Head-Coupled Kinematic Template Matching: A Prediction Model for Ray Pointing in VR

Rorik Henrikson* **Tovi Grossman†** **Sean Trowbridge‡** **Daniel Wigdor*†** **Hrvoje Benko‡**

*Chatham Labs
Toronto, ON
{rorik | daniel}@chathamlabs.com

†University of Toronto
Toronto, ON
{tovi | daniel}@dgp.toronto.edu

‡Facebook Reality Labs
Redmond, Washington
{stro, benko}@fb.com

ABSTRACT

This paper presents a new technique to predict the ray pointer landing position for selection movements in virtual reality (VR) environments. The technique adapts and extends a prior 2D kinematic template matching method to VR environments where ray pointers are used for selection. It builds on the insight that the kinematics of a controller *and* Head-Mounted Display (HMD) can be used to predict the ray's final landing position and angle. An initial study provides evidence that the motion of the head is a key input channel for improving prediction models. A second study validates this technique across a continuous range of distances, angles, and target sizes. On average, the technique's predictions were within 7.3° of the true landing position when 50% of the way through the movement and within 3.4° when 90%. Furthermore, compared to a direct extension of Kinematic Template Matching, which only uses controller movement, this head-coupled approach increases prediction accuracy by a factor of 1.8x when 40% of the way through the movement.

Author Keywords

Endpoint prediction; target prediction; virtual reality; VR; kinematics; ray pointing; template matching

CSS Concepts

• **Human-centered computing~Human computer interaction (HCI)**; Interactive paradigms; Virtual reality

INTRODUCTION

In the last several years, there has been a significant increase in the popularity of virtual reality (VR) technologies. Despite decades of research in the HCI community, many interaction challenges are still prevalent, and interface paradigms have yet to converge. Specifically, target selection via pointing, one of the core tasks in VR systems [35], remains problematic due to the spatial nature of VR environments.

One of the most common selection techniques to use in VR is a ray pointer, which acts like a laser emanating from a 6-DOF controller, and can be used to acquire distant targets. Ray pointers, however, are error prone as when targets are far away, they require high angular precision for successful acquisition. Numerous techniques have been developed to facilitate ray pointing, but they often introduce additional steps [22, 29] or input mechanisms [7]. To better leverage pointing facilitation techniques, it would be advantageous if a system could predict the target or region that a user intends to point towards, while the movement is still in progress.

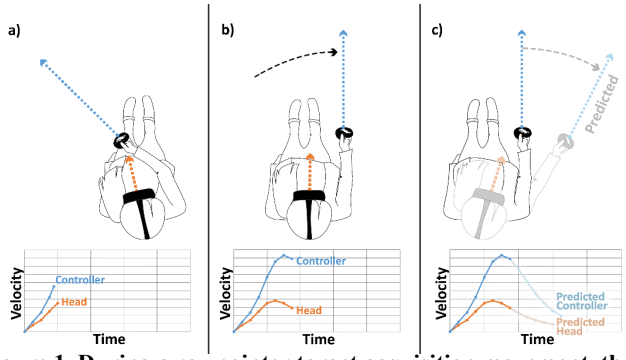

Figure 1. During a ray pointer target acquisition movement, the velocity of both the controller and the head mounted display are tracked (a, b). These velocity profiles are matched to a library of templates to predict the final ray landing position (c).

In 2D environments, many endpoint predictive models have been developed that could be used to facilitate pointing tasks [33, 45, 53, 63]. With such models, the cursor trajectory is continuously analyzed as it moves towards an intended target, and the model predicts where the final endpoint of the trajectory will be. One recent, promising technique is Kinematic Template Matching (KTM), which matches cursor velocity profiles to a library of templates from known movements to predict the trajectory's endpoint [45]. Despite the promise of such techniques, little work has applied endpoint prediction to VR environments.

This paper describes a novel endpoint prediction model, *Head-Coupled Kinematic Template Matching* (*HC-KTM)*, for ray pointing in VR (Figure 1). It builds upon 2D Kinematic Template Matching, with important adaptations and enhancements. Most notably, the prior model only considers the cursor trajectory to build and match template gestures. Our key insight is to integrate head movement trajectories into the templates – an information channel inherently available from the Head-Mounted Displays (HMD) of VR platforms. This allows predictions to be based on where users look, along with the pointer trajectory.

