimensional (1D) targeting study are shown in (a) and (b), the two-dimensional study (2D) without the presence of cursor acceleration (CA) in (c) and (d), and (e) and (f) illustrate the accuracies for the 2D study with CA.

Chapter 6

Implications to Motor Control and Endpoint Prediction

In this dissertation, we have made several contributions to endpoint prediction, and the understanding of the performance of the kinematic endpoint prediction (KEP) predictor under a wide range of conditions. In Chapter 3, we demonstrated that, for amplitude and directional constrained targets, the most discriminating information in the motion kinematics of the gesture is encoded in the motion characteristics perpendicular to the direction of the target, and the differences in orthogonal characteristics appears to be mainly concentrated in the initial 70% of gesture motion. Later in the chapter, we explored if the intended use of a target effects the kinimatic profile of the motion and demonstrated that any temporal changes can be attributed to the user spending time, motionionless, over the primary target. Based on the information presented in Chapter 3, we concluded that in order to model the kinematics of the intial 90% of motion we should focus on modeling ballistic movement.

In Chapter 4, we described a taxonomy for endpoint predictors and present the KEP predictor. We showed that the KEP predictor can accurately identify a user's intended target for both stylus and mouse-based motion. This was followed by a chapter that thoroughly evaluated the predictor. We begun Chapter 5 by describing the performance of the KEP predictor for a wide range of target widths and target distances for one-dimensional (1D) targets. This was followed by an evaluation of the predictor for two-dimensional (2D) targets with and without the presence of cursor acceleration. The results from our evaluation can be summarized as follows:

- The requirement to predict gesture path for 2D targets does not affect KEP's

prediction accuracy.

- The presence of cursor acceleration does not affect KEP's prediction accuracy for 2D targets.
- Target distance has a significant affect on KEP predictor accuracy. As target distance increases the predictor accuracy of KEP decreases.
- There exist a strong linear correlation between the mean and standard deviation of pixel accuracy and target distance.
- The link between pixel accuracy mean and target distance can be leveraged to adjust the initial predictions which results in higher target accuracy rates.
- The correlation between pixel accuracy and target distance can be leveraged to create a probability of any pixel on the display being the likely target of a pointing gesture.

In this chapter, we explore the relationship between distance and prediction accuracy in greater depth. More specifically, we are interested in understanding why we are observing the relationship between target distance and accuracy and its implications to research in endpoint prediction and motor control. Although we have shown that the relationship can be leveraged to identify targets, several questions remain including:

- Why does KEP often underestimate target location, i.e. why is the mean of pixel accuracy regularly negative? We have demonstrated that a correction could be applied to adjust the mean back to 0 (which is best). However, can KEP be adjusted so that correction is not needed? In other words, is the negative pixel mean caused by a limitation in KEP?
- Similarly, is the linear increase in the standard deviation of pixel error as distance increases an artifact of the prediction technique or the result of the ballistic component of motion? If we find that the increased distribution of error is a residual of the KEP technique then it may be possible to a create better predictor using kinematics. However, if we find that the linear relationship is the byproduct of the motion, the results presented in the previous chapter will have broader implications to endpoint prediction in user interfaces.
- Why is the distribution of collinear pixel error greater than orthogonal error?

By the end of this chapter, we will have shown that the negative correlation of mean can be attributed to the actual endpoint of the user's motion, i.e., that the user is actually hitting the front of the target at a higher rate than at the center or back of the target. We will also demonstrate that the increase in the standard deviation and magnitude of the collinear pixel error as target distance is increased can be explained by characteristics of the human motor system.

This chapter is organized as follows. First, we re-examine the means of the pixel accuracy of KEP and gesture endpoint to determine the cause of the KEP predictor underestimating gesture endpoint. This is followed by a discussion of prior work in the motor-control literature on aiming in order to gain an understanding of the observed linear relationship between the standard deviation of pixel accuracy and target distance. We conclude the chapter with a discussion of the implications of our results to endpoint prediction and motor control research.

6.1 KEP and Mean Pixel Accuracy

During our user studies, the mean of KEP's pixel accuracy continuously underestimated gesture endpoint (i.e., before target center) for both the continuous and single-shot prediction strategies. There are a couple of explanations for this behaviour: either there is a flaw in the KEP predictor, there is something in the user's behaviour that triggers the KEP predictor to underestimate target distance, or both a flaw in the KEP technique and something in user's behaviour. In this section, we take a deeper look at the results presented in Chapter 5 in order to determine the cause of KEP underestimating gesture distance.

6.1.1 Distance and Actual Gesture Endpoint

To examine the effects of distance on actual user endpoint (i.e., the location where the user clicks on the target), we calculated *user error* for each of the user studies presented in Chapter 5. We define user error as the distance from the center of the target to the location where the user terminates the pointing gesture by depressing the mouse button. Figure 6.1 illustrates the distribution of user error by distance for the user study using 2D targets without cursor acceleration (Chapter 5.3.2). As illustrated by the figure, at all target distances the distribution of user error is shifted slightly towards the negative direction, meaning that the users tended to click the front of the target rather than the back (in

relation to motion direction). In addition, as distance increased the distributions become more skewed towards the front of the target. This suggests that as distance increases, users are more likely aiming at the front of the target than the center of the target. Similar observations were also observed in our other studies (i.e., in our one-dimensional targeting task and the two-dimensional targeting task with cursor acceleration).

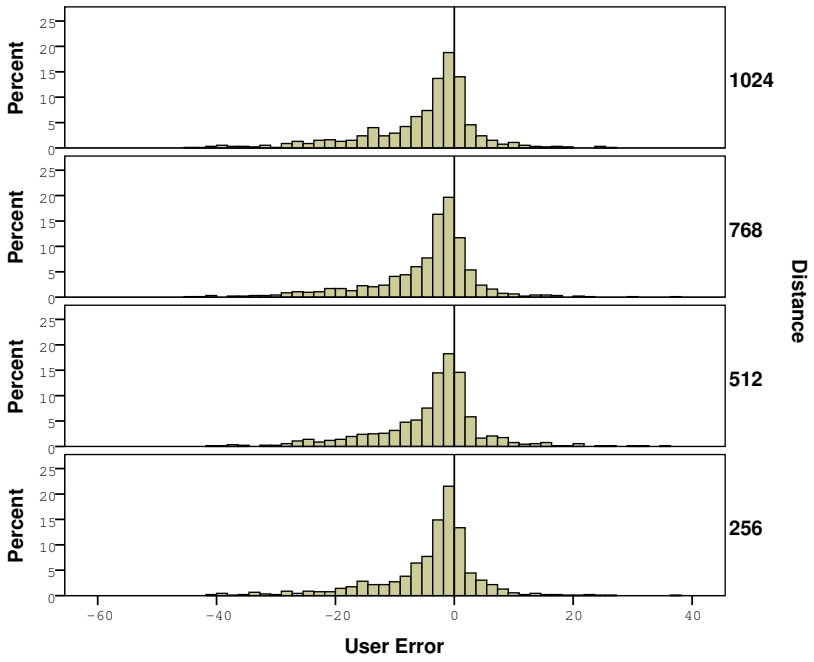


Figure 6.1: Distribution of *user error* by target distance for our two-dimensional targeting task without cursor acceleration.

Figure 6.2 illustrates *user error* by KEP predictor pixel error for the continuous predictor in the one-dimensional target study by target distance. As shown in the figure, we observe very slight correlations between pixel error and user error ($0.37 < R^2 > 0.56$), with pixel error decreasing (i.e., underestimating gesture distance) as the user clicks closer to the front of the target. We also observed weak correlations for the single shot predictor and in our other studies ($0.10 < R^2 > 0.60$ in all cases). However, given that the correlation between user error and pixel error is weak due to the low R^2 values, we can not conclude that the negative mean of pixel error observed in our studies is the result of the user clicking on the front of the target. Although, it does suggest that user error may have a slight influence on pixel error.

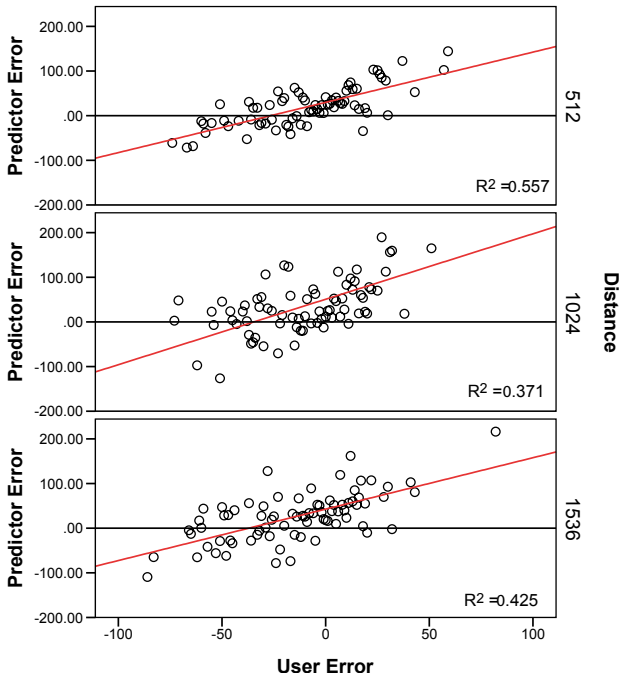


Figure 6.2: User error by KEP pixel accuracy for our one-dimensional study using the continuous prediction strategy.

6.1.2 Distance and Corrective Submovements

Another possible explanation for why the KEP predictor has a tendency to underestimate motion endpoint is that the user's initial ballistic movement requires the user to perform several submovements to reach the target. In other words, the initial ballistic movement of the gesture is falling short of the target, thus requiring the user to perform several submovements to reach the target. Given that the KEP predictor models this initial ballistic movement, if the user's initial movement is short the predicted distance will also fall short of the target.

In Chapter 3, we explored the effects of distance on the time required to complete the first 90% of gesture distance and demonstrated that the last 10% of motion distance consumes as much as 60% of the total movement time. In addition, as shown in Figure 6.3, as distance increases the time spent traversing the last 10% of gesture distance consumes a higher percentage of the total movement time. Based on these observations, the HCI community infers that the user spends a higher percentage of the gesture performing corrective submovements.

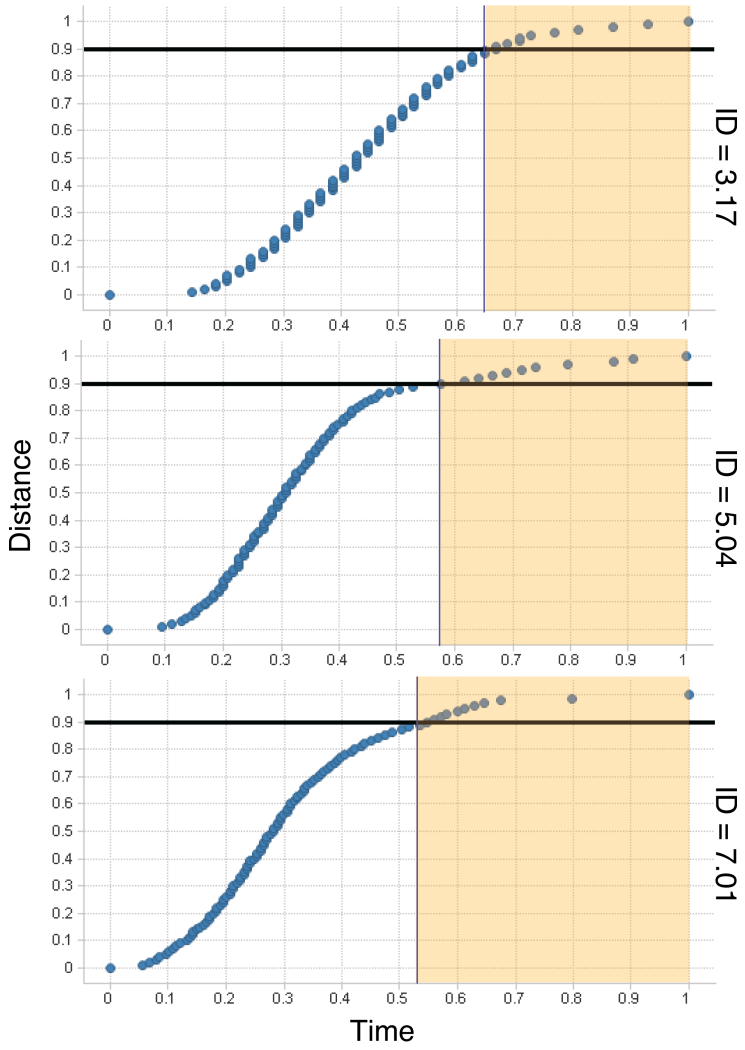


Figure 6.3: Examples of distance vs. time (both normalized) plots for three IDs. The dark horizontal line represents 90% of gesture distance. The shaded region represents the time taken to complete the last 10% of distance.

To test the hypothesis that the observed underestimation of gesture movement predicted by the KEP predictor is the result of a user's initial ballistic movement falling short of the target, we analyze each of the movement gestures collected during our user studies by using the following steps:

1. We normalize gesture length such that the distance is between 0 and 1 by dividing each collected point in time by total gesture distance.

