simplest method for target prediction selects targets simply based on their distance from the cursor position [14].

Work by Murata *et al.* [18] uses a cumulative score for each target as each new movement point is collected**.** At each sampling interval, the system calculates the angle between the movement direction vector and the vector connecting the current position to each potential target. The resulting angle is then added to the cumulative score for each target and the predicted target is selected to be the target with the lowest cumulative score; that is, the target most "on line" with the direction of movement.

The most recent work in target classification contributes two techniques [2,4]. In the first technique, the authors train a neural network on the angle, velocity, and acceleration of a movement to predict whether it has entered the corrective submovement phase. If it has**,** the direction of the movement is calculated and the intended target is chosen to be the nearest target in that direction. In the second technique, the authors use a Kalman filter based on the angle and distance to each potential target. Probabilities are then assigned to each target based on the model and the target with the largest probability is selected. Evaluating the two techniques with both able-bodied and motor-impaired users, target prediction accuracies of about 60% were achieved.

Finally, Ziebart *et al.* [23] provide an approach to attach probabilities to targets using inverse optimal control and Bayes' rule. Their technique first casts pointing as a control problem. Specifically, the authors represent an instantaneous pointing state to be the combined position, velocity, acceleration, and jerk at a given time. They represent the transitions between such states to be based on changes in velocity. Taking this perspective, inverse optimal control techniques are used to create a probabilistic model of pointing movements based on the target locations of previously collected movements. Then, assuming a uniform prior distribution over all targets, Bayes' rule is applied to provide a probability for each target. Results show that approximately 60% target accuracy is achieved when 90% of a movement has been completed. Additionally, the authors achieve higher target accuracies than previous approaches when less than 60% of a movement has been completed.

While the sophistication of the above target classification techniques varies from simple to complex, they all have the disadvantage of being target-aware. By contrast, our approach is target-agnostic, user-adaptable, and easy to implement. By framing the velocity profile of an unfolding movement as a 2-D stroke gesture, we achieve good results while balancing simplicity and sophistication.

## KINEMATIC TEMPLATE MATCHING

Kinematic template matching (KTM) frames the velocity profile of a pointing movement as a 2-D stroke gesture allowing it to be recognized via template matching in a three-step process. First, a library of templates from prior movements is constructed. Second, preprocessing is performed on the templates to prepare them for comparison to a candidate movement, which is preprocessed similarly. Third, the best-matched template is chosen and the predicted endpoint is calculated based on the total distance the matching template movement traveled. To the best of our knowledge, we are the first to conceive of time-series velocity profiles as stroke gestures and to employ template matching for endpoint prediction. The above steps are described in more detail in the following subsections.

### Building the Template Library

A library of templates is built using previously collected pointing movements in order to compare them to future movements a user makes. Each template is created from a distinct sequence of position-time points $(x, y, t)$ that describe a pointing movement; these points are first filtered and then used to produce the template's velocity profile. The movement points, velocity profile, and total distance traveled are stored as part of the template in order to reduce computational complexity while matching candidates.

The predictive accuracy of KTM is dependent on the number of templates in the library. We hypothesized that as the number of templates increased, so would the predictive accuracy. However, empirical evidence revealed that the accuracy did not significantly increase when using more than 1000 templates (Figure 2). Therefore, we found 1000 templates to be sufficient for accurate predictions when target amplitudes were between 100 and 800 pixels, which was the range of pointing movements tested.

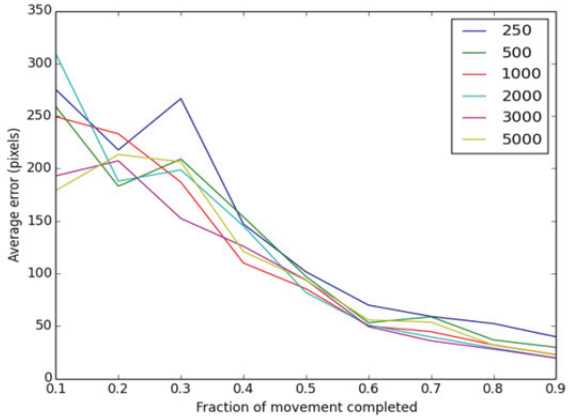


**Figure 2. Average distance of predicted endpoint from observed endpoint in pixels. Lower is better.**

When creating the template library, the goal is to create a mapping that relates velocity profiles to distances traveled. As a result, movements that overshoot their endpoint produce a velocity profile that misrepresents the final distance of a movement and, if matched by KTM, lead to overshooting the candidate movement's endpoint. Therefore, given the sequence of movement points used to build a template for KTM, a simple filter is applied to check

and correct for overshooting. For each movement point $p_i$ in the sequence of points $P = \{p_1, \ldots, p_n\}$, if

$$\|p_i - p_1\|_2 < \|p_{i-1} - p_1\|_2$$

then $P$ is truncated so that $P = \{p_1, \ldots, p_{i-1}\}$. Employing this filter is advantageous to our technique, as it allows KTM to make use of movements with overshoots, which are common when pointing with a mouse [5].

