Effects of Target Size and Distance on Kinematic Endpoint Prediction

Jaime Ruiz and Edward Lank
David R. Cheriton School of Computer Science
University of Waterloo
{jgruiz,lank}@cs.uwaterloo.ca
Technical Report CS-2009-25, University of Waterloo

ABSTRACT
Because of the ubiquity of the WIMP paradigm, many researchers seek to design new pointing facilitation techniques for Fitts-style pointing tasks. However, many of these pointing facilitation techniques make one of two simplifying assumptions: either salient targets are sparsely placed on the display, or there exists some ability to identify the endpoint, the target, of a user's movement in real time. In this paper we extend previous work on kinematic endpoint prediction (KEP), a technique that uses models of user motion to predict endpoint in Fitts-style pointing tasks. We introduce a simplified algorithm to predict user endpoint. We present a technique to measure the numerical stability of endpoint predictions in real time. We show that the distance of motion has a significant effect on predictor accuracy. Finally, we develop an accurate understanding of the relationship between movement distance and predictor accuracy and show how we can use this understanding to infer accurate, real-time probability distributions on target sets within an interface. Together, these results allow KEP to be applied in new and novel ways to pointing facilitation techniques.

Author Keywords
Cursor prediction, motion, kinematics, pointing.

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION
Pointing, with a mouse, electronic stylus, touchpad, trackpoint, 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. To facilitate pointing, HCI researchers have proposed techniques that alter the target [4,6,9,13,19], alter the distance between the cursor and target [1,7,10,11,3], or change pointing interaction [18]. In his survey of pointing facilitation techniques, Balakrishnan [2] 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 [4,13] that salient targets are frequently tiled into small regions on the display, i.e. 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. McGuffin and Balakrishnan [13] conclude that, to effectively support pointing facilitation for many common target arrangements, there is a requisite ability to accurately predict — during user motion — the likely target of a user's pointing gesture.

Several researchers have examined the ability to predict which targets a user will most likely access on a computer display. Some techniques look at underlying priors of widgets based on user models to predict interaction [17]. Others look at kinematic characteristics of the user's movement, i.e. the direction and spatiotemporal profiles of movement. Our interest in target prediction is focused on the latter category of predictors, those that use kinematic characteristics of a user's current motion to identify the user's intended target.

Of the proposed kinematic techniques for endpoint prediction [1,13], the most sophisticated technique reported in the literature is a technique by Lank et al.[12]. Using established kinematic models of motion, the researchers derived a kinematic endpoint prediction (KEP) algorithm able to predict a user's target 42% of time and an adjacent target an additional 39% of the time (assuming tiled arrangements of targets).

While the KEP algorithm is a positive direction in being able to identify a user's endpoint or target during pointing, several specific questions remain regarding the utility of the KEP algorithm for desktop interfaces. First, the algorithm was developed and evaluated using a Tablet PC. Do the assumptions that underlie the model and the accuracy of the predictor still hold for mouse movement? The algorithm is based on the stochastic optimized-submovement model [15], so it seems reasonable to assume that the assumptions and accuracy would hold for mouse input, but validation of

this is necessary. Second, the evaluation of KEP accuracy used a set of target distances (200 to 600 pixels) and target sizes (15 to 75 pixels) that are common on Tablet PCs but are considered limited given the large displays commonly found on desktop computers. Finally, we have some concern that the accuracies reported by Lank et al. may be an over-estimate of real world accuracies of the predictor. Lank et al.'s accuracies assume that a prediction can be constantly revised during motion. However, in real world usages, a predictor often has only one chance to correctly identify a target.

The goal of this work is to extend Lank et al.'s initial work on endpoint prediction. Specifically, this paper makes the following contributions to endpoint prediction using kinematic models:

- We create a simplified predictor that is both more numerically stable and easier to implement than the original KEP algorithm.
- We develop a technique to measure the numerical stability of our predictions in real time. This allows us to avoid acting on assumptions when the predictor has not collected sufficient data to make a stable prediction.
- We replicate Lank et al.'s original tablet study using mouse motion and show that the predictive accuracy of our new algorithm on mouse movement is slightly better than Lank et al.'s accuracy on stylus movement with their target size/target distance combinations. We also show that the accuracy of single-shot prediction is only slightly lower than the accuracy of continuous prediction, and argue that single-shot prediction is a more salient measure of predictor accuracy.
- We extend Lank et al.'s work to a broader set of target and distance combinations. We show that the distance of motion has a significant effect on predictor accuracy. Predictions at longer distances are less accurate than predictions at shorter distances. As well, as work on endpoint deviation [16] would suggest, the accuracy of our predictor has an inverse linear relationship with distance traveled during pointing movement.
- 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, thus allowing our new KEP algorithm to combine its predictions with other probabilistic predictors, such as priors on interface widgets or analysis of a user's task.