2. Using the resulting normalized gesture from Step 1, we sample the current speed at equal intervals (each 1% of total gesture length) along the gesture.

3. Using the resulting gestures from Step 2, we categorize the movements into one of three groups based on the *normalized pixel error*. Normalized pixel error is calculated by dividing the pixel error by target distance. The first group has a normalized pixel error between -0.1 and 0.1. These values were chosen because they represent approximately ± 1 standard deviation of the observed pixel accuracies from our user studies in the previous chapter. The second group has normalized pixel error less than $-.1$ which represents movements where the KEP predictor underestimated movement distance. The last group represents when KEP overestimated movement distance and has normalized pixel error greater than 0.1.

4. For each user, we average each point along the gesture calculated from Step 2 for every target distance and pixel error group. Finally, we calculate the average speed at each point for each of the studies presented in this dissertation.

The resulting normalized averaged kinematic curves for our 2D user study without cursor acceleration are shown in Figure 6.4. If we examine the average speed for the last 20% of gesture distance before the KEP predictor makes a prediction (recall that prediction is made using the first 85% of movement distance), we observe that the average speed is lower when we underestimate (in red) and higher when we overestimate (in blue) gesture distance. In addition, when we underestimate we also tend to see a more pronounced subcorrection as indicated by the small plateau and/or increase of speed during the final 10% of the gesture.

The normalized averaged kinematic curves also help explain why the mean pixel accuracy for our single-shot prediction strategy was often greater than for the continuous prediction strategy. When predicting endpoint with the single-shot prediction strategy, the initial ballistic movement of the underestimated group (i.e., error $< -.1$) will result in KEP predicting endpoint earlier in motion due to the shape of the curve and while the

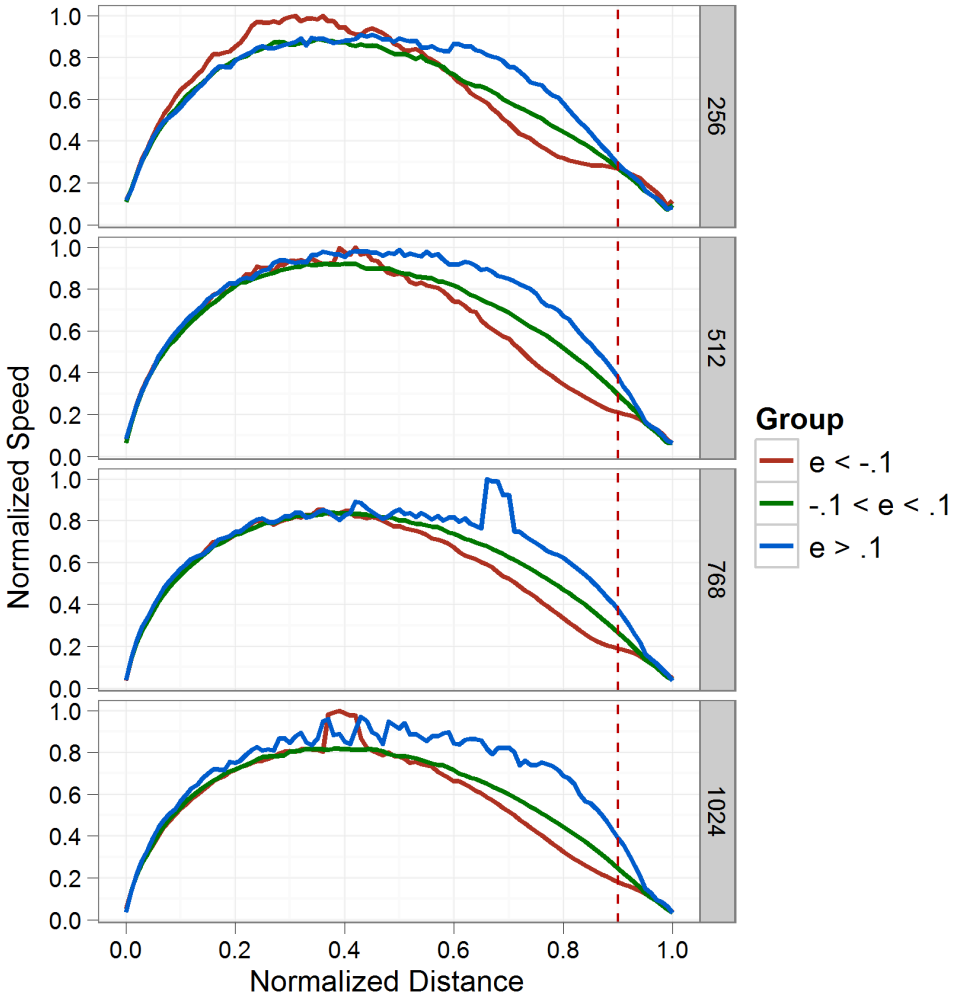


Figure 6.4: Normalized and averaged kinematic curves from the 2D user study without cursor acceleration by pixel accuracy group.

submovements may be observed before 90% of motion, given we only predict once during the gesture we are unable to take advantage of the submovement to adjust our prediction. However, in the continous prediction strategy, we continue to make predictions and the corrective movments (represented by the slower deaceleration) will influence the prediction returned by the KEP predictor (although only slightly).

6.1.3 Conclusions

In this section, we presented several hypothesis on why we continuously observed a negative mean for pixel error for the KEP predictor. From our analysis, it appears that the cause of the KEP technique underestimating gesture distance is a combination of user behaviour and KEP error. The tendency for a user to target the front of the target and a user's initial ballistic movement seem to affect the prediction accuracy of the KEP predictor. When a user's initial ballistic movement requires the user to perform additional submovements (as seen in Figure 6.4) the KEP does not fully utilize submovements that occur before 90% of gesture distance to adjust its prediction (in the case of the single-shot predictor, KEP does not refine the prediction at all). Given the user's behaviour has a role in KEP underestimating motion distance, our best solution to counter the underestimaing of motion distance is to use calibration methods simlar to the one we performed in Chapter 5.4 where we adjust endpoint based on the linear relationship between pixel error mean and distance.

6.2 KEP and Pixel Accuracy Distribution

Results from the user trials presented in this dissertation have demonstrated that the accuracy of the KEP technique to predict motion endpoint is affected by target distance. Examination of the distribution of the predictions compared to the user's intended target shows that as distance increases, the distribution of pixel accuracy (error) also increases. However, our results show that the distribution is mainly focused in the direction of motion (i.e., collinear to the movement opposed to orthogonal) for two-dimensional targets.

In this section, we examine possible explanations for this phenomenon. Using research from motor control literature, we will demonstrate that the observed distributions for pixel accuracy of the KEP predictor can be explained by users' inherent accuracy of planar reaching movements. We begin by summarizing the work of Gordon et al. [21] that examines the accuracy of reaching movements. We then show how our observations of KEP prediction accuracy can be described using these findings.

6.2.1 Variability in Movement Endpoints

Our analysis of KEP prediction variability is influenced by the findings of Gordon et al. [21] who were trying to determine the nature and origin of the coordinate system in which

reaching (pointing) movements were planned. The researchers performed an experiment where six participants used a digitizing tablet, similar to the one used in our studies, to point to targets on a computer display. To begin, participants were instructed to move the cursor to a starting location. After a brief amount of time, the participant was indicated to move to a target using a "single, quick, and uncorrected movement" [21, pg. 99]. To prevent the participant from making any corrective movements, the cursor was blanked at the start of the trial, thus, not providing the participant with any visual feedback. At the end of the pointing movement, the actual cursor trajectory was displayed on the screen and participants were given a score based on how close they came to the target.

The targets used in the study were arranged in a similar manner to our modified ISO 9421-9 [28] targets shown in Figure 5.6 except that Gordon et al.'s participants always began in the center of the display. Target distance was varied between two distances (3.2cm and 9.6cm) and target size increased as distance increased.

Results from their user study showed that velocity profiles are single-peaked and bell-shaped with a single acceleration and deceleration phase, and that peak velocity increases with distance, thus, providing more support for the minimum jerk principle [27]. They also note that movement paths were essentially straight with little to no curvature. Therefore, supporting our design rational used for path prediction when predicting motion endpoint with 2D targets. Examination of the variability in endpoints (shown in Figure 6.5) resulted in two main findings. First, that the contour of the distributions were oriented along the direction of the movement, demonstrating that the variability in distance was greater than movement direction. Second, as distance increased so did the variability of endpoints (shown by the larger oval shape in Figure 6.5). Analysis of variance of endpoint variation by distance, target size, and target angle using ANOVA, showed target size and angle had no significant effect on endpoint variation. However, target distance was shown to affect endpoint variation.

6.2.2 Variability of Endpoint and KEP Pixel Accuracy

Gordon et al.'s [21] work presented above demonstrates that the variability of endpoints of a pointing motion without corrective submovements mainly occurs in the direction of the movement and this variability increases as distance increases. This is very similar to the variability of pixel accuracy of the KEP predictor. Recall in our two-dimensional (2D) user studies (Chapter 5.3) where we observe a linear correlation between collinear pixel accuracy and distance, and constant (nearly zero) variability for orthogonal pixel accuracy.

The observations of Gordon et al. and the observed pixel accuracy of our KEP predic-

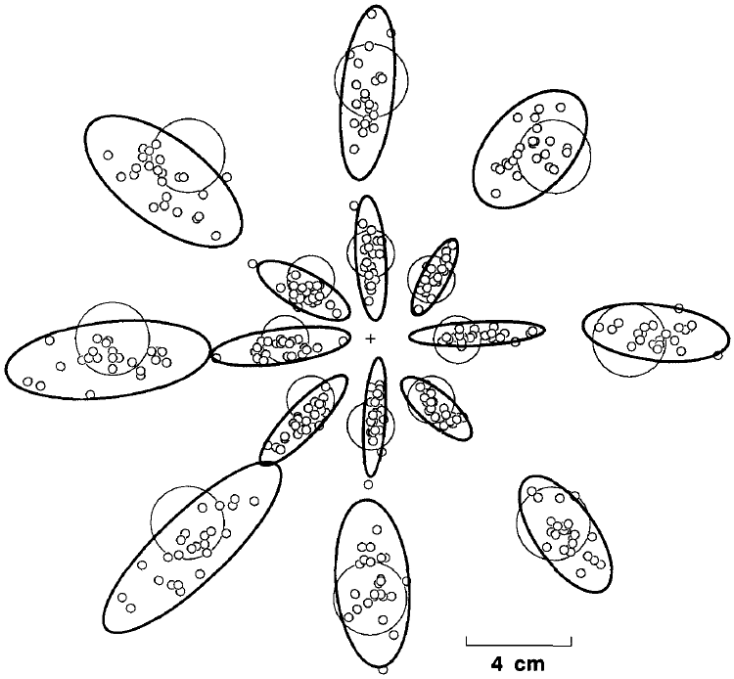


Figure 6.5: End-point distributions for movements to different targets in eight directions and two distances as collected by Gordon et al. [21]. End points for individual movements are represented by small circles; larger circles show target locations. The distributions of end points for movements to each target are fitted with surrounding ovals within which 90% of the population of endpoints should fall. Replicated from [21].

tors has several implications to predicting motion endpoint using motion characteristics. First, pointing motion is relatively straight, thus, supporting our results that the need to predict motion path is not a major factor in prediction accuracy. Second, the velocity profile of a ballistic pointing motion has a single peak, is bell-shaped, and can be described by the minimum jerk principle. Therefore, the approach we take to model this motion using the minimum jerk principle appears to be appropriate. In addition, this suggests that techniques that plan on using motion characteristics to predict endpoint will need to take a similar approach to that employed by KEP and presented in this dissertation.

Finally, given there exists a variability of endpoints in non-corrective movements and that this variability increases as distance increases, any technique that models this initial movement will also show variation. Therefore, we would expect to see variation of predictions and, in fact, we do. More importantly, we demonstrate how to calculate the variation as a function of distance using the observed linear correlation between collinear

pixel accuracy and distance. This linear correlation is something not observed by Gordon et al. probably due to their limited target distances and subject population (only six participants with two being co-authors). In addition, our results strengthen the findings of Gordon et al. by demonstrating that the initial pointing motion continues to be ballistic and bell-shaped even when the participant is able to perform corrective submovements given the visual feedback provided by the system. However, we must be cautious to say Gordon et al.'s participants performed no submovements given that subjects were scored on how close they got to a target and may have adjusted their deceleration phase to get within the target region more often than a pure ballistic movement (similar to what we observed and illustrated in Figure 6.4). Regardless, our results also support current theories of goal-directed motion (i.e., Meyer et al's [46] stochastic optimized submovement model) that the initial phase of motion is ballistic in nature followed by corrective submovements to acquire the target. The fact that we observe a linear relationship between the modeling of a user's ballistic movement (i.e., KEP prediction accuracy) and distance supports that users are trading speed for accuracy. In Gordon et al.'s study, participants were told to be accurate and not to correct their initial motion. In typical pointing tasks, such as pointing on targets on a display, subjects are trying to be both accurate and fast given they can correct their initial motion. The farther a user needs to move a cursor on the display, the more inaccurate their initial movement becomes in order to traverse the further distance. Instead of relying on this initial movement to acquire the target, the user relies on more corrective submovements. These observations not only explain the increase in standard deviation of pixel accuracy for the KEP predictor, but also the increase of time spent covering the last 10% of motion distance (Figure 6.3).