Once the filter has been applied, the points in $P$ are temporally resampled at 20 Hz in preparation for smoothing before comparisons. The resampled points are then used to create the complete velocity-over-time profile of the template. Note that each template needs to only be preprocessed in this way once, not prior to each comparison to a candidate movement.

### Preprocessing Pointing Movements

While a candidate pointing movement is being made, new points are appended in real-time to a cumulative list of position-time points, or $(x, y, t)$ values. As each new point is collected, the list is temporally resampled at 20 Hz and used to produce the candidate movement's *partial* velocity profile. The velocity profile is then smoothed using a 1-D Gaussian kernel filter with a standard deviation of 7 to reduce noise [9] (Figure 3). The chosen resampling rate and standard deviation were based on comparing different resampling rates ranging from 20-200 Hz and different kernel filters with standard deviations ranging from 3-7 (higher values smooth more). The $(20, 7)$ combination resulted in the best predictions overall.

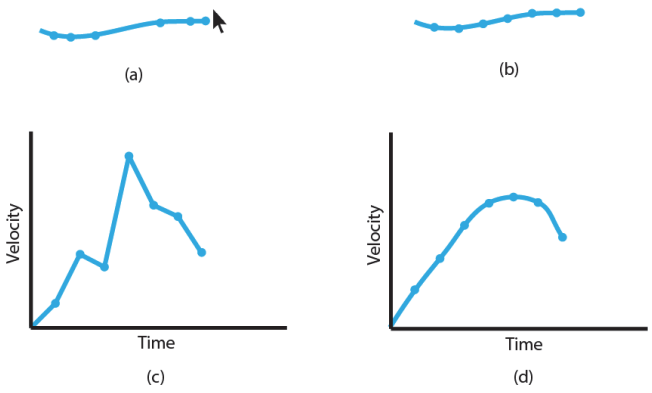


**Figure 3. Stages of preprocessing the candidate movement. (a) The raw movement points are (b) temporally resampled. These resampled points are used to create (c) the velocity profile, which (d) is then smoothed.**

Once the candidate's smoothed velocity profile has been created, the velocity profile of each template also needs to be smoothed. However, when the library is constructed, the velocity profile of each template is representative of a *complete* pointing movement. Conversely, the candidate is still in the process of unfolding and has a velocity profile that reflects movement only up to a certain point in time. To allow for equitable template-candidate comparisons, each template is transformed to reflect its motion at a similar point in time as the candidate. Therefore, before smoothing, if a template's velocity profile ends at a later time than the candidate's, the template's velocity profile is shortened to the same length as the candidate. Alternately, if a template movement is temporally shorter than the candidate, its velocity profile is left unmodified (Figure 4).

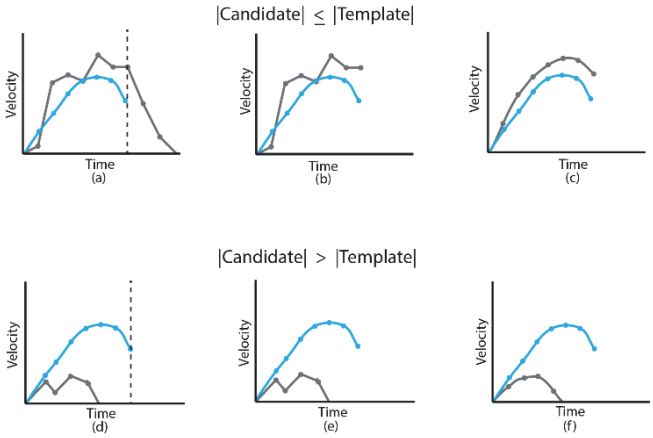


**Figure 4. (a) The gray template's velocity profile is longer than the blue candidate's, so (b) the template's profile is truncated before (c) being smoothed. (d) The gray template's velocity profile is already complete and is (e) not modified before (f) being smoothed.**

It is tempting to consider smoothing all of the template velocity profiles immediately after data collection to reduce computational complexity. However, it is important to note that truncating a template's velocity profile *after* smoothing produces a different profile than *first* truncating the same template's saved movement points and then smoothing. As a pointing movement unfolds, it is only possible to smooth the movement points received thus far. Therefore, it is no surprise that truncating each template's movement points to match the candidate movement's points *before* smoothing leads to better-matched templates and higher accuracies.