In summary, this paper presents a significant enhancement of our understanding of the use of kinematic models in target prediction for Fitts' Law [5] tasks in interfaces. We use kinematic models to make predictions for mouse movement. We develop real-time probability distributions over targets using our kinematic predictor. We argue that our real-time probability distributions can be combined with other independent probability distributions (i.e., priors on interface widgets) to improve the accuracy of prediction in interfaces. Together, this knowledge significantly boosts the potential utility of KEP algorithms in user interface techniques.

This paper is organized as follows. First we describe related work in endpoint prediction including kinematic models. This is followed by a description of how we refine Lank et al.'s KEP algorithm. Next we describe and present results of user trials conducted to examine KEP performance using mouse motion on desktop-sized displays. We conclude the paper with a discussion of the implications of our work.

RELATED WORK

There has been relatively little Human-Computer Interaction work on endpoint prediction using kinematic models. Current techniques use extrapolation to identify a user's endpoint. Prior to Lank et al.; two other predictors appeared in the literature: one by Assano et al., and one by McGuffin and Balakrishnan.

Assano et al. [1] proposed a technique to predict endpoint using peak movement velocity. Their technique consists of identifying peak velocity, looking at the gesture length just prior to reaching peak velocity, and multiplying the distance traveled before peak velocity by a scale factor. They do not report predictor accuracies

While working on expanding targets, McGuffin and Balakrishnan [13] explore whether predictions closer to the endpoint of motion might yield higher accuracy. They present a target predictor based on linear deceleration from the current point. Their technique predicts a user's target with approximately 21% accuracy.

Of the proposed techniques for endpoint prediction, the most accurate technique reported in the literature is a technique proposed by Lank et al. [12]. One reason for the higher accuracy is undoubtedly that the Lank et al. technique makes use of the entire profile of user motion, rather than basing prediction on a discrete point, as in Asano et al. [1], or a small segment of motion, as in McGuffin and Balakrishnan [11].

Lank et al. Kinematic Endpoint Prediction

The Lank et al. kinematic endpoint prediction (*KEP*) algorithm was developed using principles from the stochastic optimized-submovement model [15] and minimum jerk law [8]. The stochastic optimized-submovement model of Meyer et al. predicts that targeted movement occurs in two stages [15]. The first stage consists of primarily ballistic motion aimed at the center of the motion's target. The second stage consists of secondary submovements that correct the initial movement. Therefore, goal directed movement is a stochastic optimization problem, where the higher probability of needing secondary corrective submovements from higher initial motion amplitudes trades off against the shorter time to traverse the distance to the final target. Lank et al.'s KEP algorithm models the initial ballistic movement to predict user endpoint.

To model the ballistic movement, Lank et al. use principles from the minimum jerk law [8]. The minimum jerk law states that for unconstrained ballistic motion, jerk (the time derivative of acceleration) is minimized over the movement. Using the minimum jerk law, the researchers derive an equation to model instantaneous speed over distance. Using this equation, they demonstrate that the extrapolation of a quadratic equation can be used to determine gesture length along a partial gesture. However, the extrapolation is inaccurate [12]. To overcome inaccuracies in the quadratic approximation, Lank et al. use an "extrapolate-then-correct process" which consists of multiplying the endpoint determined by the extrapolation by a pre-determined coefficient (c_r) based on the percentage of gesture distance traveled. Therefore, the steps of the endpoint algorithm are as follows:

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. *Given* x_{calc} use the equation $d=s_i c_r x_{calc}$ and a table of (s_i, c_r) pairs, where s_i is the fractional distance from the estimated endpoint, to determine the value of the coefficient, c_r.
4. Finally, multiply c_r by x_{calc} to determine an estimate of actual gesture length.

Validation of the Model

To validate the model, the researchers performed a user trial where participants used a stylus on a tablet computer to perform a Fitts-style pointing task. Target distances ranged from 200 to 600 pixels by 100-pixel increments while target widths ranged from 15 to 75 pixels by 15-pixel increments. The twenty-five distance/width combinations resulted in an *Index of Difficulty*[1] (ID) range between 1.41 and 5.32.