Our results provide additional evidence supporting the minimum jerk principle and the stochastic optimized submovement principle. Therefore, we would expect any revised endpoint prediction techniques that use motion characteristics to be strongly based on the techniques presented in this dissertation. More importantly, given the variability of ballistic motion and the strong correlation between KEP prediction accuracy and target distance it is unlikely to see significant improvements in endpoint prediction using ballistic motion. Instead, any revisions of the KEP technique must be able to take into account a user's corrective submovement to correct the initial prediction. However, as shown in this dissertation, these movements often happen too late in the movement to support the current pointing facilitation techniques proposed in the HCI literature. In addition, as demonstrated in this chapter, incorporating submovements becomes even more challenging when the system needs to support a single-shot prediction strategy regardless of when the submovements occur.

6.3 Summary

In this chapter we explored the relationship between distance and prediction accuracy in greater depth. We showed that KEP's underestimation of gesture distance is a combination of the user's tendency to target the front of the target and the KEP technique not utilizing submovement when the user's initial ballistic movement is off course. More importantly, we showed that the correlation of prediction accuracy and target distance can be explained by the deviation of the initial ballistic motion as observed in the motor control literature. Therefore, we conclude that the KEP technique provides the optimal accuracy when prediction of motion endpoint is performed using only the ballistic components of motion and before 90% of motion distance.

Chapter 7

Applications of Kinematic Endpoint Prediction

In this chapter, we explore using the kinematic endpoint predictor (KEP) to enable expanding widgets [44]. We begin by performing a user study to characterize the accuracy necessary to improve pointing for specific display configurations that are currently resistant to standard pointing facilitation techniques. More specifically, we gain an understanding of the cost associated with using expanding widgets and the KEP technique when a target other than the user's intended target is expanded. Results from our user study demonstrates that users are able to take advantage of an enlarged target if, and only if, the target shifts less than a given amount. Later in the chapter, we use the probabilistic model presented in Chapter 5 and develop a proof-of-concept virtual keyboard that demonstrates how kinematic endpoint prediction can be used in modern graphical interfaces to facilitate pointing.

7.1 Understanding the Effects of Target Expansion and Misprediction

7.1.1 Introduction

In this section, we explore the performance of one pointing facilitation technique, target expansion, for tiled targets. Target expansion works by expanding the target a user is pointing at on the display so the target is easier to acquire. However, as noted by McGuffin

and Balakrishnan [44] and Zhai et al. [44], when targets are densely arranged on the screen, expanding all targets in a user's path results in no pointing advantage. With dense or tiled target arrangements, performance gains are only possible if one could reasonably predict the trajectory of the cursor such that the system can identify, in real time, the target a user is going to select [44]. To support target expansion for tiled targets, we examine if our endpoint predictor can be used to identify candidate targets to expand.

In Chapter 5, we demonstrated that the kinematic endpoint predictor (KEP) can be used to identify likely targets on the display and that the region predicted by KEP is defined by a normal probability distribution. In addition, the normal distribution is centered on the maximally likely target, and targets surrounding that region have decreasing likelihoods. As a result, while the predicted target may be the actual intended target, surrounding targets are also likely. Given that the strength of KEP is to identify likely targets over a region and not a single onscreen target, we explore two issues associated with pointing in tiled target arrangements. First, is it possible to expand a small group of targets rather than an individual target to boost predictor accuracy? Second, given expansion of a set of targets, does the use of the KEP improve pointing performance?

It may be the case that the present KEP accuracy is sufficient to observe an improvement in performance in pointing tasks when expanding multiple targets. For example, if the KEP correctly predicts the user's target, the user's intended target will be at the center of the expansion region, and targeting will likely be faster. However, because the KEP predicts a region, offset errors are common, and, with an offset error, the expansion region will be centered on a target other than the user's desired target. If the offset is sufficiently small, the user's desired target will be expanded, but the target expansion will be confounded with target displacement – targets closer to the center of the expansion region will push the user's desired target either toward or away from the user. If the algorithm's prediction is off by a distance greater than the expansion region, a large offset error, the user's desired target will not be expanded *and the target will be shifted.* There is a probable benefit if the KEP algorithm is correct and expands a set of targets centered on the user's desired target. It is highly likely that large offset errors (where the user's desired target is shifted and not expanded) will increase the cost of a pointing task. The effect of small offset errors, where a target is enlarged and shifted on the display, is unknown. In this section, we seek to determine whether KEP accuracies result in an *overall* improvement in pointing performance. We also wish to examine the costs of small and large-offset errors.

Through results from two experiments presented in this section, we show that it is possible to expand a small set of targets on the computer screen to improve pointing performance. We also show that, when expanding a region, the benefits of expansion are

affected by the target shift, i.e. the size of the offset error. We find a limit on shift of about 80 pixels on our 24 inch 1920x1200 displays. Finally, we demonstrate that, within the expansion region limit, any endpoint predictor must have an accuracy greater than 56.5% to realize a net benefit from expanding targets.

7.1.2 Target Expansion for Tiled Targets

As mentioned at the beginning of this section, enlarging all targets in the user's path will result in no performance gain. Therefore, to enable target expansion in tiled target arrangements, a predictor is needed to select a candidate target for expansion. Given that KEP identifies a region of interest on the display and the region is typically larger then that of the user's intended target, the predicted target may not be the user's intended target.

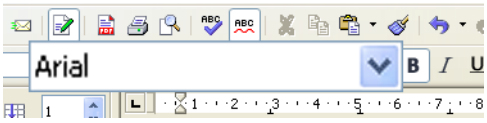
To overcome the frequency of offset errors in endpoint prediction and increase the likelihood that the user's intended target is enlarged, we propose expanding a candidate set of targets. Expanding a group of targets will obviously improve the probability that any individual candidate target will be included in the expanded set. However, expansion of a group also results in having a greater effect on neighboring targets. In this section we describe our design decisions for expanding a group of targets.

Due to the limited amount of screen space around tiled targets, expansion of a target will have an effect on neighboring targets. There are two possible effects caused by the expansion of a target in tiled targets: occlusion and displacement.

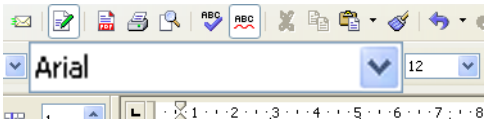
Occlusion has been suggested as a technique to avoid excessive sideways shifts of targets at the cost of interfering with the visibility of neighboring targets [44]. Previous designs using occlusion for tiled targets have been limited to targets of equal size. Therefore, doubling the height and width of one target results in a 50% occlusion of the adjacent targets. However, if target sizes differ, up to 100% occlusion of neighboring targets can occur. For example, a majority of word processing applications' toolbars include widgets for font and point size selection similar to the one shown in Figure 7.1(a). As shown in Figure 7.1(b), expansion of the font selector widget would result in the complete occlusion of the point size selection widget. Expanding multiple targets exacerbates the occlusion problem. If three targets are expanded to double their original width and height, then 1.5 targets of equal size are occluded on either side.

Due to both the heterogeneous size of widgets found in graphical user interfaces and our desire to expand multiple candidate targets, both of which would result in the total occlusion of possible targets, we choose to examine the use of displacement when expanding

(a)



(b)



(c)

Figure 7.1: Font and point size selection widgets common in word processing programs. (a) The original unexpanded widget. (b) Expansion of the font widget using occlusion resulting in 100% occlusion of the point size widget. (c) Expansion of the font widget using displacement.

multiple targets in tiled arrangements. Displacement occurs when an expanding target causes neighboring targets to shift in order to make room for the target's new size. However, displacement is also not without limitations. Expanding a target that is not the user's intended target, what we call an offset error, results in the user needing to acquire a target that has shifted position. In the example of the font and point size widgets, expansion of the font selection widget results in the point size widget being displaced 100% of its width (see Figure 7.1(c)).

Expanding multiple targets will cause an even larger co-linear displacement compared to expanding a single target, because the group of targets consume more space than a single target. Finally, visual disruption of the display is caused by expanding multiple targets, and this visual disruption may negate the performance benefits of expanding the target size.

Because of these potential risks, we describe two studies of target expansion with endpoint prediction. The first uses a simulated predictor with high accuracy, while the second uses the real-time KEP algorithm.

7.1.3 Expansion With Simulated Endpoint Prediction

The goal of our first experiment is to determine whether or not it is possible to expand a candidate set of targets, and whether any amount of target displacement is possible. To do this, we use a simulated predictor based on the accuracies reported in for stylus-

based motion. Recall that for stylus-based motion, KEP was able to correctly identify the user's intended target 42% of the time and an adjacent target an addition 39% of the time. Our goal was to determine whether simulating these accuracies would result in a measurable performance improvement in pointing tasks. We also wished to determine how great the cost of displacement is with small offset errors (off-by-one) where the user's target is enlarged and shifted slightly, and large offset errors (off-by-two) where the user's target is not enlarged and is shifted.

In this section, we describe an experiment that shows that, if we expand a candidate set of three targets - the predicted target ± one target - in a tiled arrangement, it aids target acquisition based on the stylus-based probabilities presented in Chapter 4.3.2. As aspects of this experimental design are replicated in our second study, we spend some time on the specific details of this study.

Method

Apparatus The experiment was conducted on a generic desktop computer (P4, 2.0GHz) with a 24-inch 1920x1200 LCD display running custom software written in C# using Microsoft .NET. Input was collected using a Wacom Intuos3 five button mouse on a 12x19 inch tablet with cursor acceleration turned off.

Task To test the performance of expanding a set of targets,we devised an experimental task that mimics that of previous work on target expansion [44, 74, 7]. Our task differs only in its use of seven tiled targets instead of an isolated target and in that target expansion occurs over more than one target.

The task for our experiments was a simple one-dimensional pointing task. Initially a green starting rectangle was randomly displayed close to one of the horizontal boundaries of the screen. The task began when the participant used the cursor to click within the starting location. At that time, seven tiled targets would appear on the display orthogonal to the starting position. Participants were required to move the cursor to the red target and use the mouse button to click on the target. A successful target acquisition (i.e., clicking within the target region) was indicated by the target changing color. Users were told to acquire the target as quickly and accurately as possible, similar to other Fitts' Law tasks. To prevent users from always targeting the center of the multi-target widget, the location of the desired target was varied between the third, fourth, and fifth target.

Similar to previous work in expanding targets [74, 44], task ID ranged from 3.17 to 7.01 bits. However, our experiment contains 15 Distance/Width (D/W) combinations

resulting from presenting each task ID at three different distances. The three distances were chosen to correspond to close movements (512px), average movements (1024px), and distant targeting tasks (1526px). Our fifteen combinations of D/W (in screen pixels) were 512/4, 512/8, 512/16, 512/32, 512/64, 1024/8, 1024/16, 1024/32, 1024/64, 1024/128, 1536/12, 1536/24, 1536/48, 1536/96, and 1536/192.

Experimental Conditions In our pilot study, there are two conditions. The first condition is a *Static/No Expansion* condition where the target size never changes during the movement. The other condition is an expansion condition where the candidate targets' widths expand by a factor of two when the cursor has covered 80% of the target distance (D). The targets expanded around a predicted target's center. To simulate predictor behaviour, 40% of the time the predicted target and the user's intended target were the same; 20% of the time one target before and 20% of the time one target after the intended target was the predicted target; and 10% of the time two targets before and 10% of the time two targets after were the predicted targets. As shown in Figure 7.2, mispredictions resulted in the intended target being shifted. If an off-by-two error occurred, the user's target was shifted and was not expanded.

Participants Twelve adult volunteers, seven male and five female, between the ages of 18-32 (mean=24.7, SD=4.1) participated in the study. All participants were affiliated with a local university and received a $10 gift certificate for a local coffee shop for their participation.

Procedure and Misprediction Conditions The experiment consisted of three blocks: one practice block (no expansion) and two experimental blocks: no expansion and expansion using the simulated predictor. The practice and no expansion block consisted of 15 D/W combinations presented six times (twice for each possible target location), resulting in 90 tasks per block. The expansion block consisted of each D/W combination being presented to the user ten times resulting in 150 tasks per block. The order of presentation of the combinations was randomized and the order of the experimental blocks was counterbalanced.

For the ten pointing tasks at each D/W combination, we introduce misprediction conditions to simulate the behaviour of our initial predictor (presented in Chapter 4.3.2). These conditions correspond to the conditions shown in Figure 7.2.

- *Correct Prediction:* In 4 of the 10 movements at each D/W combination, the predicted target was the user's intended target.

- *-1 Prediction:* In 2 of the 10 movements at each D/W combination, the predicted target was the target immediately before the user's intended target, along the user's path of motion.
- *+1 Prediction:* In 2 of the 10 movements at each D/W combination, the predicted target was the target immediately beyond the user's intended target.
- *-2 Prediction:* In 1 of the 10 movements at each D/W combination, the predicted target was two targets before the user's intended target, along the user's path of motion.
- *+2 Prediction:* In 1 of the 10 movements at each D/W combination, the predicted target was two targets beyond the user's intended target, along the user's path of motion.