Two data collection experiments were performed to validate

CHI '20, April 25–30, 2020, Honolulu, HI, USA

ACM 978-1-4503-6708-0/20/04…$15.00
https://doi.org/10.1145/3313831.3376489

the efficacy of this new model. The first experiment analyzed acquisition behaviors to build and tune the model under controlled conditions. It revealed that during the first half of a movement, head trajectory data is more indicative of the final landing position than controller trajectory data. The second experiment validates the *HC-KTM* model by capturing target acquisition data across a continuous range of angles and target sizes. The prediction model was then applied to the captured data, and its accuracy analyzed. Results show that our new model's predictions are within 7.3° of the true landing position, 50% of the way through the movement, and within 3.4° at 90%. Furthermore, compared to a direct extension of KTM, the head-coupled approach increases prediction accuracy by a factor of 1.8x at 40% of the way through the movement.

After describing the results of our studies, we discuss two potential ways in which the predictive model could potentially be utilized to improve VR user experiences. First, pointing facilitation techniques could be enhanced by biasing towards the model's predicted region. Second, we discuss broader applications, beyond pointing facilitation, such as haptic retargeting [12], foveated rendering [2], and latency reduction techniques [61], which could all be improved if a user's intentions could be inferred prior to the associated actions occurring.

The key contributions of this work are (1) the adaptation of KTM endpoint prediction to VR environments, (2) a head-coupled kinematic template matching (HC-KTM) algorithm that integrates velocity data from both the controller and the head, and (3) empirical data showing that this new technique can predict ray pointer landing positions and outperform more direct adaptations of prior approaches.

RELATED WORK

This work builds upon prior research in VR selection, distant pointing on large displays, cursor endpoint prediction, and gaze-based user interfaces.

VR Selection Techniques

Numerous techniques have been developed to facilitate pointing in 3D environments (for a comprehensive survey, see the work of Argelaguet & Andujar [3]). Early research identified two main classes of selection techniques - ray pointing or ray casting [17, 34, 47] and 3D cursors (i.e., virtual hand) [18, 24, 35, 48]. Facilitation techniques have been developed for 3D cursors [48, 57], but ray-based approaches have proved to be more efficient [22].

With ray pointing, a cursor emanates like a laser pointer from the hand or a 6-DOF handheld controller. Despite its prominence in current VR platforms, it is difficult to select small and distant targets due to the angular accuracy needed. One general solution is to use a conic area for the ray [18, 35], however, this can require disambiguation techniques if multiple targets are in the selection region.

The use of predictive heuristics can enable one to automatically differentiate between multiple selected targets. This can be done by highlighting the closest target [34], having the last intersected object persist until a new object is intersected [56], or by using more weighting schemes that continuously update as the cursor moves [14, 22, 55]. Another method to disambiguate between targets is to provide an explicit mechanism for users to indicate a target of interest. Techniques such as using a secondary radial menu [22, 48], supporting progressive refinement using a series of quad-menus [30], or explicitly controlling a marker along the depth of the ray [6, 22, 51], have all been proposed.

Although these techniques improve the final selection process, the literature is lacking in ways that can predict the landing position of a ray pointer while moving.

Distant Pointing on Large Displays

Ray (or virtual laser) pointing is also a common selection metaphor to use when interacting at a distance with large displays [25, 28, 40]. To address the performance detriments associated with the required angular accuracy, researchers have explored techniques to increase precision, such as adapting the CD ratio based on cursor velocity [28].

Another approach leverages a technique where ray pointing is first used for coarse positioning and then for precise positioning [59]. Nancel et al. thoroughly explore the design space of such "dual-precision" techniques [41, 42]. Most related to our work, they propose a "dual-channel" technique where head orientation provides coarse control of the cursor and a handheld device handles precise positioning [42]. We extend this approach by instead leveraging the head-movement to inform a predictive model of the ray pointer landing position while the controller is still in motion.

Cursor Endpoint Prediction

In desktop configurations, significant efforts have been made to develop *endpoint prediction* techniques while the mouse is still in motion. Three main approaches have been proposed in the literature: regression-based extrapolation, target classification, and kinematic template matching.

With regression-based extrapolation, existing models of cursor movement behaviors are used to predict the location of a distant target based on partial movements [3, 25]. Most successful is Lank et al.'s motion kinematics approach [33], which they subsequently improved to take into account the stability of the prediction [53].

An alternative approach is to use target classification, which integrates knowledge about targets in the environment to identify the most probable candidate target [39]. Recent work has shown that Neural Networks and Kalman filters can be used to predict user intent based on the kinematics of the cursor [5, 8]. Ziebart et al. assigned probabilities to targets, using inverse optimal control and Bayes' rule [63]. While such techniques are promising, they are complex and require knowledge of the target locations.