### Template Matching and Endpoint Prediction

The candidate movement $C$ is compared to each template $T_i$ in the library at the arrival of each new movement point using the following cumulative scoring function $S(T_i)$:

$$S(T_i) = S(T_i^*) + \begin{cases} \frac{\sum_{j=0}^{n_c} |C_j - T_{ij}|}{n_c}, & n_c \leq n_t \\ \frac{\sum_{j=0}^{n_t} |C_j - T_{ij}| + \sum_{j=n_t+1}^{n_c} C_j}{n_c}, & n_c > n_t \end{cases} \quad (1)$$

where $T_i$ is the $i^{th}$ template in the library; $S(T_i^*)$ is the cumulative score for $T_i$ from all prior movement points considered thus far; $C_j$ and $T_{ij}$ are the $j^{th}$ velocity values from the candidate's and current template's smoothed velocity profiles, respectively; $n_c$ is the number of points in the candidates smoothed velocity profile; and $n_t$ is the number of points in the current template's smoothed velocity profile. Once $S(T_i)$ is computed, it will become the next $S(T_i^*)$ upon the arrival of the next movement point.

In the case that $n_c \leq n_t$, the score is simply the normalized total difference between the velocity values at each timestamp (Figure 5a). However, when $n_c > n_t$, it becomes less likely that selecting $T_i$ as the best-matched template will produce a desirable prediction. In order to negatively weight these shorter templates, the candidate's remaining velocity values are added to increase the score (Figure 5b).

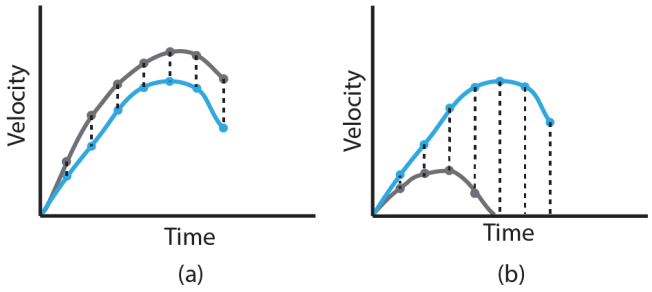


**Figure 5. (a) Comparison of two velocity profiles of equal length. (b) Comparison of two profiles of unequal length.**

We found using a cumulative scoring function to be crucial in the success of KTM. As stated, once $S(T_i)$ is computed in Equation 1, it becomes the next $S(T_i^*)$ upon the arrival of the next movement point. The resampled velocity profile of a candidate movement changes as each new point is added and, in most cases, a template's smoothed velocity profile will change as well. Assuming that movements that unfold similarly over time cover similar distances, a cumulative score effectively assigns higher weight to templates that consistently scored well over the arrival of each new candidate movement point.

Once the candidate has been compared to the entire template library, the template with the lowest cumulative score is chosen as the best-matched template. Finally, the total as-the-crow-flies length of the best-matched template ($d_t$) is calculated and the candidate movement's endpoint is predicted to be $d_t$ pixels away from its original start point in the current direction of motion.

### Summary

The complete process for predicting the endpoint of a candidate movement in real-time is as follows:

For each new movement point added to the candidate:

1. Temporally resample the points of the candidate movement at 20 Hz;
2. Construct the velocity-over-time profile of the candidate by taking the derivative of the temporally resampled points;
3. Smooth the newly constructed velocity profile using a 1-D Gaussian Kernel filter with a standard deviation of 7;
4. For each template, $T_i$, in the library:
    a. If necessary, truncate the velocity profile of $T_i$ as shown in figure 4;
    b. Smooth the velocity profile of $T_i$;
    c. Compare the candidate $C$ to $T_i$ using Eq. 1.
5. Select the template with the lowest cumulative score and add its final length to the candidate's original start point in the current direction of motion.

We now turn our focus to evaluating how well KTM works to predict endpoints of mouse movements, comparing KTM to the revised kinematic endpoint prediction (KEP) algorithm [19] in the process.

## EXPERIMENT

We ran a study to evaluate the predictive accuracy of kinematic template matching (KTM) in both one-dimensional and two-dimensional pointing tasks.

### Method

#### *Participants*

Seventeen able-bodied participants (13 males, 4 females) participated in our study with an average age of 25.0 years (*SD*=7.3). All but two participants were right-handed and when asked to self-rate their computer proficiency (0-10, with 0 being lowest), results ranged from 4 to 10 (*M*=7.4, *SD*=1.8). Subjects were given a small payment for participating in the study.

#### *Apparatus*

We created a custom C# application to administer the tasks, log data, and calculate results. Our application ran on a 27-inch Apple iMac desktop running Windows 7 64-bit displaying a screen resolution of 2560 × 1440. The computer was equipped with a 2.7 GHz Intel i5 processor, 8 GB of RAM and a Microsoft Basic Optical Mouse v2.0, which was used as the input device.

#### *Procedure*

Participants were asked to complete two sets of 1100 trials, where a "trial" consisted of clicking a single target. Targets were sequentially displayed on the screen in blocks of 21 trials. Each trial displayed only one target on the screen and consisted of a single click, which initiated the next trial. If the target was missed, a *ding!* sound was played and the trial was logged as an error. Subjects were asked to click the targets "as quickly and accurately as possible" and were allowed to take a break between blocks to avoid fatigue. Each set of 1100 trials took approximately 20 minutes to complete.