The visual effect of each misprediction condition on the intended targets in screen space is illustrated in Figure 7.2.

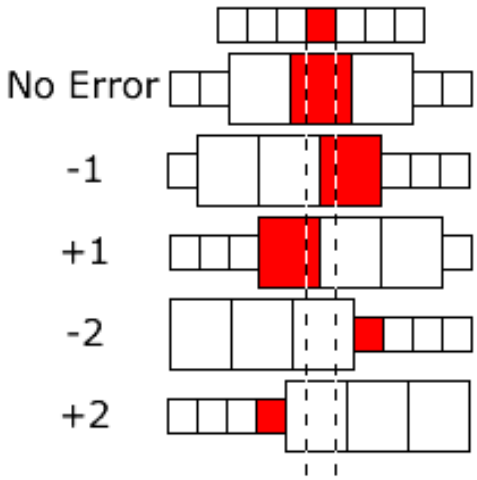


Figure 7.2: Illustrations of each of the five experimental conditions for expansion condition.

Results

Of the 2880 tasks recorded, 4.1% resulted in the user not correctly hitting the target. These tasks were removed from our analysis.

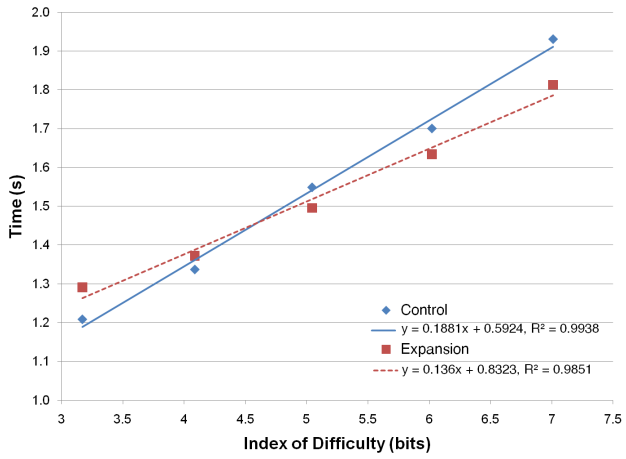


Figure 7.3: Movement times by Index of Difficulty by expanding condition for the user trial.

Figure 7.3 illustrates the overall movement time for experimental versus control conditions by ID. Figure 7.4 further segments movement time using each of the expanding conditions (Correct, $+1$, -1, $+2$, -2). Analysis of variance shows a significant effect for condition (expansion or no expansion) ($F_{1,11} = 8.51, p < 0.01$), misprediction condition (Correct, $\pm 1, \pm 2$) ($F_{4,8} = 27.73, p < 0.01$), and ID ($F_{4,8} = 396.20, p < 0.001$) on movement time. We also observed condition * misprediction interactions ($F_{4,20} = 24.15, p < 0.001$), and condition * ID interactions ($F_{8,16} = 5.97, p < 0.01$) on total movement time.

Analyzing Mispredictions Results from our user trial indicate that even in the presence of mispredictions, users benefit from target expansion. Analysis of variance for tasks in the expansion condition show a significant effect for ID ($F_{4,8} = 154.18, p < 0.001$), misprediction error condition ($F_{4,8} = 59.21, p < 0.001$), order ($F_{2,10} = 18.17, p < 0.01$), and ID * misprediction interactions ($F_{4,24} = 2.507, p < 0.05$) on movement time. Post Hoc analysis using Bonferroni correction shows the -2 misprediction condition to result in the slowest mean movement time followed by the $+2$ condition. Post Hoc analysis also shows the -1 misprediction condition to be significantly slower than the correct prediction and $+1$ condition. However, Bonferroni correction does not show a significant effect on movement time for the correct prediction and $+1$ misprediction condition. Finally, in all cases, when the target expands, the user outperforms the control condition. As well, overall the expansion condition outperforms the control condition based on the simulated

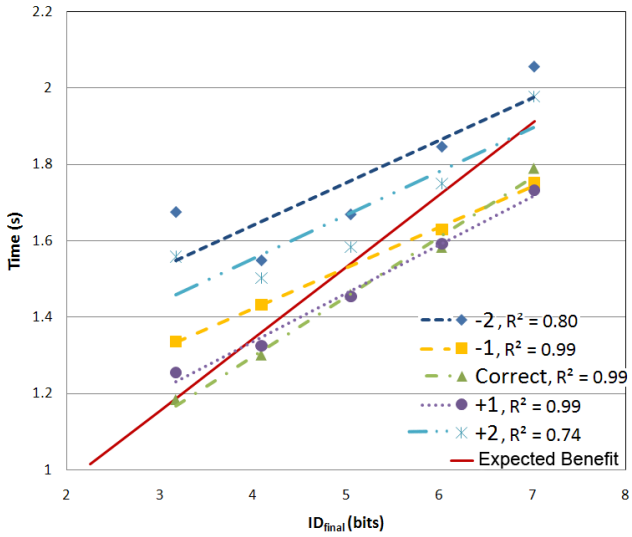


Figure 7.4: Movement time by final Index of Difficulty for the screen expansion condition.

error rates we used. Qualitatively, we note that for all but the lowest IDs (largest, closest targets) the experimental condition was faster than the control condition.

As described by McGuffin and Balakrishnan[44], the maximum expected benefit of expanding a target can be calculated using Fitts' Law with the target's expanded size. Using the target's final width to calculate the task's Index of Difficulty, represented as ID_{final}, we plot the effect on movement time by misprediction alongside the maximum expected benefit represented as a solid line (Figure 7.4). This line was calculated using Fitts' law coefficients from the static condition in Figure 7.3 assuming that we always predict and expand the correct target. In Figure 7.4, we see that correct prediction slightly outperforms the maximum expected benefit, and that the +1 condition, also outperforms the expected benefit for all but the lowest ID. The -1 condition performs, on average, at the lower bound (slightly better for high ID tasks and slightly worse for low ID tasks). Finally, the +2 and -2 conditions do perform worse than the optimal expected benefit, but these conditions are relatively rare with current predictor accuracy (10% probability each).

7.1.4 Real-time Prediction and Target Expansion

Our results in our pilot study demonstrate that expanding a candidate set of targets on the computer screen coupled with optimistic endpoint prediction accuracies improves pointing

performance in a tiled target pointing task. We have shown that the screen expansion of multiple targets improves pointing performance and that users are able to capitalize on an enlarged target despite the presence of mipredictions and target shifts. However, in chapter 5 demonstrated that while high accuracies could be replicated at the distances used by our initial predictor, as distance increased the distributions of predicted endpoints also increased, resulting in lower probabilities of the correct target being predicted. Therefore, at distances further than 512-pixels, even if three targets are expanded, it is much less likely that a user's target will be expanded. To increase the likelihood of target expansion in real world prediction, it is necessary to increase the size of the candidate set as distance increases.

In this section we describe an experiment conducted to examine expanding a region of targets using a real-time implementation of the kinematic endpoint prediction algorithm. In particular, the goal of the experiment is to answer the following questions:

- Will using a real-time predictor and a candidate set of targets, enable us to expand the user's intended target at accuracies defined by a normal probability distribution?
- Is there a limit to the amount of displacement that can occur before performance degrades?
- Finally, is the current state of the art in endpoint prediction accurate enough for enabling expanding targets in interfaces?

Kinematic Endpoint Prediction

To conduct this experiment, we implement the KEP algorithm described in Chapter 4.4 and incorporate it into our pointing task.

Results from our study presented in Chapter 5 show that the standard deviation of the predicted endpoint of a user's motion can be approximated by taking 10% of the distance a user travels. As a result, we consider candidate targets within two standard deviations ($\pm$10%) highly likely, i.e. 68.2% likelihood of target expansion. We base the expansion region of this experiment on this result, as described below.

Method

Task The task was the same task as described in pilot study with one exception. Instead of displaying seven targets, the whole screen was tiled with targets. As a result, the KEP

predictor could choose any region on the screen as the predicted endpoint instead of limiting predictions to, for example, seven targets, as in the pilot study.

Expanding Conditions Similar to our pilot study, our experiment has two conditions.

No Expansion/Control: Target size never changed during the movement.

Expansion: Using a real-time implementation of the kinematic endpoint prediction (KEP) algorithm, predictions were made continuously throughout the motion. At 89% of predicted distance (i.e. current distance is 89% of the predicted endpoint) a prediction was acted upon. At that time, the prediction was used to create a candidate set of targets around the predicted target by either determining the number of targets that can occupy a region defined by ±10% of motion distance (D) or ±1 target, which ever was greater. For example, for an 8-pixel target at a distance of 512 pixels, a candidate set of 13 targets will be expanded because 6 targets fall within 51 pixels on each side of the predicted target. However, for a 64-pixel target at the same distance, only 3 targets will belong to the candidate set since each 64-pixel target is larger than the 51-pixel region defined by 10% of distance traveled. As mentioned above, 10% of motion distance is approximently one standard deviation of the probablity distribution for the region defined by Ruiz and Lank's KEP algorithm. Therefore, we would expect the user's intended target to be contained within a region of ±10% of motion distance (or two standard deviations) approximately 68.2% of the time.

Procedure The experiment consisted of three blocks: one practice block consisting of no expansion and two experimental blocks: control and expansion. Each block consisted of 15 D/W combinations presented ten times resulting in 150 tasks per block. The order of presentation of the combinations was randomized and the order of the experimental blocks was counterbalanced.

Participants 12 adult volunteers, 8 male and 4 female, between the ages of 21-33 (mean=25.8, SD=3.6) participated in the study. All participants were affiliated with a local university and received a $10 gift certificate for a local coffee shop for their participation.

Results

Of the 3600 tasks recorded, 4.1% resulted in the user not correctly hitting the target. There was no significant difference between error rates in the control (5.1%) and expanding (3.1%) conditions.Tasks with errors were removed from our analysis.

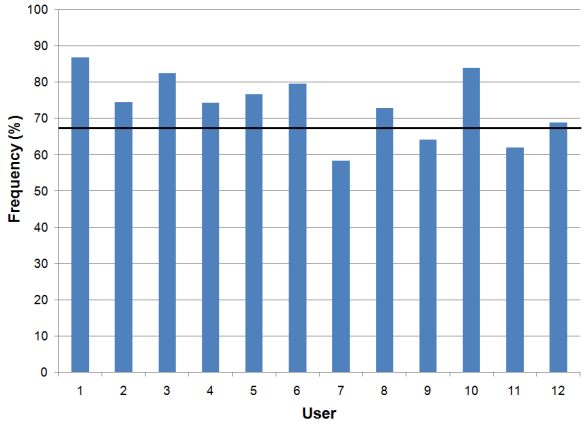


Figure 7.5: Frequency of the user's target expanding as part of the candidate set by user. The bold horizontal line represents the expected frequency of 68.2%

In our experiment, candidate targets consisted of targets occupying ±10% of motion distance around the predicted target. As mentioned above, the region predicted by the KEP algorithm is defined by a normal probablity distribution around the predicted target and approximately 10% of distance represents one standard deviation of the distribution. By expanding a region consisting of ±10% of motion (i.e. two standard deviations), we would expect the user's target to be expanded 68.2% of the time. Results from our experiment indicated that the user's intended target was expanded 73.7% of the time. Figure 7.5 displays the accuracies by user. As shown in the figure, observed accuracies were typically better than expected, with only three users with frequencies below 68.2% (Users 7, 9 and 11). Accuracies reached as high as 86.9% for one user in our study.

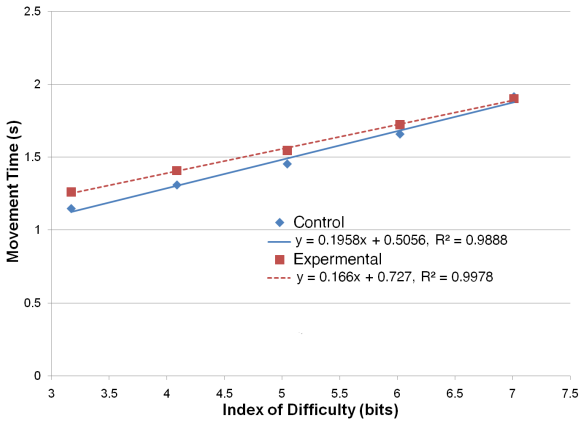


Figure 7.6: Movement times by Index of Difficulty by condition.

Overall movement times by condition and ID are shown in Figure 7.6. Results from the experiment indicate that using the kinematic endpoint predictor to identify a candidate set of targets and expanding that set resulted in slower movement times than the control condition. To examine why, we now focus our attention on the analysis of how mispredictions, both small offset mispredictions and large offset mispredictions, affected movement time.

Analyzing Mispredictions We examine the performance of target expansion for tiled targets by investigating the effect of offset errors on pointing speed. To perform this analysis, we normalize the time taken by motion by subtracting the average time taken in the control condition from the time taken for each individual motion. A value of 0 indicates that the movement speed was identical to the average control condition speed. Negative values indicate the user was faster than their average in the control condition. Positive values indicate that the user was slower.