A final approach is *Kinematic Template Matching* [45]. With this technique, the velocity profile of a partial pointing

movement is compared to a library of known "template" movements to predict the final cursor location. Although untested, the authors suggest that this prediction could then be combined with target selection techniques such as target expansion [20] or gravity wells [9]. Template matching offers a number of advantages over the other techniques: it is target-agnostic, user-adaptable, and easy to implement [45]. As such, this work builds upon this approach and will be reviewed later in more detail.

Outside the domain of 2D cursors, target prediction for touch-based interfaces has been explored, often to reduce perceived latency [11, 43, 44]. For example, Xia et al. leveraged hover information to predict the time and location of a touch just before it occurred [61]. Ahmad et al. applied similar predictive models for in-car, mid-air selection [1]. In VR, LaViola explored filtering techniques for making predictions to reduce the perceived latency in VR environments [34]. These techniques have also been used to predict saccade landing positions to reduce the latency of foveated rendering in head-mounted displays [2].

Despite the promise of these techniques, we are unaware of work that applies prediction models within the realm of VR ray pointing to predict selection intent.

Gaze and Head Input

Gaze is an established method to provide input to interactive systems, in particular for contexts which require hands-free operations [15]. Gaze can also be used as an implicit channel to detect users' intent [23]. For example, the MAGIC technique leverages gaze information to warp the cursor to a general area of interest [16, 62]. Similar efforts have also been made in 3D environments, such as combining gaze with dwell or pinch [13, 46, 58]. However, gaze can suffer from the Midas touch problem, and can be slower than a hand-controlled ray pointer [13].

Cassallas et al. demonstrated that by coupling head and hand movement features, one can predict intended moving targets [10]. Kyto et al. showed that gaze and head pointing could be combined with refinement to support the precise selection of 2D targets in AR [31]. In automotive user interfaces, Roider and Gross showed that gaze data could be used to improve the accuracy of pointing gestures [52]. In VR, Cheng et al. presented a technique for predicting user intention by analyzing gaze and hand movement, so that the hand can be redirected towards a physical proxy [12].

Such work demonstrates the benefit of coupling hand-controlled pointing with gaze or head movements. Our goal is to build upon such work by using head movement data to predict the landing position of a ray pointer movement.

REVIEW OF 2D KINEMATIC TEMPLATE MATCHING

The technique proposed in this paper is inspired by Pasqual and Wobbrock's Kinematic Template Matching (KTM) technique, one of the more accurate endpoint prediction techniques in the literature [45]. It also has the practical advantages of being both target-agnostic and simple to implement. The concept behind KTM is to utilize a cursor's velocity profile as a 2D stroke gesture, allowing it to be recognized using a template matching algorithm (Figure 2). The approach uses a four-step process: building a template library, preprocessing new candidate pointing movements, matching the template, and estimating the cursor endpoint.

In the first step, a library of templates is generated using a collection of previous pointing movements. Each template consists of (i) a cursor velocity curve as it progresses towards the target and (ii) the total distance travelled. The velocity profiles are truncated to remove overshoots and resampled to 20 Hz. It is important to note that their technique compares a user's movement to the user's own template library. This allows results to be personalized to individual's pointing behaviors, at the cost of requiring individual data collection.

The next step occurs when a new pointing movement is being made (i.e., the *candidate movement*). In real time, the velocity profile is resampled to 20 Hz and smoothed using a Gaussian filter. To prepare for template matching, each template in the library is truncated to match the duration of the candidate movement, and then the same smoothing is then applied. Note that the smoothing of the templates happens *after* they are truncated, which was found to lead to better matched templates and higher accuracy.

Once preprocessing is completed, the candidate movement is compared to each library template. This comparison occurs when each new candidate movement point arrives. A cumulative scoring function compares the candidate movement to the template:

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

where T_i is the i[th] template in the library; $S(T_i^*)$ is the prior calculated score; C_j and T_{ij} are the j[th] velocity values from the candidate and current template smoothed velocity profiles, respectively; and n_c and n_t are the number of points in the candidate and current template smoothed velocity profiles, respectively. Once the candidate movement has been compared to all templates, the template with the lowest score is selected as the best match.