The first set used vertical ribbon targets 32 pixels wide to collect data on 1-D pointing movements. The first target of each block was placed in the center of the screen and data for this trial was not recorded. A random distance between 100 and 800 pixels was generated at the start of each consecutive trial and the new target was placed this distance away from the previous target. Once all blocks had been completed, participants were encouraged to take a break before starting the second task.

The second set used circular targets 32 pixels in diameter to capture 2-D pointing movements. The targets of each block were displayed one at a time using a layout similar to the ISO 9241-9 ring-of-circles arrangement. The first target of each block was chosen to be the top-most circle in the ring.

However, distances from one target to the next were randomly set between 100 and 800 pixels.

For both sets of trials, the goal of using *random* distances between targets was to generate sufficient data for a continuous range of target amplitudes, and to more closely simulate pointing in the real world.

*Analysis*

We created two separate template libraries for each of the 17 subjects: one using data from the 1-D task and one using the data from the 2-D task. The template libraries were evaluated independently of each other. The 1-D and 2-D trials had average error rates of 7.1% and 6.3% respectively; trials marked as errors were nevertheless included in the evaluation of KTM.

The evaluation of KTM proceeded as follows. First, a template was chosen at random from the template library and removed. The points from the selected template were appended to a new list, one at a time, which was used to simulate a candidate movement. The overshoot filter was *not* used on the selected template in order to produce a natural, unaltered movement path. The KTM algorithm was used to predict the final distance of the simulated candidate each time a new point was added. Each predicted distance was compared to the movement's ground truth distance to calculate the accuracy of a prediction. After the final prediction was made, the selected template was admitted back into the library and a new template was chosen at random. This process was repeated 100 times for each template library.

To compare the predictive accuracy of KTM to the current state of the art in *target-agnostic* prediction, we also implemented the revised kinematic endpoint prediction (KEP) algorithm of Ruiz *et al*. [19]. KEP was evaluated using the same data as we used to evaluate KTM; however, trials marked with errors were not used, as per KEP's formulation. Movements were selected from the log files and fed pointwise through KEP. Distances were predicted as each point was added, which were then compared to the ground-truth distance of the given movement. KEP performance was tested for both 1-D and 2-D tasks.

The predictive accuracy of both KTM and KEP was evaluated at movement-distance-percentage intervals from 10% to 90% in 10% increments. Although we evaluate KTM in isolation based on percent time complete, we compare KTM to KEP using percent distance complete because with so few collected points at early times in a movement, KEP tends to return predictions with extremely high error rates, which is a side effect of extrapolating. We felt, therefore, it was fairer to KEP to report results based on percent distance complete, which was calculated by dividing the distance traveled at the moment of evaluation by the total distance of the completed movement.

**Results**

*1-D Pointing Task*

Figure 6 shows a box-and-whisker plot of the predictive accuracy of KTM at different percentages of horizontal 1-D movements. Boxes contain the upper and lower quartiles of the error values and whiskers extend to the most extreme data point within 25%-75% of the error range. Predictions are generally centered around the true endpoint (zero on the *y*-axis), but do seem slightly more prone to overshoot rather than undershoot, particularly in the 20%-40% range (the mean is a little above zero). Overall, the mean of the box-plot centers is +32.4 (*SD*=23.6). That the spread looks to be well balanced around the center confirms that our prediction approach using kinematic template matching is not systematically biased with respect to the true endpoint.

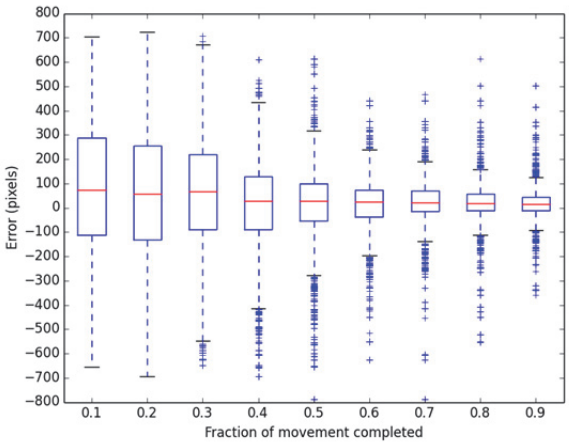


**Figure 6. Distribution of KTM prediction errors in pixels from true endpoint based on 1-D pointing trials. Fraction of movement complete is based on percent time.**

As it can be seen from Figure 7, in the case of one-dimensional horizontal movements, KTM predicts with significantly better accuracy than KEP for the first 40% of distance traveled. When more than 40% of a movement has been completed, KTM and KEP preform similarly, although KEP slightly outperforms KTM during the last 20%.

Not surprisingly, both KTM and KEP reach their peak predictive performance when approximately 90% of a movement has been completed and predict, on average, to within 69 and 51 pixels of the true distance, respectively. Additionally, when we examine KTM at 90% of completed movement duration (*i.e.*, percent time), we see predictions within 48 pixels of the true endpoint.