We also categorize each data point into one of three categories: correct, negative off, and positive off. The correct category represents when the user's intended target was included in the expanded candidate set. Negative and positive off represent when the user's target was not in the candidate set. For Negative off, the algorithm underestimated the user's motion, so the user's intended target was moved farther away from the user. For Positive off, the predicted endpoint was beyond the user's target, so the user's target moved toward the user.

Offset error is measured in two ways, relative to the target's original width and in

Figure 7.7: Performance Benefit/Cost by relative displacement of a user's target for targets of size 4, 16, 32, and 64-pixels for the correct experimental condition.

absolute pixels. Relative offset error defines offset error as a measure of the target's original width. For example, if the KEP predicts the correct target, then the target is not shifted and the relative offset error is 0. If the KEP is off by one target, the relative offset error would be equal to 1. For an error of k targets, the relative offset error is k.

The performance benefit/cost by relative offset error for the correct category is shown in Figure 7.7. As illustrated by the figure, relative displacement has no correlation to the observed benefit/cost. For example, for a 4-pixel target a benefit is observed for relative offset errors as high as 11. Therefore, even though the user's intended target was displaced 11 times the original target's width (44 pixels), the user still capitalizes on the enlarged area of the target. In contrast, for 32-pixel targets, a relative target offset error as low as 1 (32 pixels) negates any benefit of an enlarged target.

Due to the lack of correlation for relative offset error on performance, we focus on absolute offset error, which, as shown in Figure 8, is more strongly correlated with benefit/cost. Absolute offset error is defined as the number of pixels the center of the target was shifted. For example, if the KEP predicts the correct target, then the absolute offset error is 0, the target is not shifted. If the KEP is off by one target, the absolute offset error would be equal to W, the target width, and for an error of k targets, the absolute offset error is kW.

Examination of Figure 7.8 suggests that 0-pixel target displacement (displayed on the horizontal axis) is a point of reflection. Post-hoc analysis using Bonforroni correction confirms this, showing no significant difference between the negative and positive off

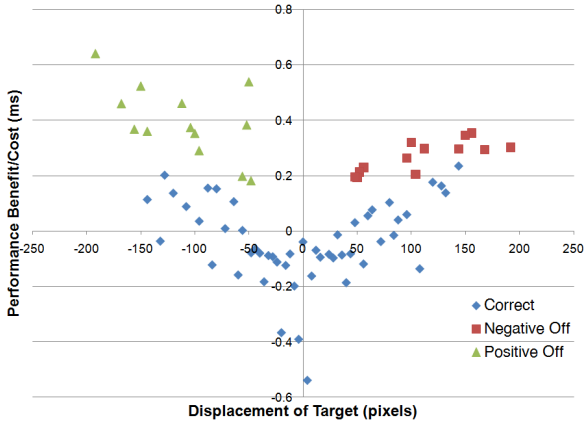


Figure 7.8: Performance Benefit/Cost by absolute displacement of user's target for each category in the experimental condition.

categories. Therefore, we simplify our categories by taking the absolute value of target offset and combining the positive and negative off category into a single *Error* category. Figure 7.9 illustrates the normalized time taken versus offset error for each of our resulting categories, Correct, and Error. The solid line represents the best linear fit for each category.

As in the pilot study, we can calculate the expected best performance from expanding the user's intended target by using the Fitts' Law coefficients obtained from the control condition. Using these coefficients, we calculate the maximum performance benefit for our participants to be -0.20. As in the pilot study, when expansion occurs around the user's intended target, resulting in no horizontal displacement, we observed a performance benefit near the maximum benefit expected by Fitts' Law. However, the performance benefits of the enlarged target quickly degrades as the user's intended target shifts. When the offset error reaches 80 pixels (shown in Figure 7.9(a)), the time taken to acquire a target matches the control condition. For shifts greater than 80 pixels in our experimental configuration, the user performed worse than control, even if the correct target was expanded.

The performance cost for the error condition is shown in Figure 7.9(b). As shown in the figure, even at the lowest level of displacement (50 pixels), the performance is worse than the control condition, suggesting a significant cost for target displacement.

7.1.5 Discussion

While our pilot study using a candidate set of three targets and the simulated prediction accuracies reported in Chapter 4 resulted in promising results, they did not extend to larger sets of targets using a real-time implementation of the KEP predictor. Results from our experiment demonstrate that there is a limit to the amount a user's target can shift when expanded. On our experimental setup, the limit on target displacement was approximately 80 pixels. If displacement was greater than 80 pixels, then target expansion did not result in a net benefit. Figure 7.9(a) plots this expected benefit as a linear function. While the 80-pixel limit on displacement is undoubtedly a function of the resolution of our computer monitor and input device, for any display/input device a relatively simple pointing task can be used to calculate the limits on target displacement for a specific user. See, for example, Wobbrock et al.'s work [69] calculating an error function for Fitts' Law pointing tasks, where they use a simple test to calibrate users.

As we note earlier, we show that the KEP algorithm can be used to calculate a normal distribution around a predicted endpoint with standard deviation, σ_i, approximately equal to 10% of the user's movement. A normal distribution can be used to calculate specific probability of a value lying between the mean and any arbitrary number of standard deviations, s, using the erf function as follows:

$$p(x) = \frac{2}{\sqrt{\pi}} \int_0^s e^{-x^2} dx \tag{7.1}$$

Using our 80 pixel limit for our monitor/input device configuration, we can use this to calculate the necessary accuracy of our predictor. At 0 pixels of displacement, we see a maximum benefit, calculated as about 1/5 of a second, i.e. 0.20 seconds. This benefit shrinks to 0 seconds at 80 pixels, yielding a straight line equation of the form:

$$t_{saved} = -0.20 + \frac{0.20}{80} x \tag{7.2}$$

Beyond 80 pixels, our targets are not expanded, yielding a constant cost based on 80 pixels of displacement of 0.25 seconds of additional time. We can claim the following:

$$p(x) t_{saved} > 0.25(1 - p(x)) \tag{7.3}$$

Essentially, the probability of any time saving associated with expansion must *outweigh* the likely cost associated with 80-pixel target displacements when no expansion occurs. The probability of no expansion is exactly equal to $1 - p(x)$, where $p(x)$ is the probability that

the target expands. Therefore, 80 pixels is equivalent to the number of standard deviations, s in Equation 7.1 such that the inequality, Equation 7.3 holds. Solving this analytically in Maple yields a result that $s = 0.729$. Our displacement limit, 80 pixels, is an arbitrary function of our hardware. However, the predictor accuracy must be greater than 56.5% for our maximum displacement limit. We obtain this calculation by evaluating the integral in Equation 7.1 with $s = 0.73$.

Considering the current standard deviations associated with the KEP, we see that only for distances less than 800 pixels will the predictor perform with sufficiently high accuracy to result in performance gain on our current experimental configuration. Therefore, for distances used for our validation original studies, i.e. distances less than 600 pixels, we would expect to see a net benefit. However, when implementing the real-time predictor on a larger display, only at 512 pixels is the KEP sufficiently accurate to allow a net improvement in performance, and then only if a region of 80 pixels is expanded. Expanding smaller regions will drop the predictor accuracy and limit some of the benefit, resulting in suboptimal benefit for expanding targets.

7.2 Kinematic Endpoint Predictor with Additional Probabilities

Results from the user studies presented in Chapter 5 demonstrated that the KEP technique can be used to create a probability distribution that contains the user's goal target. While this is useful, as target distance increases so does the region identified by KEP making it possible that several targets are identified as likely targets (i.e. have similar probabilities). In this section, we provide an example how KEP can be used in conjunction with other probabilities to correctly identify the user's intended target, even as target distance increases.

7.2.1 EXPECT-K: Expanding Predictive Endpoint Cued Tablet Keyboard

With the increase in popularity of pervasive technology, touch screens and stylus pens have presented themselves as an alternative to physical keyboards in a wide range of computing devices. However, users of these new devices still occasionally require text input. To fill the void of a physical keyboard, these devices allow text input using speech recognition, handwriting recognition, gestures, and/or a virtual keyboard.

Our main focus is on one of the most prevalent text entry techniques for touch/stylus interaction, virtual keyboards. Virtual keyboards allow users to enter text by tapping on an image of a keyboard located on the device's display. Although prevalent, virtual keyboards are much less efficient than their physical equivalent. Unlike with physical QWERTY keyboards where it is common for an experienced typist to reach 60 wpm, performance models estimate the upper bound for text input using a QWERTY virtual keyboard to be between 30-34 wpm [64, 75].

To facilitate faster typing speeds in virtual keyboards, we have designed *EXPECT-K*, the first virtual keyboard to incorporate endpoint prediction, target expansion and visual cues to speed text entry on Tablet PCs. Our work is influenced by performance models [64, 75, 62] that identify two temporal phases required to enter text using a virtual keyboard: the time required to visually locate the appropriate key, and the movement time required to acquire a key.

Unlike previous work that focused on minimizing movement time by optimizing the key layout (i.e. OPTI [39] and ATOMIK [73]), EXPECT-K is designed to minimize both visual search time and movement time through visual cues and target expansion. Figure 7.10 depicts EXPECT-K's visual cues (7.10(a)) and target key expansion (7.10(b)). Our techniques work on QWERTY keyboards and on other optimized soft keyboards, including ATOMIK (7.10(c)) and FITALY.

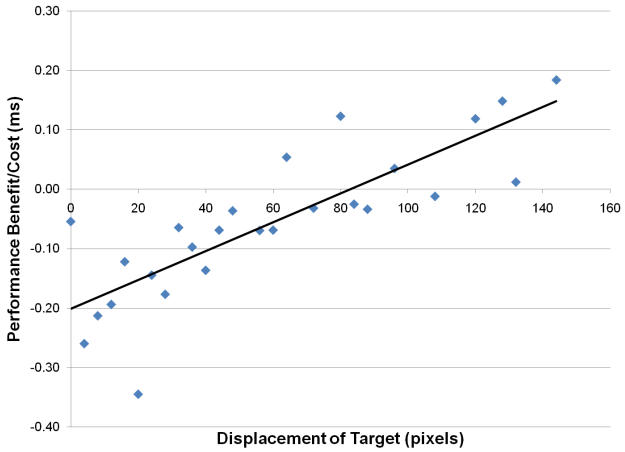
Similar to work by Magnien et al. [40], we propose highlighting likely keys as a technique to minimize the visual search time by potentially reducing the visual search space. To determine which keys should be highlighted, our keyboard uses a tetra-gram model representing adjacent letter frequencies to calculate the probabilities of a key being selected given the user's previous input. The tetra-gram letter frequencies are initialized using the MacKenzie and Sourkoreff phrase set [38] during software installation and then updated continuously during use. The dynamic nature of the model allows the keyboard to adjust to the individual's language usage.

The visual clues work as follows: Initially no keys are highlighted. After the user enters an initial character, the four keys that represent the most frequent tetra-grams are highlighted. The decision to highlight the top four keys is based on our need to minimize the visual search space while maintaining a high probability that the user's intended key will be highlighted.

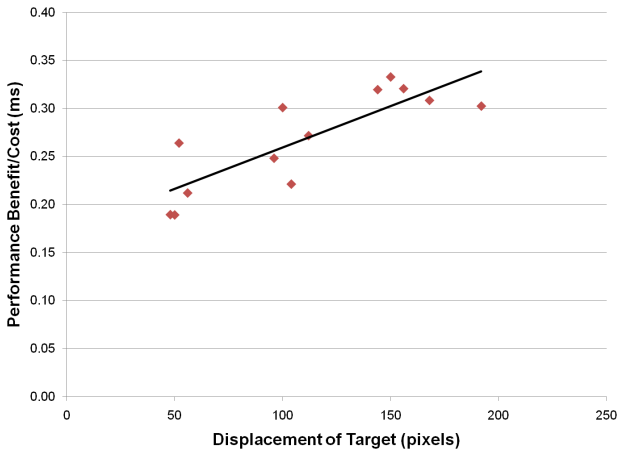
To minimize the time required to acquire a key, EXPECT-K uses expanding widgets. Expansion of the user's intended key is made possible by a real-time implementation of the KEP endpoint prediction algorithm. The result from the endpoint predictor, in conjunction with the tetra-gram letter frequencies, is used to predict which key should be expanded.

The inclusion of the tetra-gram frequencies allows us to reduce the error rate of the endpoint prediction algorithm.

As an expanded key occludes neighboring keys and there is a probability that the key expanded is not the user's intended key, we allow the user to reset the expanded key to its default size by simply moving the stylus out of the Tablet PC's tracking space above the tablet surface.



(a) Correct



(b) Error

Figure 7.9: Performance Benefit/Cost by absolute value of the displacement of user's target for the correct and error categories.