The results from their study demonstrated that the algorithm was twice as accurate as previous techniques, predicting a user's target 42.5% of the time and an adjacent target an additional 39% at 80% of gesture length [12]. However, as noted by the researchers, the predictive power of the algorithm was significantly lower for the 15-pixel target at 600 pixels distance, indicating that a possible ID limit to predictive power may exist.

REFINING KINEMATIC ENDPOINT PREDICTION (KEP)

While the primary goal of our work was a generalization of KEP to desktop interfaces, three potential refinements to the design and analysis of Lank et al.'s original KEP algorithm became apparent during our research. First, the algorithm could be simplified. Second, we found that numerical stability problems existed, and developed a technique to measure the numerical instability and to identify a *stabilization point* for the KEP algorithm. Finally, we found that Lank et al.'s accuracies, while valuable in understanding KEP's potential best-case performance, contributed little to inform us of KEP's expected case performance. We address each of these issues here.

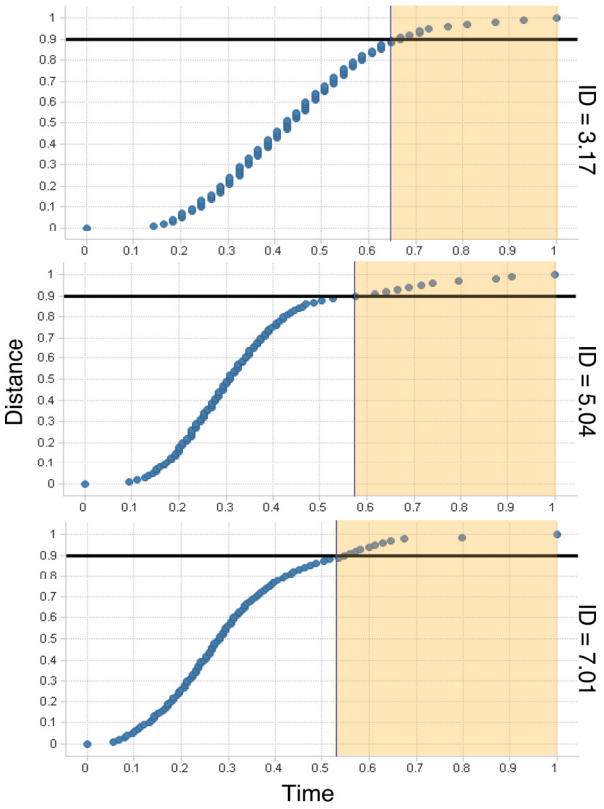

Figure 1: Examples of distance vs. time (both normalized) plots from the study for three IDs. The dark horizontal line represents 90% of gesture distance. The shaded region represents the time taken to complete the last 10% of distance.

Simplifying Endpoint Prediction

Both KEP and McGuffin and Balakrishnan's algorithm [13] predict endpoint quite late during movement. In the case of KEP, prediction is made at 80% of motion, while in McGuffin and Balakrishnan [13], prediction is performed at 90% of motion. To understand why this late prediction is possible, it is informative to look at graphs of distance versus time for experimental participants in our studies (shown in Figure 1).

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 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.

[1] *Index of Difficulty* (*ID*) is the logarithmic product of target distance (A) and target size (W) plus 1, i.e., $\log_2\left(\frac{A}{W}+1\right)$. See [5].

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. Particularly in the ID = 7.01 and ID = 5.04 motion profiles, 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 their work on KEP, Lank et al. use an extrapolate-then-correct procedure, where they fit a quadratic (degree-2) polynomial to speed versus distance profiles. This underestimates endpoint, so they 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 [13]; 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

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, and Lank et al. spend significant time discussing this issue[12]. Figure 2 in our results section also demonstrates this numerical instability. Particularly early in movement, when we try to extrapolate distant points, the boxes and whiskers on our diagram demonstrate 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 80% 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 algorithm, we capitalize on numerical instability to identify when an accurate prediction is unlikely. During pilot studies, we observed that over the movement, our modified KEP stabilizes such that:

$$\frac{l_n - l_{n-1}}{l_n} \rightarrow 0 \quad (1)$$

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 Algorithm

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 1 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 1. If the prediction is determined to be unstable, i.e. Equation 1 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.

Analysis of Kinematic Endpoint Prediction

The final issue we deal with in this section is how to accurately represent real world accuracies of KEP.