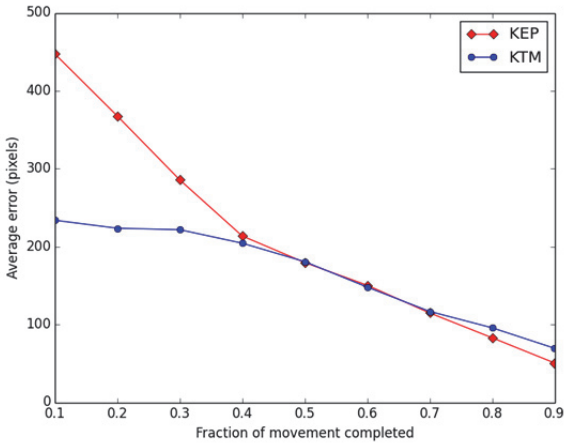


**Figure 7. The predictive accuracy of KTM compared to KEP for 1-D pointing trials. Fraction of movement complete is based on percent distance.**

Importantly, both KTM and KEP are target-agnostic and not using any knowledge of target locations or dimensions to predict endpoints. That said, given that 1-D targets of 32 pixels in width were used to collect this data, we can show how often the predicted endpoint actually landed within the target. Table 1 below shows the percentage of times 1-D targets would have been hit for both KTM and KEP at each movement distance increment. The hit-rate for KTM is significantly higher than KEP according to a Wilcoxon signed-rank test ($Z$=-2.07, $p$<.05).

| Percentage of Distance Traveled | Target Hit Rate (KTM) | Target Hit Rate (KEP) |
|---|---|---|
| 50% | 3.3% | 4.0% |
| 60% | 5.0% | 3.9% |
| 70% | 8.1% | 4.9% |
| 80% | 11.3% | 6.6% |
| 90% | 18.8% | 13.33% |

**Table 1. Target hit rates for KTM and KEP for 1-D trials.**

*2-D Pointing Task*

Figure 8 shows a box-and-whisker plot of the predictive accuracy of KTM at different percentages through 2-D movements. The boxes and whiskers show the same ranges of values as displayed in Figure 6 for the 1-D task. Similar to the 1-D task evaluation, KTM seems to overshoot the endpoint more often than fall short of it, but the spreads still look to be well balanced around the center (zero on the $y$-axis). Overall, the mean of the box-plot centers is +19.5 ($SD$=8.8).

Additionally, the distribution of 2-D prediction errors is much tighter than that of the 1-D task. In the 2-D task, the average box height is 151.5, whereas in the 1-D task, it is 197.9. These differences in the distribution of undershoots and overshoots between 1-D and 2-D tasks were significant ($Z$=-2.55, $p$<.02). This finding indicates that KTM is generally more accurate in 2-D than in 1-D.

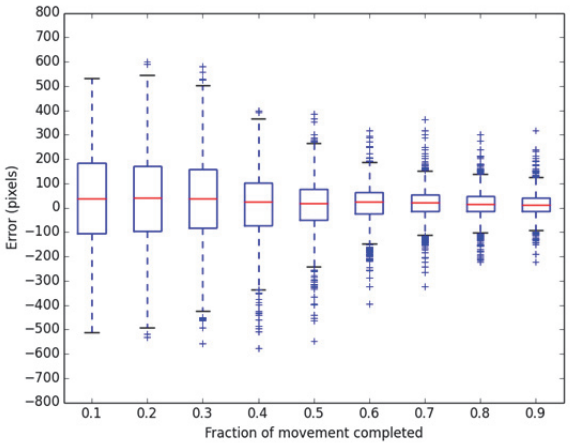


**Figure 8. Distribution of KTM prediction errors in pixels from true endpoint based on 2-D pointing trials. Fraction of movement complete is based on percent time.**

Both KTM and KEP perform more accurately overall for 2-D pointing tasks than they did for 1-D pointing tasks. KTM predicts with lower errors throughout most of the movement, although KEP is slightly more accurate when movements have traveled approximately 90% of their total distance (Figure 9). Again, both approaches predict with increasing accuracy throughout and reach their peak accuracy at 90%. At this point, KTM on average predicts within 55 pixels of the true endpoint and KEP predicts within 47 pixels. As for 1-D movement, because of the large advantage of KTM in the first half of movement, KTM is significantly more accurate than KEP overall ($Z$=-2.43, $p$<.02). Additionally, when we examine KTM at 90% of completed movement duration (*i.e.*, percent time), we see predictions within 39 pixels of the true endpoint.