Finally, to predict the candidate movement's final endpoint, the technique uses the travel distance associated with the best matched template and applies that distance to the current direction of the candidate's movement from the original start point. As with other endpoint prediction techniques, the accuracy of KTM improves as the candidate movement progresses towards the target. It was found that on average, KTM predicts the endpoint location within 83 pixels of the true endpoint when 50% of the movement has been completed, 48 pixels at 75%, and 39 pixels at 90%. This performance was better than prior kinematic endpoint prediction methods [33].

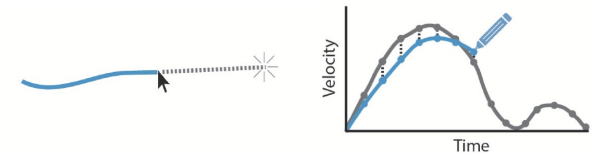

Figure 2. The KTM approach considers the velocity profile of a movement as a 2D gesture. A partial candidate movement is compared to a known template, and the endpoint is inferred. Figure taken from [45]

HEAD-COUPLED KINEMATIC TEMPLATE MATCHING

To adapt the KTM approach for ray pointing in VR environments, we consider a traditional ray pointer which acts like a virtual laser pointer. The virtual ray has 5 degrees of freedom: the user must specify the origin (*X, Y, Z*) and direction (θ_h, θ_v) of the ray. The goal of our technique is to predict the final controller location and direction while the pointer movement is still in progress. We adapt and extend the KTM technique in three important ways:

1) The KTM method was built for 2D cursor pointing, predicting the (*X, Y*) coordinates of the movement's endpoint. To adapt the technique for 3D ray pointing, we are not predicting an "end point" per se, rather the final landing position of a ray. Thus, estimates of not only the 3D coordinates of the handheld controller, but also the angle at which the ray is being emitted, are needed.

2) The KTM method only considers the velocity profiles of the cursor in the template matching procedure. Thus, a cursor's velocity profile across targets with different distances may not be distinguishable in the first part of its movement [19, 26]. We extend this method to consider user head movement, hypothesizing that this additional channel may increase prediction accuracy. We call this head-coupled variation *HC-KTM*.

3) The KTM method selects one, best matching template, to estimate the endpoint distance. We extend the method to a *top-n* approach, where the weighted average of multiple matching templates may be used, allowing for the compensation of a poorly matching individual template. We call this top-n variation *KTM-N*.

Together, these 3 enhancements form the basis of our Head-Coupled Kinematic Template Matching (HC-KTM-N) technique, which maintains the desirable properties of KTM. The technique is target-agnostic, straight-forward to implement, and can be personalized to individual users. The approach follows the same 4 general steps of KTM, described below.

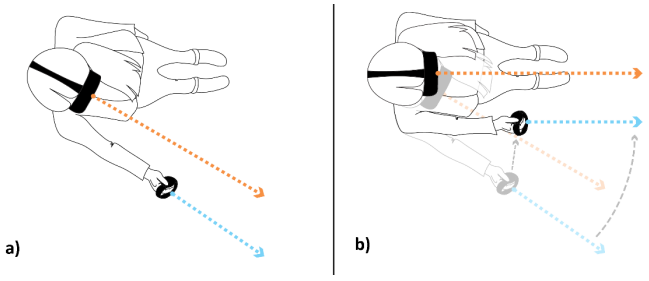

Figure 3. Top view of a ray pointer acquisition movement. Both the head and controller change in position and angle.

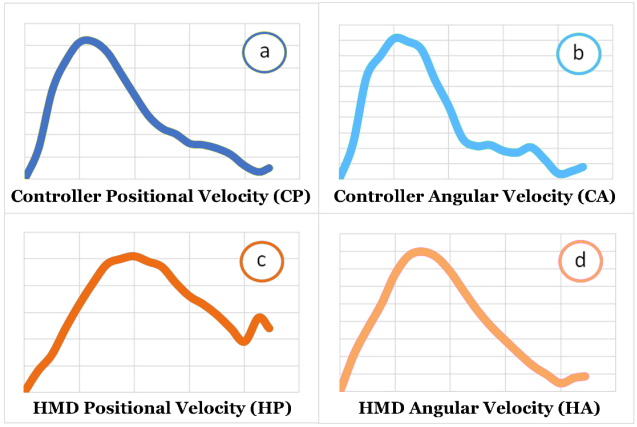

Figure 4. Each template consists of four velocities: a) Controller positional velocity. b) Controller angular velocity. c) Head positional velocity. d) Head angular velocity.