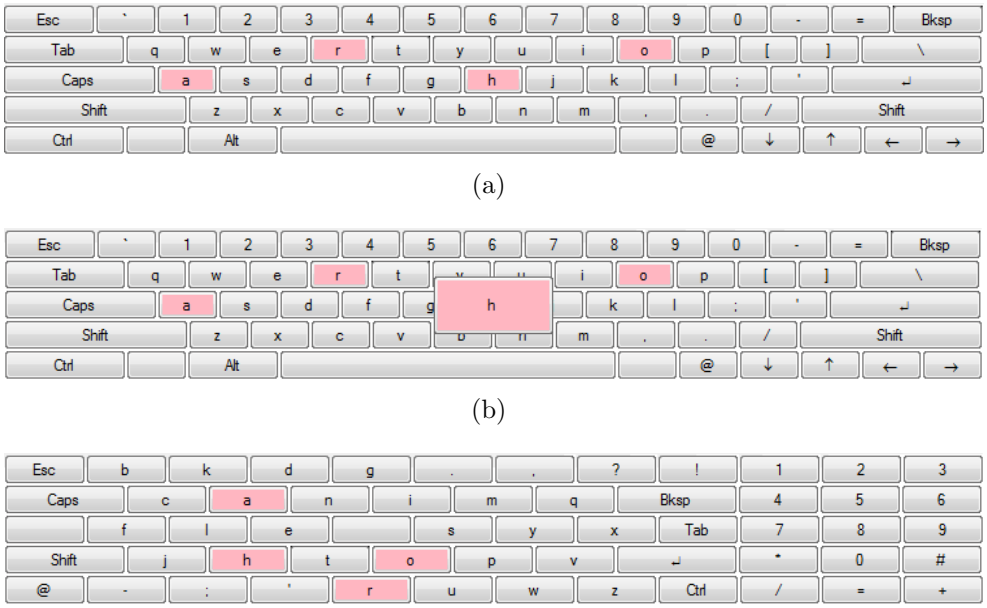


Figure 7.10: Screenshots of EXPECT-K. (a) Visual highlighting of keys. (b) An example of an expanded key. (c) Visual highlighting of keys on the ATOMIK layout.

Chapter 8

Conclusions

As stated in Chapter 1, the goal of this dissertation was to develop and analyze how an understanding of motion kinematics can enhance our ability to predict the target or endpoint of a goal-directed movement in graphical user interfaces. To reach this goal, we presented a wide body of work that examined seven main research problems:

(a) What characteristics need to be accounted for when modeling pointing motion to predict gesture endpoint?

(b) How do the constraints in the interface affect motion characteristics?

(c) Can we identify a design space describing the use of endpoint prediction to enabling pointing facilitation?

(d) Can we develop a technique to predict user endpoint?

(e) How does target size and target distance affect our ability to predict user endpoint?

(f) What level of prediction accuracy is necessary to improve pointing for specific display configurations that are currently resistant to standard pointing facilitation techniques?

(g) Can an endpoint predictor that models the characteristics of motion be used to enable pointing facilitation techniques?

In this chapter, we revisit each of the main research problems and summarize our approach and main findings.

8.1 What characteristics need to be accounted for when modeling pointing motion to predict gesture endpoint?

To determine which characteristics of motion we should model in order to predict endpoint, we began by examining the current literature on goal-directed motion. Using principles of the stochastic optimized-submovement model [46] we hypothesized that the initial 90% of gesture distance was primarly ballistic. Based on this hypothesis we used principles of the minimum jerk principle [66] to calculate theoritical speed versus time and distance curves of a pointing gesture. We then examined data collected from users and found that the real-world data supported our hypothesis. Therefore, we concluded that the first 90% of gesture distance was primarly ballistic. This was also supported by the analysis of the accuracy of the kinematic endpoint predictor using gestures collected from four seperate user studies.

8.2 How do the constraints in the interface affect motion characteristics?

In this dissertation we presented two studies to examine how constraints in the interface affect the motion characteristics of a pointing gesture. First, we examined how target constraint (i.e. whether the target has an amplitude or directional constraint) affects a kinematic profile. Results from our study demonstrated that changes in kinematic profiles occur in the motion perpendicular to the motion gesture and these differences can be detected using the first 70% of motion gesture. However, when examining the speed profiles of the motion, these changes are negated by the dominant colinear motion. Therefore, target constraint does not have a significant effect on the initial ballistic speed profile of a pointing gesture.

In our second study, we examined the effects of intended use on motion kinematics. Results from this study demonstrated that any observable and statistically significant temporal or kinematic changes resulting from the task a user wishes to perform once a target is acquired is mainly limited to the final 10% of motion. Results from these two studies supported our initial hypothesis that the first 90% of gesture distance is primarily ballistic.

8.3 Can we identify a design space describing the use of endpoint prediction to enabling pointing facilitation?

In Chapter 4, we examined the previous work in pointing facilitation in order to describe a taxonomy for techniques that aim to predict gesture endpoint. We began by classifying pointing facilitation into two categories, those that act on a widget (or target) and those that act on a pixel level. Next, we described the design characteristics an endpoint predictor should provide to support each of these categories. The resulting taxonomy consisted of four dimensions (temporal, complexity, scope, execution) and informs interaction designers on the requirements endpoint predictors must fulfill in order to provide support for current pointing facilitation techniques.

8.4 Can we develop a technique to predict user endpoint?

Using our understanding of the characteristics of pointing motion and ballistic motion, we developed the kinematic endpoint predictor technique (KEP) to predict motion endpoint of a pointing gesture. Through several experimental studies, we examined how prediction accuracy is affected by target distance, target size, target dimension (i.e., one or two dimensional targets), and cursor acceleration. We demonstrated that distance has a major affect on our ability to predict motion endpoint and that there exists a strong linear relationship between the deviations of predictions and distance. More importantly, we showed that this relationship can be leveraged to enable KEP to be used to create a probability distribution over a region of targets on this display.

8.5 How does target size and target distance effect our ability to predict user endpoint?

In this dissertation we presented several experimental studies where we examined prediction accuracy of our kinematic endpoint predictor (KEP) for a wide range of target distances, target sizes, and target dimensions (i.e., one or two dimensional targets). Results from our studies demonstrated that as target distance increases our ability to identify user endpoint

decreases in a linear manner. However, we are able to leverage the the relationship between prediction accuracy and distance to identify likely targets on a computer display.

8.6 What level of prediction accuracy is necessary to improve pointing for specific display configurations that are currently resistant to standard pointing facilitation techniques?

In Chapter 7, we presented a user study that examined the effects of mispredicting a user's intended target on movement time. Results from this study demonstrate that it is possible to expand a small set of targets on the computer screen to improve pointing performance. and that, when expanding a region, the benefits of expansion are affected by the target shift, i.e. the size of the offset error and there is a limit to the amount of shift that can occur before benefits are negated. Finally, we demonstrate that, within the expansion region limit and for our display setup, any endpoint predictor must have an accuracy greater than 56.5% to realize a net benefit from expanding targets.

8.7 Can an endpoint predictor that models the characteristics of motion be used to enable pointing facilitation techniques?

To determine if our technique could be used to facilitate pointing, we began by providing evidence from the motor control literature supporting the design decisions we made while creating our technique as well as suggesting that our technique provides optimal results for a technique that attempts to predict user endpoint using only motion kinematics during the first 90% of gesture distance.

Later in the dissertation we demonstrated how our kinematic endpoint prediction technique can be used to enable pointing facilitation by presenting EXPECT-K, a virtual keyboard that uses our endpoint prediction technique and a tetra-gram language model to predict the user's next key during motion.

8.8 Future Work

Given these results presented in this thesis, our future work will have three main focuses: exploring the use of KEP for accessibility, exploring uses for KEP beyond pointing facilitation, and extending KEP beyond the desktop.

Using our understanding of the characteristics of a pointing movement for healthy adults, we intend to explore how this knowledge can be leveraged to facilitate pointing for users with motor impairments. One possible technique we are considering is to map a kinematic profile of a pointing motion from a person suffering from a motor control ailment to one a profile of a healthy adult to allow the system to smooth the cursors trajectory and path, thus, making it easier for the individual to use a mouse. Another technique we are considering is to use KEP to adjust the default cursor acceleration functions which would allow individuals to travel further distances faster while not requiring fine motor movements when trying to click on a target.

While this thesis mainly focused on using KEP to support current pointing facilitation techniques, there are other opportunities to use KEP to enhance interaction. For example, if used in conjunction with a web browser, KEP can be used to predict the next link that the user is likely to click on and begin to pre-fetch the content from that link resulting in faster page loads and a better browsing experience. In future work we plan on exploring possible uses of KEP beyond pointing facilitation.

Finally, since KEP is based on models of physical pointing, we plan on extending KEP beyond the desktop. As large displays and touch-enabled surfaces have become more pervasive, we plan on exploring how KEP can be adapted to different computing paradigms in order to enhance user interactions around these devices.

8.9 Summary

In this dissertation we presented a body of work that explored how an understanding of motion kinematics can enhance our ability to predict the target or endpoint of goal-directed movement in graphic interfaces. Grounded in previous research on goal-directed motion, we examined the initial physical characteristics of motion and determined that the motion is primarily ballistic and that this ballistic motion could be modeled by the minimum jerk principle [27]. Using this understanding, we created the kinematic endpoint prediction (KEP) predictor to identify candidate targets of pointing during motion. Through several experimental studies, we examined how prediction accuracy is affected by target distance,

target size, target dimension (i.e., one or two dimensional targets), and cursor acceleration. We demonstrated that distance has a major effect on our ability to predict motion endpoint and that there exists a strong linear relationship between the deviations of predictions and distance. More importantly, we showed that this relationship can be leveraged to enable KEP to be used to create a probability distribution over a region of targets on this display.

In Chapter 6, we further analyzed the observed accuracies of the KEP predictor from our user studies. We demonstrated that KEP's prediction accuracy behaviour can be explained by the underlying human behaviour associated with goal-directed motion. More importantly, we showed that KEP may provide optimal accuracy for the class of predictors that aim at predicting motion endpoint during the first 90% of motion gesture. In addition, our results suggest that in order to improve prediction accuracy a predictor must account for corrective submovements and that these submovements often occur during the last 10% of motion dustance.

We concluded this body of work by examining the level of prediction accuracy is necessary to improve pointing for specific display configurations that are currently resistant to standard pointing facilitation techniques. Through user experimentation we demonstrated the cost and benefits of using the expanding widgets [44] pointing facilitation in tile target arrangements when mispredictions will occur. We also presented EXPECT-K, an on-screen keyboard that allows users KEP, expanding widets, and a tetra-gram language model to speed text entry. EXPECT-K demonstrates how the probability distributions of KEP can incorporated with additional probabilities to facilitate interaction and enhance interaction in graphic user interfaces.

References

[1] Johnny Accot and Shumin Zhai. Beyond fitts' law. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '97*, pages 295–302, Atlanta, Georgia, United States, 1997. 33

[2] Johnny Accot and Shumin Zhai. Refining fitts' law models for bivariate pointing. In *Proceedings of the conference on Human factors in computing systems - CHI '03*, page 193, Ft. Lauderdale, Florida, USA, 2003. 11, 26, 28, 33, 37, 75, 82, 92, 161, 162

[3] Takeshi Asano, Ehud Sharlin, Yoshifumi Kitamura, Kazuki Takashima, and Fumio Kishino. Predictive interaction using the delphian desktop. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 133–141, New York, NY, USA, 2005. ACM. 1, 25, 76

[4] Ravin Balakrishnan. "Beating" fitts' law: virtual enhancements for pointing facilitation. *International Journal of Human-Computer Studies*, 61(6):857 – 874, 2004. Fitts' law 50 years later: applications and contributions from human-computer interaction. xiii, 1, 9, 20, 22

[5] Patrick Baudisch, Edward Cutrell, Dan Robbins, Mary Czerwinski, Peter Tandler, Benjamin Bederson, and Alex Zierlinger. Drag-and-Pop and Drag-and-Pick: techniques for accessing remote screen content on touch- and pen-operated systems. In *Proceedings of Interact 2003*, pages 57–64, August 2003. xiv, 1, 20, 21, 22

[6] Eric A Bier and Maureen C Stone. Snap-dragging. *ACM SIGGRAPH Computer Graphics*, 20(4):233–240, 1986. 21

[7] Renaud Blanch, Yves Guiard, and Michel Beaudouin-Lafon. Semantic pointing: improving target acquisition with control-display ratio adaptation. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 519–526, New York, NY, USA, 2004. ACM. 1, 21, 23, 24, 28, 62, 128

[8] Renaud Blanch and Michaël Ortega. Rake cursor. In *Proceedings of the 27th international conference on Human factors in computing systems - CHI '09*, page 1415, Boston, MA, USA, 2009. 21

[9] L. Buck. Motor performance in relation to control-display gain and target width. *Ergonomics*, 23(6):579–589, 1980. 23

[10] J. Callahan, D. Hopkins, M. Weiser, and B. Shneiderman. An empirical comparison of pie vs. linear menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '88*, pages 95–100, Washington, D.C., United States, 1988. 20, 21

[11] S. K Card, W. K English, and B. J Burr. Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT. *Ergonomics*, 21(8):601–613, 1978. 9, 10

[12] John Carroll and ScienceDirect (Online service). *HCI models, theories, and frameworks toward a multidisciplinary science*. Morgan Kaufmann,, San Francisco, Calif. :, 2003. xiii, 9, 10