Predicting Endpoint

In this work, we use two different prediction strategies to analyze the endpoint predicted by the KEP algorithm, *continuous* and *single-shot*. Continuous prediction is similar to that performed by Lank et al.[12]. First, each task is normalized using interpolation such that distance ranges from 0 to 1 at regular intervals. At each interval, the KEP algorithm is then used to predict the user's endpoint.

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 [14]. Using our algorithm, 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.

Measuring Overall Performance

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 is the measurement from the predicted endpoint to the center of the user's target in pixels (0 pixels is best). Thus, it is independent of target size. Target accuracy is how accurately the predictor is able to correctly identify the user's target. To compare the results from our study with that of Lank et al., we classify target accuracy into one of five groups:

- Correct: The predicted target was the user's intended target.
- -1: The predicted target is the preceding target.
- +1: The predicted target is the following target.
- Negative/Positive Off: The category representing 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 1, 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.

MOUSE-BASED KINEMATIC ENDPOINT PREDICTION

In this section, we present descriptions and results from our studies examining the accuracy of our kinematic endpoint prediction (KEP) algorithm on mouse-based pointing tasks. First, we describe a replication study of Lank et al.'s tablet experiments using a mouse. In the second study, we extend the target width and distance combinations to be more representative of target sizes and distances found in desktop computer interfaces.

Replication Study: Mouse versus Stylus

Before applying the KEP algorithm to a wide range of targets, we first wanted to replicate the Lank et al. study. As mentioned above, since the KEP algorithm is based on models of motion, we hypothesize that accuracies for a mouse would match the published results for a stylus on a tablet computer.

Method

The experiment 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. The Wacom tablet was used because of its high sampling rate. The custom software captured mouse movements at 200Hz. CPU clock time, X position, and Y position were captured for each registered mouse movement.

The task was a discrete, one-dimensional pointing task. As Lank et al. note [12], the goal with any endpoint prediction algorithm is to predict distances. Goal directed movements (i.e. Fitts' style targeting tasks) typically follow straight line or shallow elliptical trajectories, so calculating the primary direction of motion is trivial. Our goal is to, therefore, evaluate how accurately our algorithm can predict distance of movement, and a one-dimensional pointing task is most effective for this task.

Initially a green starting rectangle was displayed on the screen. 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 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 tasks.

The experiment consisted of a within-subjects design with repeated measures. As in the Lank et al. study [12], target distances (D) varied between 200-600 pixels in 100-pixel increments while target size (W) was varied from 15-75 pixels in 15-pixel increments resulting in 25 D/W combinations.

The experiment consisted of two blocks: a practice block and an experimental block. Each block consisted of 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 blocks. The experiment took approximately 40 minutes to complete.

Eight male graduate students participated in the experiment. All participants were right handed.

Of the 1,000 tasks recorded, 3.1% resulted in the user not correctly hitting the target. These tasks were removed from our analysis.