Step 1. Building the Template Library

The template library is built by capturing selection movements for known targets, considering both the motion of the controller *and* the head during selection. Because these are spatial input channels, we must consider both the location and the angle of the controller and head (Figure 3). As such, each template consists of *four* velocity profiles (Figure 4):

- *CP (mm/s)*: Controller positional velocity - the change in the controller's (*X*, *Y*, *Z*) origin coordinates over time.
- *CA (deg/s)*: Controller angular velocity - the change in angle of the controller's forward-facing vector over time.
- *HP (mm/s)*: Head positional velocity - the change in the HMD's (*X*, *Y*, *Z*) origin coordinates over time.
- *HA (deg/s)*: Head angular velocity - the change in angle of the HMD's forward-facing vector over time.

In the previous movement angle distance KTM technique, the template library was modified to crop any backtracking from a template; initial testing found that adequate results were achieved without performing this step and it was omitted. Unlike KTM, we found it necessary to do an initial smoothing of the templates, given the noise introduced by midair 6-DOF devices. A Gaussian smoothing operation was performed on each of the velocities using a 5-point window. The profiles were then resampled to 20 Hz in preparation for comparison to subsequent candidate movements.

Step 2. Preprocessing Candidate Pointing Movements

As a new candidate movement is captured, the position and angle values of the head and controller are collected. They are used to create the four partial velocity profiles, which are smoothed using a 5-point Gaussian window and resampled to 20 Hz as each new point is collected. As with KTM, each velocity profile in the template library that is longer in duration than the candidate movement is truncated to the same length as the candidate movement.

Step 3. Matching Candidate Movements

The candidate movement, *C,* is then compared to each template, T_i, at the arrival of each new movement point using the same cumulative *controller* scoring function presented in Equation 1. However, in this case, the scoring calculation is repeated four times, once for each of the four velocity profiles, resulting in four scores (i.e., S_{cp}, S_{ca}, S_{hp}, and S_{ha}), that correspond to the velocity profiles *CP*, *CA*, *HP*, and *HA*.

The final cumulative scoring function, *S(Ti)*, is defined as a weighted sum of the four individual scores:

$$S(T_i) = aS_{cp}(T_i) + bS_{ca}(T_i) + cS_{hp}(T_i) + dS_{ha}(T_i) \quad (2)$$

Where, *a*, *b*, *c*, and *d* are tuning parameters. Note that by setting *a*, *c*, and *d* to 0, the model reduces to the KTM model and uses only the velocity profile of the controller angle.

Step 4. Calculating the Expected Landing Position

Unlike KTM, which considers only top matching templates, the *n*-best template matches are ranked by the minimum values of $S(T_i)$. To calculate the expected final *movement angle distance* of the ray, a weighted average of the movement angle distances of the top-*n* templates is computed. The template's weight, w_i, is defined as the reciprocal of $S(T_i)$, and the template's movement angle distance as d_i. Using these values, the weighted average angular distance is calculated:

$$\mu = \frac{\sum_{i=1}^{n}(w_i * d_i)}{\sum_{i=1}^{n} w_i} \quad (3)$$

Using this weighted average angular distance (μ), the controller's initial *angle* is rotated by the magnitude of μ, along the current angle of motion. The same approach is used to calculate the expected controller *location*. Using the weighted average of the top-*n* template's controller distances, the magnitude of this average is added to the initial controller's position along the current direction of movement. By combining the expected *angle* and *location*, the final ray pointer landing position is calculated (Figure 5).

EXPERIMENT 1 – HUMAN POINTING BEHAVIOR

The first experiment gathered initial data to better understand human head and hand behaviors during ray pointer movements using controlled target distances and sizes. The

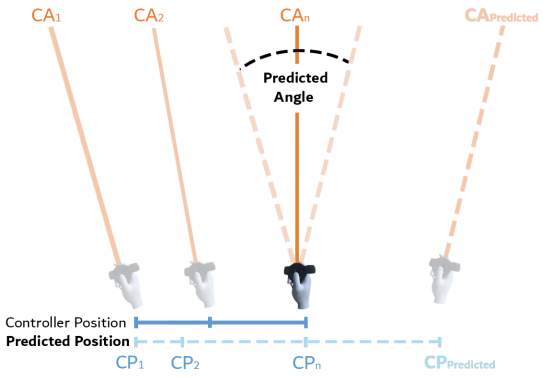

Figure 5. To predict the final landing position of the ray, the prediction for the final angle and position of the controller are combined.