[13] Andy Cockburn and Philip Brock. Human on-line response to visual and motor target expansion. In *Graphics Interface 2006*, pages 81–87, Toronto, Ont., Canada, Canada, 2006. Canadian Information Processing Society. 9, 75, 162

[14] E. R. F. W Crossman and P. J Goodeve. Feedback control of hand-movement and fitts' law. *The Quarterly Journal of Experimental Psychology Section A: Human Experimental Psychology*, 35(2):251 – 278, 1983. 14

[15] Digby Elliot and Michael Khan. *Vision and Goal-Directed Movement: Neurobehavioral Perspectives*. Human Kinetics, 2010. 14, 25

[16] Gaolin Fang, Wen Gao, Xilin Chen, Chunli Wang, and Jiyong Ma. Signer-independent continuous sign language recognition based on SRN/HMM. In *Proceedings of the IEEE ICCV Workshop on Recognition*, pages 90–95, 2001. 159

[17] S. Feiner, S. Nagy, and A. Van Dam. An integrated system for creating and presenting complex computer-based documents. *ACM SIGGRAPH Computer Graphics*, 15(3):181–189, 1981. 21

[18] P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47:381–391, 1954. 9, 10

[19] P. M Fitts and J. R Peterson. Information capacity of discrete motor responses. *Journal of Experimental Psychology*, 67(2):103–112, 1964. 10, 14

[20] T. Flash and N. Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of neuroscience*, 5(7):1688, 1985. 17, 61, 64

[21] James Gordon, Maria Felice Ghilardi, and Claude Ghez. Accuracy of planar reaching movements. *Experimental Brain Research*, 99(1):97–111, 1994. 10.1007/BF00241415. xvii, 119, 120, 121

[22] Evan D. Graham and Christine L. MacKenzie. Physical versus virtual pointing. In *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground*, CHI '96, pages 292–299, New York, NY, USA, 1996. ACM. 30

[23] T. Grossman and R. Balakrishnan. A probabilistic approach to modeling two-dimensional pointing. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 12(3):435–459, 2005. 12, 92

[24] Tovi Grossman and Ravin Balakrishnan. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 281–290, New York, NY, USA, 2005. ACM. 1, 21

[25] Yves Guiard, Renaud Blanch, and Michel Beaudouin-Lafon. Object pointing: a complement to bitmap pointing in GUIs. In *GI '04: Proceedings of Graphics Interface 2004*, pages 9–16, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society. 1, 20, 21

[26] E. R Hoffmann. Effective target tolerance in an inverted fitts task. *Ergonomics*, 38(4):828–836, 1995. 21

[27] N. Hogan. An organizing principle for a class of voluntary movements. *J. Neurosci.*, 4(11):2745–2754, 1984. 3, 120, 149

[28] ISO. *9421–9 Ergonomic requirements for office work with visual display terminals (VDTs) - Part 9: Requirements for non-keyboard input devices.* International Organization for Standardization, 2000. 49, 93, 120

[29] Paul Kabbash and William A. S Buxton. The prince technique: Fitts' law and selection using area cursors. In *CHI '95: Proceedings of the SIGCHI conference on*

Human factors in computing systems, pages 273–279, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. 1, 21, 23

[30] Steven W. Keele. Movement control in skilled motor performance. *Psychological Bulletin*, 70(6, Pt.1):387–403, 1968. 14

[31] David V. Keyson. Dynamic cursor gain and tactual feedback in the capture of cursor movements. *Ergonomics*, 40(12):1287–1298, 1997. 1, 21, 23

[32] Masatomo Kobayashi and Takeo Igarashi. Ninja cursors: using multiple cursors to assist target acquisition on large screens. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 949–958, New York, NY, USA, 2008. ACM. 1, 21

[33] F. Lacquaniti, C. Terzuolo, and P. Viviani. The law relating the kinematic and figural aspects of drawing movements. *Acta Psychologica*, 54(1-3):115–130, 1983. 17

[34] F. Lacquaniti, C. Terzuolo, and P. Viviani. Global metric properties and preparatory processes in drawing movements. In *Preparatory states & processes: proceedings of the Franco-American conference, Ann Arbor, Michigan, August, 1982*, page 357, 1984. 17

[35] Edward Lank, Yi-Chun Nikko Cheng, and Jaime Ruiz. Endpoint prediction using motion kinematics. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 637–646, New York, NY, USA, 2007. ACM. 67, 72

[36] Christine L. MacKenzie, R. G. Marteniuk, C. Dugas, D. Liske, and et al. Three-dimensional movement trajectories in fitts' task: Implications for control. *The Quarterly Journal of Experimental Psychology A: Human Experimental Psychology*, 39(4, Sect A):629–647, 1987. 30

[37] I. S Mackenzie. *Fitts' law as a performance model in human-computer interaction*. PhD thesis, University of Toronto, 1991. 9, 10, 57

[38] I. Scott MacKenzie and R. William Soukoreff. Phrase sets for evaluating text entry techniques. In *CHI '03 extended abstracts on Human factors in computing systems - CHI '03*, page 754, Ft. Lauderdale, Florida, USA, 2003. 141

[39] I. Scott MacKenzie and Shawn X. Zhang. The design and evaluation of a high-performance soft keyboard. In *Proceedings of the SIGCHI conference on Human factors in computing systems the CHI is the limit - CHI '99*, pages 25–31, Pittsburgh, Pennsylvania, United States, 1999. 141

[40] L. Magnien, J. L Bouraoui, and N. Vigouroux. Mobile text input with soft keyboards: optimization by means of visual clues. *Mobile Human-Computer Interaction–MobileHCI 2004*, pages 197–218, 2004. 141

[41] Regan L. Mandryk and Calvin Lough. The effects of intended use on target acquisition. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 1649–1652, New York, NY, USA, 2011. ACM. xv, 18, 19, 26, 28, 47, 48

[42] R.G. Marteniuk, C.L. Mackenzie, M. Jeannerod, S. Athenes, and C. Dugas. Constraints on human arm movement trajectories. *Canadian Journal of Psychology/Revue canadienne de psychologie*, 41:365–378, 1987. 18, 26, 50

[43] Michael McGuffin and Ravin Balakrishnan. Acquisition of expanding targets. In *Proceedings of the SIGCHI conference on Human factors in computing systems Changing our world, changing ourselves - CHI '02*, page 57, Minneapolis, Minnesota, USA, 2002. 21, 23

[44] Michael J McGuffin and Ravin Balakrishnan. Fitts' law and expanding targets: Experimental studies and designs for user interfaces. *ACM Trans. Comput.-Hum. Interact.*, 12(4):388–422, 2005. 1, 9, 21, 23, 24, 25, 28, 29, 71, 75, 76, 80, 82, 83, 85, 92, 124, 125, 126, 128, 132, 150, 161, 162

[45] Stefan Müuench and Rüdiger Dillmann. Haptic output in multimodal user interfaces. In *Proceedings of the 2nd international conference on Intelligent user interfaces*, IUI '97, pages 105–112, New York, NY, USA, 1997. ACM. 21

[46] D. Meyer, J. Smith, S. Kornblum, R. Abrams, and C. Wright. Speedaccuracy tradeoffs in aimed movements: Toward a theory of rapid voluntary action. In *Attention and Performance XIII*, pages 173 – 226. Erlbaum Hillsdale, 1990. xiii, 2, 16, 17, 29, 122, 146

[47] Ian Oakley, A. Adams, Stephen Brewster, and Philip Gray. Guidelines for the design of haptic widgets. *PEOPLE AND COMPUTERS*, pages 195–212, 2002. 21

[48] Ian Oakley, Stephen Brewster, and Philip Gray. Solving multi-target haptic problems in menu interaction. In *CHI '01 extended abstracts on Human factors in computing systems - CHI '01*, page 357, Seattle, Washington, 2001. 21

[49] Hilde Oirschot and Adrian Houtsma. Cursor trajectory analysis. In Stephen Brewster and Roderick Murray-Smith, editors, *Haptic Human-Computer Interaction*,

volume 2058 of *Lecture Notes in Computer Science*, pages 127–134. Springer Berlin / Heidelberg, 2001. 64

[50] J. Ou, L. Min Oh, S.R. Fussell, T. Blum, and J. Yang. Analyzing and predicting focus of attention in remote collaborative tasks. In *ICMI'05: Proceedings of the International Conference on Multimodal Interfaces*, pages 116–123, 2005. 159

[51] R. Plamondon and Sargur N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000. 159

[52] William H Press, William T Vetterling, Saul A Teukolsky, and Brian P Flannery. *Numerical Recipes in C++: the art of scientific computing*. Cambridge University Press, New York, NY, USA, 2nd edition, 2002. 64, 66

[53] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, February 1989. 159

[54] David A. Rosenbaum. *Human motor control*. Academic Press, San Diego ; Toronto, 1991. 15

[55] David A. Rosenbaum, Rajal G. Cohen, Ruud G. J. Meulenbroek, and Jonathan Vaughan. Plans for grasping objects. In Mark L. Latash and Francis Lestienne, editors, *Motor Control and Learning*, pages 9–25. Springer US, 2006. 10.1007/0-387-28287-4_2. 18

[56] Jaime Ruiz, Andrea Bunt, and Edward Lank. A model of non-preferred hand mode switching. In *GI '08: Proceedings of graphics interface 2008*, pages 49–56, Toronto, Ont., Canada, Canada, 2008. Canadian Information Processing Society. 59

[57] Jaime Ruiz and Edward Lank. Speeding pointing in tiled widgets: understanding the effects of target expansion and misprediction. In *Proceeding of the 14th international conference on Intelligent user interfaces*, IUI '10, pages 229–238, New York, NY, USA, 2010. ACM. 9

[58] D.D. Salvucci. Inferrring intent in eye-based interfaces: Tracing eye movements with process models. In *CHI'99: Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 254–261, 1999. 159

[59] R. A Schmidt, H. N. Zelaznik, and J. S. Frank. Sources of inaccuracy in rapid movement. *Information processing in motor control and learning*, pages 183–203, 1978. 15

[60] R. A Schmidt, H. N. Zelaznik, B. Hawkins, and J. S. Frank. Motor-output variability: A theory for the accuracy of rapid motor acts. *Psychological Review*, 86(5):415–451, 1979. 15

[61] Julia Schwarz, Scott Hudson, Jennifer Mankoff, and Andrew D. Wilson. A framework for robust and flexible handling of inputs with uncertainty. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, pages 47–56, New York, NY, USA, 2010. ACM. 63

[62] A. Sears, J. A Jacko, J. Chu, and F. Moro. The role of visual search in the design of effective soft keyboards. *Behaviour & Information Technology*, 20(3):159–166, 2001. 141

[63] C. E Shannon and W. Weaver. *The mathematical theory of communication*. University of Illinois, 1949. 9

[64] R. W Soukoreff and I. S MacKenzie. Theoretical upper and lower bounds on typing speed using a stylus and a soft keyboard. *Behaviour & Information Technology*, 14(6):370–379, 1995. 141

[65] I.E. Sutherland. Sketchpad: a man-machine graphical communication system. In *Proceedings AFIPS Conference, Michigan*, volume 23, pages 329–346, 1963. 21

[66] P. Viviani and T. Flash. Minimum-jerk, two-thirds power law, and isochrony: converging approaches to movement planning. *J Exp Psychol*, 21:32–53, 1995. 18, 29, 61, 146

[67] S. A Wallace and K. M Newell. Visual control of discrete aiming movements. *The Quarterly Journal of Experimental Psychology Section A*, 35(2):311–321, 1983. 15

[68] Tian-Shu Wang, Heung-Yeung Shum, Ying-Qing Xu, and Nan-Ning Zheng. Unsupervised analysis of human gestures. *Lecture Notes in Computer Science*, 2195:174–181, 2001. 159

[69] Jacob O. Wobbrock, Edward Cutrell, Susumu Harada, and I. Scott MacKenzie. An error model for pointing based on fitts' law. In *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, page 1613, Florence, Italy, 2008. 139

[70] Jacob O Wobbrock, James Fogarty, Shih-Yen Liu, Shunichi Kimuro, and Susumu Harada. The angle mouse: target-agnostic dynamic gain adjustment based on angular

deviation. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 1401–1410, New York, NY, USA, 2009. ACM. 1

[71] R.S. Woodworth. The accuracy of voluntary movement. *The Psychology Review*, III(2), July 1899. 2, 13, 14

[72] Aileen Worden, Nef Walker, Krishna Bharat, and Scott Hudson. Making computers easier for older adults to use: area cursors and sticky icons. In *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 266–271, New York, NY, USA, 1997. ACM. 1, 21, 23

[73] S. Zhai, M. Hunter, and B. A Smith. Performance optimization of virtual keyboards. *Human–Computer Interaction*, 17(2):229–269, 2002. 141

[74] Shumin Zhai, Stéphane Conversy, Michel Beaudouin-Lafon, and Yves Guiard. Human on-line response to target expansion. In *Proceedings of the conference on Human factors in computing systems - CHI '03*, page 177, Ft. Lauderdale, Florida, USA, 2003. 1, 21, 23, 24, 28, 29, 92, 128

[75] Shumin Zhai, Alison Sue, and Johnny Accot. Movement model, hits distribution and learning in virtual keyboarding. In *Proceedings of the SIGCHI conference on Human factors in computing systems Changing our world, changing ourselves - CHI '02*, page 17, Minneapolis, Minnesota, USA, 2002. 141

Appendix A

Hidden Markov Models for Kinematic Analysis

Hidden Markov Models (HMMs) are a type of graphical model, essentially a probabilistic finite state automaton. HMMs are particularly well suited to learning and classifying sequential data. HMMs were first introduced to the machine learning community in the 1990's by Rabiner [53]. HMMs are commonly used in pen and non-pen based gesture recognition [16, 51, 68]. In HCI research, HMMs have been used for purposes such as recognizing user intent in eye-based interfaces (i.e., interfaces controlled by eye movements) [58] and predicting a user's focus of attention in remote collaborative tasks [50].