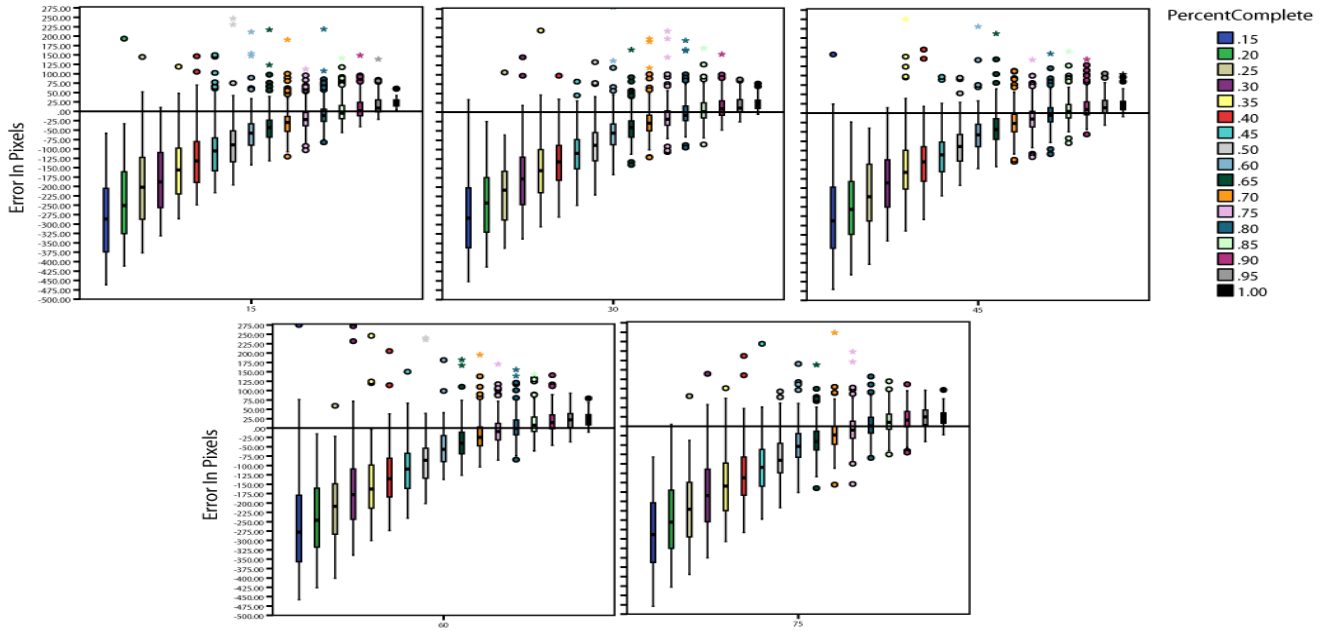

Figure 2. Pixel accuracy of the KEP algorithm along the gesture path for each target width.

Continuous Prediction

Figure 2 shows the pixel accuracy distributions for endpoint predictions by percentage of gesture. Unlike the results from stylus input presented by Lank et al., our KEP algorithm tends to underestimate endpoint distance early in mouse motion because we do not use coefficients. However, as shown in Table 1, target prediction accuracy rates are slightly higher than those of Lank et al. [12].

	Correct	± 1 Target
80% Gesture Length	49.3%	34.1%
85% Gesture Length	51.0%	35.8%
90% Gesture Length	51.4%	36.4%
Lank et al. (80%)	42.4%	39.0%

Table 1: Observed frequencies of predicting the user's target by percentage of gesture length compared to probabilities reported by Lank et al.

Single-shot Prediction

While our results support Lank et al.'s findings, we 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. Using KEP set to a 90% distance threshold, we were able to correctly identify the user's target with 41.9% accuracy and ±1 target with an additional 38.5% accuracy. This accuracy is almost identical to Lank et al.'s stylus accuracy.

As mentioned above, we need to determine when (at what point during motion) predictions can reliably be made. If predictions are occurring after 90% of motion, any benefits from manipulating the user's target may be questionable [13]. As shown in Table 2, 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 24.9% of the time, whereas a threshold of 85% of gesture length results in the highest accuracy identifying the correct target 33.7% of the time and the adjacent target an additional 30.7% of the time.

	Actual Gesture Length			
	Before 90%		**After 90%**	
Threshold	**Correct**	**± 1**	**Correct**	**± 1**
80%	32.8%	33.8%	0%	0.3%
85%	33.7%	30.7%	3.5%	7.9%
90%	24.9%	21.9%	17.0%	16.6%

Table 2: Target accuracies for varying thresholds categorized by percentage of actual gesture distance.

Summary

Using the continuous prediction strategy for endpoint prediction, we observe target accuracy rates significantly better than those of Lank et al. While Lank et al. saw target accuracies of approximately 40% using stylus motion, for mouse motion we are typically predicting the correct target with almost 50% accuracy.

Unfortunately, when we move to single-shot prediction analysis, our performance does degrade. At a threshold of 85% of motion, we are able to correctly predict the target approximately 34% of the time. This represents a significant improvement over McGuffin and Balakrishnan's predictor [13], but is a poorer result than we obtain using continuous prediction.

Distance and Target Effects on KEP Accuracy

Given that our refinement of the KEP algorithm outperforms Lank et al.'s KEP algorithm for mouse movement on their target/distance combinations, we now describe a broader study of target/distance combinations. The purpose of this study is to determine whether our KEP algorithm can effectively predict user endpoint using target/distance combinations typical of modern desktop computer interfaces.

Given that Lank et al. hypothesize that a limit may exist to predictor accuracy (i.e. noting that small, distant targets had poorer accuracy), we also wish to determine whether distance or target width have an effect on endpoint accuracy. Specifically, we are interested in answering the following questions:

- 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 our KEP accuracy?
- What effect does target size have on KEP accuracy?
- Does interaction between target size and distance (i.e. does ID) effect KEP accuracy?

Method

The experiment consisted of a within-subjects design with repeated measures. The independent variables were distance and target width. Target sizes 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 an Index of Difficulty range between 1.87 and 8.59.

The apparatus and task were identical to that of our replication study. The experiment consisted of a practice and experimental block. Each block consisted of the 24 D/W combinations presented to the user 10 times in random order. Between blocks the participant was encouraged to take a break. The experiment took approximately 30 minutes to complete.