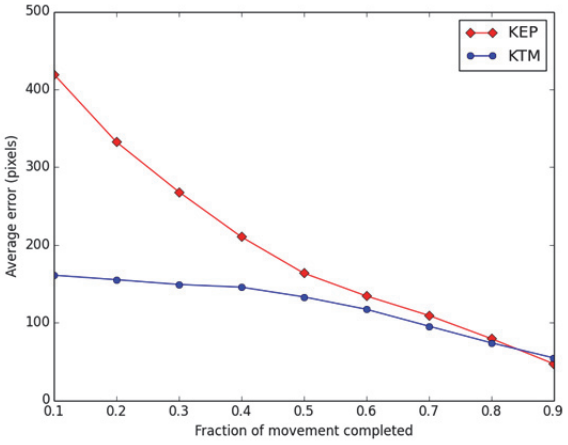


**Figure 9. The predictive accuracy of KTM compared to KEP for 2-D pointing trials. Fraction of movement complete is based on percent distance.**

Again, we can examine how often predictions from KTM and KEP would actually hit the 32 pixel circular targets used as stimuli in data collection. Table 2 below shows the

percentage of times 2-D targets would have been hit for both KTM and KEP at each movement distance increment. The hit-rate for KTM is again significantly higher than KEP according to a Wilcoxon signed-rank test ($Z$=-2.67, $p$<.01). These accuracies are notably lower than those reported by Ruiz *et al.* [19]. We discuss possible reasons for this in our discussion, below.

| Percentage of Distance Traveled | Target Hit Rate (KTM) | Target Hit Rate (KEP) |
|---|---|---|
| 50% | 5.6% | 3.1% |
| 60% | 6.6% | 4.1% |
| 70% | 10.8% | 4.2% |
| 80% | 14.6% | 4.8% |
| 90% | 21.5% | 9.2% |

**Table 2. The target accuracy of KTM and KEP for 2-D trials.**

Clearly, these hit rates themselves are not particularly high, but it is important to remember that target-agnostic endpoint predictions can be useful for pointing facilitation even when not directly hitting the target. For example, attractive gravity can be put at any screen location while pointing, and even if that gravity is outside by nearby the intend target, it could still facilitate quicker target acquisition, especially if it knew to "turn off" upon seeing a submovement correction at odds with its attractive force. Also, prior work [16] has shown that predictions even as late as 90% through a movement can be beneficial to pointing performance.

## DISCUSSION

Our study shows that kinematic template matching (KTM) is an effective technique for predicting the endpoints of aimed pointing movements. KTM performs with higher accuracy compared to the revised kinematic endpoint prediction (KEP) algorithm [19]—the current state of the art for target agnostic techniques. Although both approaches predict with similar pixel accuracy at many percentage intervals for 1-D trials, KTM significantly outperforms KEP at nearly all percentage intervals for 2-D trials, in terms of both pixel distance and target accuracy. Additionally, KTM performs particularly well in comparison to KEP during the early stages of pointing movements for both 1-D and 2-D trials. Although the target accuracies reported are not as impressive as prior approaches, we believe this is mainly due to evaluating KTM using a set target size of 32 pixels as compared to the variable, and larger, target sizes used in other studies.

As stated above, the target accuracies we report based on our implementation of the revised KEP algorithm are lower than those reported by Ruiz *et al.* [19]. We attribute this discrepancy to be mainly due to differences in methodology. First, Ruiz *et al.* report percentages of movements by dividing the current distance traveled by the predicted distance. Second, Ruiz *et al.* use a range of target sizes in their experiment (15-75 pixels in 15 pixel steps) and average accuracies across these target sizes. As a test, we evaluated KEP using their reported method for calculating percent complete and found that target accuracy more than doubled for our data. However, we believe that our method of evaluation is fairer, as it is based on ground truth instead of predicted values, which are sometimes quite different from the true distance.

The backbone of KTM is the library of prior movements it compares to as templates. While endpoint prediction techniques could be developed that are accurate but computationally expensive, the ability to make predictions in real-time is crucial for deployment in interactive settings. Because KTM exhaustively searches through the library to find the best-matched template, increasing the number of templates in the library negatively affects the execution time. So, when developing the algorithm, we were curious to find what library size resulted in the best size/speed tradeoff. We expected accuracies to increase as libraries grew in size, but it was very interesting to find that after a certain number of templates—1000 in this case—accuracies did not significantly increase (Figure 2). Again, our chosen size of 1000 worked well for the given target amplitude range of 100-800 pixels, however we would expect this number would have to increase with a larger range of target amplitudes. Although past approaches such as KEP are potentially advantageous in that no "setup" is required, we find that no-setup approaches are either based on theoretical laws of human motion, or are too simple; when presented with non-normative movements, they perform poorly. Using a template library, on the other hand, offers levels of personalization that prior target agnostic approaches do not.

When developing KTM, we originally left movements with overshoots unaltered. However, during our initial testing, we noticed that KTM over-predicted many of the endpoints. While it occurred to us that the submovements made post-overshoot could potentially be used to improve the accuracy of KTM, we settled on filtering out these submovements. We felt that adding a special case to the KTM algorithm to utilize these submovements would unduly increase the complexity of the algorithm and detract from implementation ease. Also, based on prior work [16] pointing facilitation techniques benefit the user only if less than 90% of a pointing movement has been completed, but overshoots are usually closer than this. Therefore, we were skeptical that such a modification to the KTM algorithm would be useful.