At a high level, an HMM can be viewed as a functional mapping of a sequence of observations to a probability. The probability represents the likelihood that the automaton generated the observations. A different HMM is created (trained using labeled data) for each of the possible classifications of the observations. To perform recognition, an unlabeled observation is assigned to the class whose HMM has the highest likelihood of having generated that observation. In the remainder of this section we describe the distinction between a Markov model and a Hidden Markov Model, describe how a HMM classifies information, and describe specifically how HMMs are used in this work.

A Markov model is a graph with nodes and arcs. Each node represents an internal state while each arc represents a transition between nodes. A Hidden Markov Model is an extension of a Markov model where the current internal state cannot be directly inferred from the observation, as multiple states could produce the same observation.

To model a pointing motion, one state will model a specific region of the motion. Together, all the states of a HMM will contribute to the complete pointing motion. This

is illustrated in Figure A.1. The upper plot shows the pointing motion superimposing the region that each state represents. The lower figure illustrates the path through the HMM that corresponds to the pointing motion. In this situation a particular observation o_i is most probably generated by state 3 as it has a value of $V_x = 5$. Using this rational we can determine the probability that the entire observed sequence was generated by the HMM.

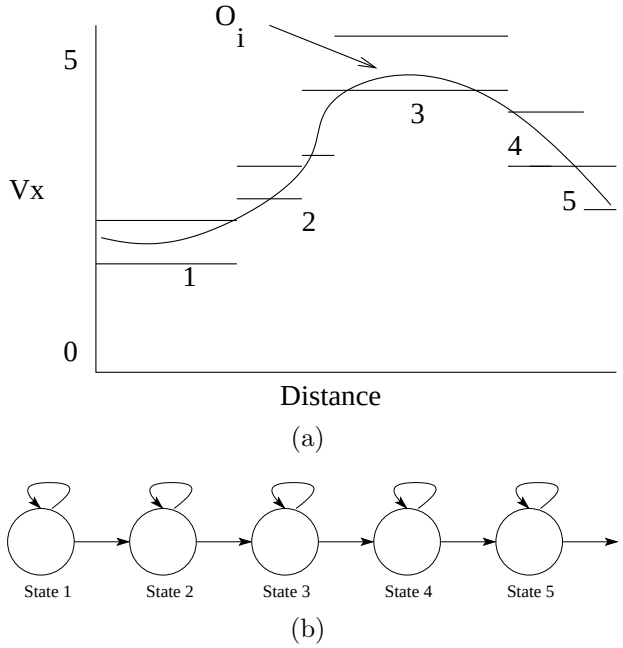


Figure A.1: A depiction of a HMM. At the top, the probability distributions for the states within the HMM given two observations, a distance and the x-component of velocity. At the bottom, the topology of the HMM.

In the context of this thesis, continuous Gaussian HMMs are used as follows. First, we learn the parameters of a HMM so that a single HMM, $\lambda_{amplitude}$, models amplitude-constrained motion and a second HMM, $\lambda_{directional}$ models directionally-constrained motion. Second, given an unlabeled motion, O, we can compute the probability that the motion was generated by either $\lambda_{amplitude}$ or $\lambda_{directional}$. We classify the motion as amplitude constrained if $P(O|\lambda_{amplitude})$ is greater than $P(O|\lambda_{directional})$.

Appendix B

Supplemental User Trial: Precision Over Target IDs

The goal of this experiment was to determine the effect of Index of Difficulty (ID) and distance on the accuracy of the KEP algorithm.

B.1 Method

The study was conducted on a generic desktop computer (Core 2 Duo, 3.0GHz) with a 24-inch 1920x1200 LCD display running custom software written in C#. Input was collected using a Wacom Intuos3 five-button mouse on a 12x19 inch tablet set to a 1:1 control display ratio. The 1:1 control display ratio ensured that motor space and visual space coincided throughout the pointing task as well as maintaining experimental validity and consistency with other studies [44, 2]. The Wacom tablet was used because of its high sampling rate. The custom software captured mouse movements at 200Hz.

The experimental task was a discrete, one-dimensional pointing task. As our goal is to contrast mouse pointing to Lank et al.'s stylus predictions, a one-dimensional pointing task preserves internal validity.

Initially a green starting rectangle was displayed on the screen. The task began when the participant used the cursor to hover over the starting location. After a period of 1 second, a red target would appear on the display. Participants were required to move the cursor to the red target and use the mouse button to click on the target. A successful target acquisition (i.e., clicking within the target region) was indicated by the target changing

color. Users were told to acquire the target as quickly and accurately as possible, similar to other Fitts' Law experiments (e.g. [44, 2, 13]).

The experiment consisted of a within-subjects design with repeated measures. The independent variables were target ID and distance. Five IDs (3.17, 4.09, 5.04, 6.02, and 7.01) at three distances (512, 1024, and 1536 pixels) were used to create 15 distance/width combinations (in pixels): 512/4, 512/8, 512/16, 512/32, 512/64, 1024/8, 1024/16, 1024/32, 1024/64, 1024/128, 1536/12, 1536/24, 1536/48, 1536/96, and 1536/192.

The experiment consisted of eight blocks: one practice block and seven experimental blocks. Each block consisted of 15 D/W combinations presented twice resulting in 30 tasks per block. The order of presentation of the dis-tance/width combinations and constraints were randomized. To minimize fatigue, participants were required to take a five minute break between blocks. The experiment took approximately 60 minutes to complete. Eight subjects, two female and six male, all right-handed, participated in the experiment. All participants were university students and received a remuneration consisting of a $10 gift certificate to a local coffee shop for participating.

Task errors occurred when the user did not correctly acquire the target and accounted for 5.1% for all tasks collected. There was no significant effect of ID on error rate. Task errors were omitted from our analysis of endpoint predic-tion accuracy.

B.2 Results

In this section we describe the results from our user study. Results are presented as described in the Analysis and Measurements section (Section 5.1) found in Chapter 5.

B.2.1 Continuous Prediction

Table B.2.1 shows target prediction accuracies by percentage of gesture length. As in the replication study presented in Chapter 5, the predictor appears to have highest predictive power at 90% of gesture length. However, the KEP algorithm performs significantly below the levels seen in the replication study. As target ID increases, the ability for the KEP algorithm to predict the user's target decreases.

Figure B.1 illustrates prediction accuracy by distance and ID. Analysis of variance of prediction accuracy (pixel error) for distance, ID and user at 90% of gesture length shows a sig-nificant effect for distance ($F_{2,xx} = 65.42, p < .001$), ID ($F_{4,xx} = 30.02, p < .001$)

	Correct	**±1 Target**
80% Gesture Length	10.3%	27.1%
85% Gesture Length	15.5%	30.5%
90% Gesture Length	20.7%	36.0%

Table B.1: Observed frequencies continuous prediction by gesture length.

and user ($F_{4,xx} = 15.11, p < .001$). ANOVA also shows a significant effect for distance*ID interaction (F8, 1252 = 7.03, p ¡ .001), user*distance interac-tion ($F_{14,xx} = 5.71, p < .001$), and user*ID interaction ($F_{28,xx} = 2.11, p < .001$). Post-hoc analysis using Bonferroni correction shows significant differences between all the distances ($p < .001$ for all cases). Post-hoc analysis also indicates a significant difference between all other IDs ($p < .05$ in all cases).

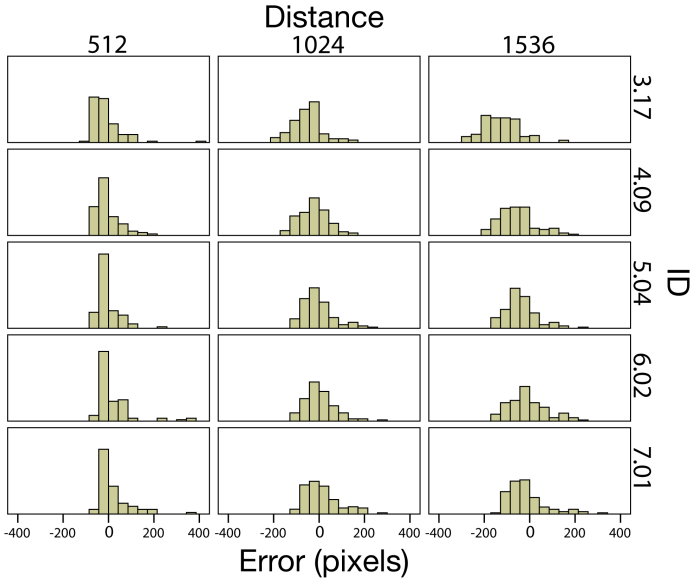


Figure B.1: Prediction accuracy distributions by distance and ID for the continuous prediction strategy.

B.2.2 Single-shot Prediction

We examine three different thresholds for our single-shot predictor. Results from the analysis are summarized in Table B.2. Again while a 90% threshold results in high overall accuracies, most correct target predictions occur after 90% of actual gesture length. Using the 85% gesture length threshold resulted in the highest target accuracy of 10.6% of the time the correct target being predicted and within ±1 target a total of 30.4% of the time. As in the replication study, the single-shot predictor target accuracy is below that of the continuous pre-dictor highlighting the difficulty of determining the proper time to make a prediction.

	Correct	**±1 Target**
80% Gesture Length	10.3%	27.1%
85% Gesture Length	15.5%	30.5%
90% Gesture Length	20.7%	36.0%

Table B.2: Accuracy rates for single-shot prediction by threshold and percentage of actual distance.

Analysis of variance of prediction accuracy (shown in Figure B.2) using the 85% gesture length threshold yielded similar results as our continuous prediction analysis showing a significant effect for distance ($F_{2,xxx} = 17.01, p < .001$), ID ($F_{4,xxx} = 13.46, p < .001$), and user ($F_{7,xxx} = 23.70, p < .001$). ANOVA on pixel error also indicates distance*ID interac-tion ($F_{8,xx} = 3.04, p < xxxx$), distance*user interaction ($F_{14,xxx} = 7.27, p < .001$), and ID*user interaction ($F_{28,xxx} = 1.66, p < .001$) . Post-hoc analysis for distance shows a significant difference between all distances ($p < .01$ in both cases). Post-hoc analysis for ID shows a significant difference be-tween ID 3.17 and all other IDs ($p < .005$ in all cases).

B.3 Discussion

Regardless of evaluating the KEP algorithm using a continuous or single-shot prediction strategy, ID and distance have a significant effect on prediction accuracy. As shown in Figure B.1 and Figure B.2 (moving left to right), as distance increases, the distribution of prediction accuracy also in-creases. However, as ID increases (moving top to bottom), the distribution tends to decrease. Recall that as ID increases, target width decreases. Therefore, the improvement in prediction accuracy as ID increases may be a result of a smaller target and not a distance*target size interaction captured by a target's ID.

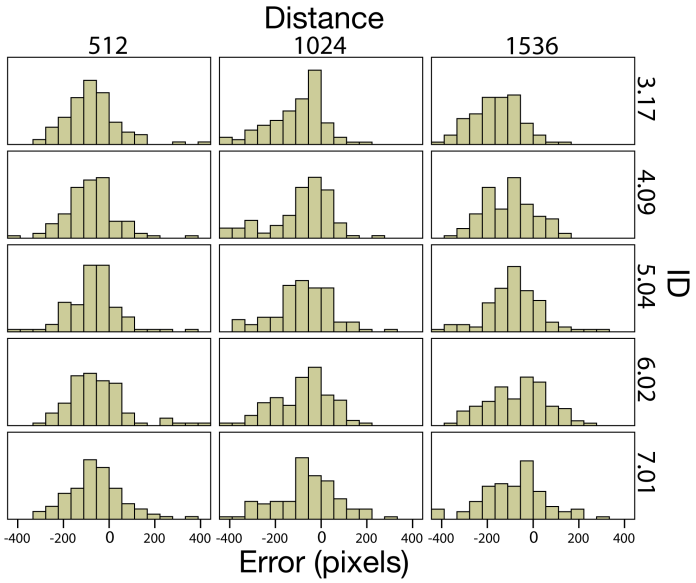


Figure B.2: Prediction accuracy by distance and ID for the single-shot prediction strategy.

In order to examine if distance or target width have a greater effect on prediction accuracy, user experiments in Chapter 5 analyze the effect of distance and target size, to determine whether, distance, target size, or distance/target size interaction (i.e. ID) is more important in determining KEP prediction accuracy.