Eight subjects participated in the experiment, 7 males and 1 female. All participants where affiliated with the local university.

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

Continuous Prediction

Overall accuracies for predicting the user's target are shown in Table 3. As in the previous two trials, the highest target accuracy occurs at 90% of gesture length and not at 80% as previously reported by Lank et al. [12]. Accuracies, again, approach those of Lank et al. for continuous predictors. At 90% of motion, our KEP predicts target approximately 35% of the time and is off-by-one 25% of the time.

	Correct	**± 1 Target**
80% Gesture Length	28.1%	23.4%
85% Gesture Length	31.1%	23.9%
90% Gesture Length	34.8%	24.7%

Table 3: Observed target accuracy frequencies for continuous prediction.

Analysis of variance of pixel accuracy by target width, and distance shows a significant effect for distance ($F_{2, 1840} = 5.93, p < .001$) and target width ($F_{6, 1836} = 3.61, p < .001$) but not for distance*target width interaction. As a result, we conclude that ID does not have a significant effect on pixel accuracy. Instead target size and distance affect accuracy independently. User ($F_{7, 1835} = 28.40, p < .001$) also has a significant effect on pixel accuracy. Finally, ANOVA also indicates significant effect on pixel accuracy for distance*user interaction ($F_{14, 1828} = 2.55, p < .01$) and target width*user interaction ($F_{28, 1814} = 1.66, p < .001$).

Post-hoc analysis using Bonferroni correction shows a significant difference between 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.

Single-Shot Prediction

Table 4 lists target accuracies for our single-shot predictor with three distance thresholds. As we saw in the previous study, the single-shot predictor target accuracy is less accurate than our continuous prediction strategy, and higher thresholds result in predictions occurring beyond 90% of gesture length. Using our single-shot predictor with 85% gesture length threshold, the user's target was predicted 22.4% of the time and ± 1 target an additional 21.0% of the time.

	Actual Gesture Length			
	Before 90%		After 90%	
Threshold	Correct	± 1	Correct	± 1
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 4: Target accuracy rates for single-shot prediction by threshold and percentage of actual distance traveled when prediction occurred.

Examining effects of distance and target width on target accuracy using the 85% threshold (shown in Table 5), 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. This is especially evident between large target widths at the 512 and 1536-pixel distances.

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 3). Analysis of variance for pixel accuracy at a 85% distance threshold shows a significant effect for distance ($F_{2,\ 1840} = 18.10$, $p < .001$) and user ($F_{7,\ 1835} = 26.87$, $p < .001$), distance*user interaction ($F_{14,\ 1828} = 2.49$, $p < .005$), and target width*user interaction ($F_{42,\ 1880} = 2.24$, $p < .001$). Post-hoc analysis using Bonferroni correction shows a significant difference between all distances ($p < .001$ in all cases). Target width does not have a significant effect on pixel accuracy.

Summary

In our second experiment, we examine 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. Target size definitely has an effect on target accuracy (i.e. our target or ± 1 target accuracies), as 4-pixel targets are very hard to discriminate on a large display with large movements. This effect on target accuracy is shown in Table 5,

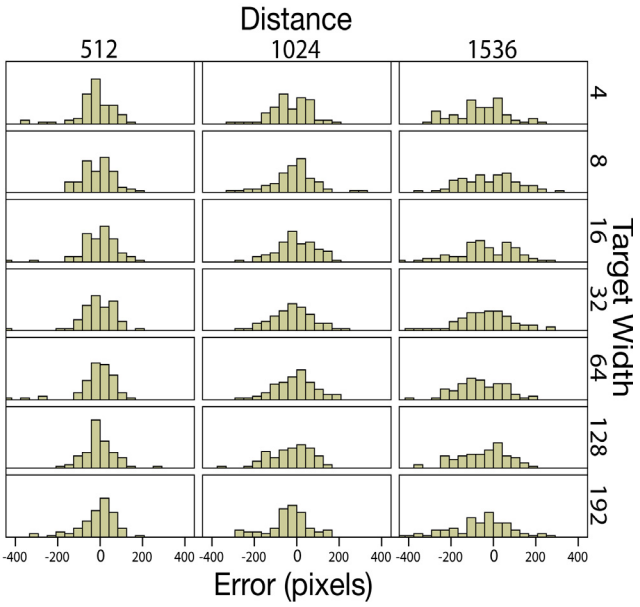

Figure 3: Distribution of pixel accuracy by distance and target width.

where we see low accuracies for 4, 8, and 16-pixel targets. Only when targets reach sizes of 32 and 65 pixels do accuracies improve.