We found developing the scoring function to be a particularly challenging issue. Although the scoring function's effect on prediction time was not as significant as the size of the template library, it was something to keep in mind throughout the process. While our final implementation uses velocity values alone, we explored using additional movement profiles, such as acceleration and jerk to score templates. However, these additional profiles did not add to our predictive power, and velocity profiles were sufficient in isolation. Although KTM is a naïve algorithm, it still performs with sufficient speed for

interactive use. While heuristics could be added to optimize KTM, doing so would increase the complexity of implementing KTM. However, if optimizing KTM was ever a necessity, peak velocity could potentially be used as a simple heuristic filter, as work by Takagi *et al.* [21] revealed a relationship between peak velocity and final distance. Ease of implementation is a cornerstone of our approach and, with this in mind, we are impressed by the predictive accuracy that can be achieved using the velocity profile of movements alone.

## FUTURE WORK

Kinematic template matching provides an excellent basis for further research in the area of predictive pointing. First, while the data used for this paper was collected using a custom test-bed application in a controlled lab environment, we postulate that pointing movements collected "in the wild" using a system such as the *Input Observer* [8] would perform comparably well. Integration with such a system would ease the burden of collecting a template library, as it could be collected from daily computer use instead of from a specialized application. Furthermore, the ability to evaluate KTM with data extracted from actual pointing movements could potentially lead to design insights that we are currently unable to perceive using data gathered from controlled experiments. Second, KTM could be used in conjunction with proposed pointing facilitation techniques, such as attractive gravity wells [11] to speed target acquisition. While pointing facilitation techniques are commonly used as motivations in many prior endpoint prediction papers, high prediction accuracy is of high importance when it comes to making these techniques usable. KTM outperforms prior techniques in terms of accuracy, making it an important stepping stone to developing an endpoint prediction technique that is deployable in a real-world system. Additionally, KTM is unique in that it is not based on regression yet remains target-agnostic; we see KTM as being particularly well-suited to situations where targets are out of physical reach from the user. For instance Baudisch *et al.'s* "Drag-and-Pop" technique [3] for acquiring distant targets on wall-sized displays could potentially be improved using KTM. Finally, because templates are user-specific, we believe KTM could provide a more beneficial approach for people with non-standard pointing abilities compared to a technique such as KEP, which is based on a normative model.

## CONCLUSION

Kinematic template matching treats the velocity profiles of pointing movements as 2-D stroke gestures and employs template matching to predict movement endpoints. This approach is user-specific, target-agnostic, and is both easier to implement and more accurate than prior techniques. We found that on average, KTM is able to predict within 83 pixels of the true endpoint when 50% of the movement has been completed, 48 pixels at 75%, and within 39 pixels at 90%, using 1000 templates per participant. To the best of our knowledge, our work is the first to conceive of time-series velocity profiles as stroke gestures in 2-D space, and to utilize stroke gesture template matching as a means of prediction the endpoints of aimed pointing movements. It is our hope that this work will pave the way for the development and deployment of real-world pointing facilitation techniques based on endpoint prediction.

## ACKNOWLEDGEMENTS

The authors thank James Fogarty, Mayank Goel, and Krzysztof Z. Gajos for early discussions of this work. This work was supported in part by the National Science Foundation under grant IIS-0952786. Any opinions, findings, conclusions or recommendations expressed in this work are those of the authors and do not necessarily reflect those of the National Science Foundation.

## REFERENCES

[1] Asano, T., Sharlin, E., Kitamura, Y., Takashima, K. and Kishino, F. (2005). Predictive interaction using the Delphian Desktop. *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '05)*. Seattle, Washington (October 23-27, 2005). New York: ACM Press, 133-141.

[2] Aydemir, G.A., Langdon, P.M. and Godsill, S. (2013). User target intention recognition from cursor position using Kalman filter. *Proceedings of the 7th Int'l Conference on Universal Access in Human-Computer Interaction (UAHCI '13)*. Las Vegas, Nevada (July 21-26, 2013). Lecture Notes in Computer Science, vol. 8009. Berlin, Germany: Springer-Verlag, 419-426.

[3] Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P., Bederson, B. and Zierlinger, A. (2003). Drag-and-Pop and Drag-and-Pick: Techniques for accessing remote screen content on touch- and pen-operated systems. *Proceedings of the 9th IFIP TC13 Int'l Conference on Human-Computer Interaction (INTERACT '03)*. Zurich, Switzerland (September 1-5, 2003). Amsterdam, The Netherlands: IOS Press, pp. 57-64.

[4] Biswas, P., Aydemir, G.A., Langdon, P. and Godsill, S. (2013). Intent recognition using neural networks and Kalman filters. *Proceedings of Human-Computer Interaction and Knowledge Discovery (HCI-KDD '13)*. Maribor, Slovenia (July 1-3, 2013). Lecture Notes in Computer Science 7947. Berlin, Germany: Springer-Verlag, 112-123.

[5] Casiez, G., Vogel, D., Balakrishnan, R. and Cockburn, A. (2008). The impact of control-display gain on user performance in pointing tasks. *Human-Computer Interaction 23* (3), 215-250.