However, when we measure pixel accuracy (Figure 3) we see that distance also has an effect on predictor behavior. In particular, over long distances our pixel accuracy (the distance between predicted endpoint and the center of the user's target) also increases. This broadening of the distribution is easy to see as one moves from 512 to 1536 pixels.

While we would, of course, like higher accuracy, it is important to note that algorithm behavior is still quite good. For 64-pixel targets (i.e. icon-sized) objects, we can identify target with between 16% (at 1536 pixels) and 36% (at 512 pixels) accuracy for single-shot prediction and continuous prediction has even higher accuracy. As well, at 90% of motion these numbers also increase significantly. Given these results, we now discuss our results and examine how our KEP might be used in desktop interfaces.

	Target Distance					
	512		1024		1536	
Target Width	Correct	±1	Correct	±1	Correct	±1
4	1.4%	8.2%	1.3%	1.3%	2.6%	2.6%
8	2.5%	6.3%	3.8%	13.9%	2.5%	1.3%
16	10.4%	14.3%	6.5%	14.3%	0.0%	9.0%
32	14.1%	41.0%	11.5%	30.8%	7.5%	23.8%
64	35.5%	53.9%	28.2%	38.5%	15.6%	33.8%
128	66.3%	32.5%	45.0%	50.0%	47.5%	37.5%
192	85.7%	11.7%	73.4%	24.1%	54.4%	38.0%

Table 5: Target accuracies for our single-shot predictor using an 85% gesture length threshold.

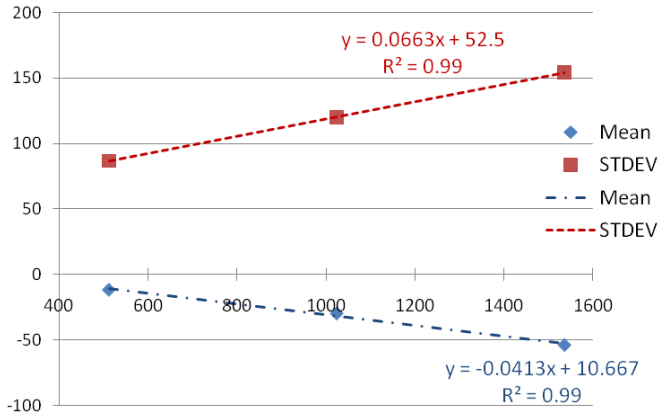

Figure 4: Mean and standard deviation (STDEV) for pixel accuracy by distance.

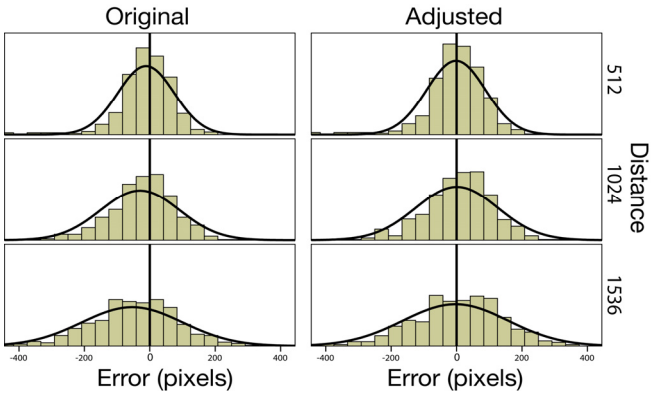

Figure 5: Distributions of pixel accuracy before and after applying the offset calculated using the linear correlation between distance and prediction accuracy mean.

DISCUSSION

The major goals of this research 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 (similar to Lank et al.) 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. 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.

From Theory to Practice

In our results, we take a pessimistic approach to prediction accuracy, arguing that continuous prediction may not be possible, and that post-90% predictions may be unusable. As a result, we may be giving a mistaken impression that the KEP algorithm we develop 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 3), we see that we have 35% accuracy for predicting target, and that we predict ± 1 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 4 illustrates the mean and standard deviation for the single-shot predictor collected in our second experiment. 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, a replication of Rosenbaum's result [16], showing that standard deviation is directly proportional to distance of motion. These numbers also hold for our replication 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 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 and would define a likelihood region around the predicted endpoint. Figure 5 compares these distributions. For example, at a distance of 1536 pixels, the relationships between distance and mean and standard deviation we identify allows us to claim that the user's target will be enclosed by a ± 153 pixel region from the predicted endpoint 68.2% of the time. In summary, because of the linear relationship between distance, pixel accuracy, and

standard deviation of pixel accuracies we can calculate a probability distribution around our predicted endpoint.

The usefulness of this probability distribution is significant. If we have prior probability distributions on the underlying interface, perhaps by modeling command usage or user task, then these two independent distributions can be combined to identify maximally likely targets within a region, i.e. to increase the accuracy of our KEP algorithm, perhaps significantly. As a result, our KEP algorithm can be used as one component of endpoint prediction that uses multiple sources of information.

CONCLUSION

In this paper we presented a modified version of the KEP algorithm to provide a single-shot prediction strategy. We demonstrate that the accuracy for mouse input is similar to that of a stylus and that the ability to predict endpoint using motion kinematics is linked to the length of motion. Through results from our studies we have shown that there exists linear relationship between prediction accuracy and distance, allowing the KEP algorithm to predict an endpoint and calculate a probability distribution around that endpoint.

REFERENCES

1. Asano, T., Sharlin, E., Kitamura, Y., Takashima, K., and Kishino, F. Predictive interaction using the delphian desktop. *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, ACM (2005), 133–141.
2. Balakrishnan, R. "Beating" Fitts' law: virtual enhancements for pointing facilitation. *International Journal of Human-Computer Studies 61*, 6 (2004), 857 - 874.
3. Baudisch, P., Cutrell, E., Robbins, D., et al. Drag-and-Pop and Drag-and-Pick: Techniques for Accessing Remote Screen Content on Touch- and Pen-operated Systems. *Proceedings of Interact 2003*, (2003), 57–64.
4. Blanch, R., Guiard, Y., and Beaudouin-Lafon, M. Semantic pointing: improving target acquisition with control-display ratio adaptation. *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (2004), 519–526.
5. Fitts, P.M. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology 47*, (1954), 381-391.
6. Grossman, T. and Balakrishnan, R. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (2005), 281–290.
7. Guiard, Y., Blanch, R., and Beaudouin-Lafon, M. Object pointing: a complement to bitmap pointing in GUIs. *GI '04: Proceedings of Graphics Interface 2004*, Canadian Human-Computer Communications Society (2004), 9–16.
8. Hogan, N. An organizing principle for a class of voluntary movements. *J. Neurosci. 4*, 11 (1984), 2745-2754.
9. Kabbash, P. and Buxton, W.A.S. The "prince" technique: Fitts' law and selection using area cursors. *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press/Addison-Wesley Publishing Co. (1995), 273–279.
10. Keyson, D.V. Dynamic cursor gain and tactual feedback in the capture of cursor movements. *Ergonomics 40*, 12 (1997), 1287–1298.
11. Kobayashi, M. and Igarashi, T. Ninja cursors: using multiple cursors to assist target acquisition on large screens. *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, ACM (2008), 949–958.
12. Lank, E., Cheng, Y.N., and Ruiz, J. Endpoint prediction using motion kinematics. *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (2007), 637–646.
13. McGuffin, M.J. and Balakrishnan, R. Fitts' law and expanding targets: Experimental studies and designs for user interfaces. *ACM Trans. Comput.-Hum. Interact. 12*, 4 (2005), 388–422.
14. McGuffin, M.J. and Balakrishnan, R. Fitts' law and expanding targets: Experimental studies and designs for user interfaces. *ACM Trans. Comput.-Hum. Interact. 12*, 4 (2005), 388-422.
15. Meyer, D., Smith, J., Kornblum, S., Abrams, R., and Wright, C. Speedaccuracy tradeoffs in aimed movements: Toward a theory of rapid voluntary action. In *Attention and Performance XIII*. 173 - 226.
16. Rosenbaum, D.A. *Human motor control*. Academic Press, San Diego ; Toronto, 1991.
17. Sears, A. and Shneiderman, B. Split menus: effectively using selection frequency to organize menus. *ACM Trans. Comput.-Hum. Interact. 1*, 1 (1994), 27–51.
18. Wobbrock, J.O., Fogarty, J., Liu, S., Kimuro, S., and Harada, S. The angle mouse: target-agnostic dynamic gain adjustment based on angular deviation. *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, ACM (2009), 1401–1410.
19. Worden, A., Walker, N., Bharat, K., and Hudson, S. Making computers easier for older adults to use: area cursors and sticky icons. *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (1997), 266–271.