[6] Chang, C.-h.J., Amick, B.C., Menendez, C.C., Katz, J.N., Johnson, P.W., Robertson, M. and Dennerlein, J.T. (2007). Daily computer usage correlated with undergraduate students' musculoskeletal symptoms. *American Journal of Industrial Medicine 50* (6), 481-488.

[7] Dixon, M., Fogarty, J. and Wobbrock, J.O. (2012). A general-purpose target-aware pointing enhancement using pixel-level analysis of graphical interfaces. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '12)*. Austin, Texas (May 5-10, 2012). New York: ACM Press, 3167-3176.

[8] Evans, A. and Wobbrock, J.O. (2012). Taming wild behavior: The Input Observer for obtaining text entry and mouse pointing measures from everyday computer use. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '12)*. Austin, Texas (May 5-10, 2012). New York: ACM Press, 1947-1956.

[9] Fisher, R., Perkins, S., Walker, A. and Wolfart, E. (2003). Gaussian smoothing. *Hypermedia Image Processing Reference*. Edinburgh, Scotland: University of Edinburgh.

[10] Flash, T. and Hogan, N. (1985). The coordination of arm movements: An experimentally confirmed mathematical model. *The Journal of Neuroscience 5* (7), 1688-1703.

[11] Hwang, F., Keates, S., Langdon, P. and Clarkson, P.J. (2003). Multiple haptic targets for motion-impaired computer users. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '03)*. Ft. Lauderdale, Florida (April 5-10, 2003). New York: ACM Press, 41-48.

[12] Johnson, P.W., Dropkin, J., Hewes, J. and Rempel, D.M. (1993). Office ergonomics: Motion analysis of computer mouse usage. *Proceedings of the American Industrial Hygiene Conference and Exposition (AIHCE '93)*. New Orleans, Louisiana (May 15-21, 1993). Falls Church, Virginia: American Industrial Hygiene Association, 12-13.

[13] Keuning-Van Oirschot, H. and Houtsma, A.J.M. (2001). Cursor displacement and velocity profiles for targets in various locations. *Proceedings of EuroHaptics 2001*. Birmingham, United Kingdom (July 1-4, 2001). Paris, France: EuroHaptics Society, 108-112.

[14] Lane, D.M., Peres, S.C., Sandor, A. and Napier, H.A. (2005). A process for anticipating and executing icon selection in graphical user interfaces. *International Journal of Human-Computer Interaction 19* (2), 241-252.

[15] Lank, E., Cheng, Y.-C.N. and Ruiz, J. (2007). Endpoint prediction using motion kinematics. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '07)*. San Jose, California (April 28-May 3, 2007). New York: ACM Press, 637-646.

[16] McGuffin, M. and Balakrishnan, R. (2002). Acquisition of expanding targets. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '02)*. Minneapolis, Minnesota (April 20-25, 2002). New York: ACM Press, 57-64.

[17] Mikkelsen, S., Vilstrup, I., Lassen, C.F., Kryger, A.I., Thomsen, J.F. and Andersen, J.H. (2007). Validity of questionnaire self-reports on computer, mouse and keyboard usage during a four-week period. *Occupational and Environmental Medicine 64* (8), 541-547.

[18] Murata, A. (1998). Improvement of pointing time by predicting targets in pointing with a PC mouse. *International Journal of Human-Computer Interaction 10* (1), 23-32.

[19] Ruiz, J. and Lank, E. (2009). Effects of target size and distance on kinematic endpoint prediction. *Technical Report CS-2009-25*. Cheriton School of Computer Science. Waterloo, Ontario: University of Waterloo.

[20] Ruiz, J. and Lank, E. (2010). Speeding pointing in tiled widgets: Understanding the effects of target expansion and misprediction. *Proceedings of the ACM Conference on Intelligent User Interfaces (IUI '10)*. Hong Kong, China (February 7-10, 2010). New York: ACM Press, 229-238.

[21] Takagi, R., Kitamura, Y., Naito, S. and Kishino, F. (2002). A fundamental study on error-corrective feedback movement in a positioning task *Proceedings of the 5th Asia-Pacific Conference on Computer-Human Interaction (APCHI '02)*. Beijing, China (November 1-4, 2002). Beijing, China: Science Press, 160-172.

[22] Wobbrock, J.O., Fogarty, J., Liu, S.-Y., Kimuro, S. and Harada, S. (2009). The Angle Mouse: Target-agnostic dynamic gain adjustment based on angular deviation. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '09)*. Boston, Massachusetts (April 4-9, 2009). New York: ACM Press, 1401-1410.

[23] Ziebart, B.D., Dey, A.K. and Bagnell, J.A. (2012). Probabilistic pointing target prediction via inverse optimal control. *Proceedings of the ACM Conference on Intelligent User Interfaces (IUI '12)*. Lisbon, Portugal (February 14-17, 2012). New York: ACM Press, 1-10.