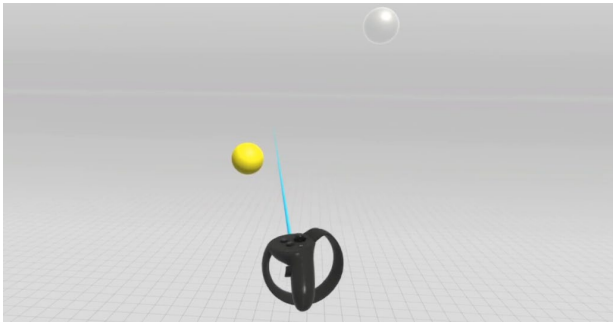

Figure 6. First person view of the study environment.

experiment consisted of a pointing task in a VR environment, using a ray pointer, without prediction enabled.

Participants

Seventeen participants (11 female), with no major motor impairments and normal or corrected-to-normal vision (only contact lenses were allowed) were recruited. They ranged in age from 18 to 26 (M=20.9, SD=2.0). Participants were compensated $30 for their time. A Randot Stereo Optical Test was administered prior to the experiment to ensure adequate stereo vision. All participants were right-handed and operated the controller with their right hand.

Apparatus

The experiment was conducted using an Oculus Rift CV1 head-mounted display, with a resolution of 2160×1200, using a single Oculus Touch handheld controller for input. The Index Trigger button was used for selection. The position and angle of the HMD and controller was tracked using Oculus Constellation Sensors. The system ran on a 3.7 GHz Intel Core i7-8700k desktop computer with an NVIDIA GeForce RTX2080 graphics card and was developed in Unity3D. The HMD output updated at a frequency of 90 Hz, and both the HMD and controller positions and angles were updated at a rate of 90 Hz. The handheld controller manipulated a ray pointer using an absolute mapping, with the ray originating from the tip of the controller, aligned with the z-axis of the local handheld controller coordinate system.

Procedure

The task was a reciprocal three-dimensional pointing task, wherein participants pointed back and forth, in succession, between a start and end target. No distractor targets were included. The target to be selected was yellow, and the other was semi-transparent gray. The background of the scene was a gray gradient, and in the virtual environment, subjects stood on an elevated platform above an infinite grid ground plane. The two targets were rendered as spheres (Figure 6).

The goal target turned green when it was intersected by the ray, to indicate the target could be selected. Upon successful selection, the targets swapped colors. If the ray did not intersect the goal target when a button-click occurred, the trial counted as an error, and the participant tried again until successful. Subjects were asked to complete the task as quickly as possible, without exceeding an error rate of 4%. The error rate was displayed after each block of trials.

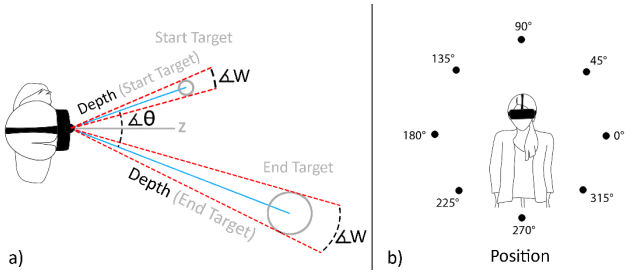

Figure 7. The target layout. a) Reciprocal targets were located on opposite sides of the z-axis at varying depths, with equal angular widths. b) Targets appeared at one of 8 angles.

During the study, participants stood on a marked floor position and were asked not to move their feet. The software and the experimenter ensured their feet were in the proper position prior to each trial. To calibrate, prior to the study, the coordinate system was reset with the participant on a marked spot with the HMD in a resting state. The experimenter could recalibrate at any time during the study, if needed. The point between the eyes was the origin, with the positive axes being: left to right (X), bottom to top (Y), and back to front (Z). Before each session, participants performed practice trials to become familiar with the task – lasting about 2 minutes.

Design

A repeated-measures, within-participant design was used. The position of the goal target varied based on three independent variables – *Depth (3m, 6m, 9m), Theta (25°, 50°, 75°),* and *Position (0°-315° at 45° increments)* (Figure 7). *Depth* manipulated the distance between the target center and the origin. *Theta* changed the magnitude of the angle between the vectors generated by connecting each target to the origin, with the center vector of these two vectors laying along the Z-axis. For each combination of *Theta* and *Depth,* there was a ring of possible target locations (i.e., *Position*), evenly distributed at 45° increments (Figure 7b). In each reciprocal task, targets were placed in opposite locations of the ring, but their depth values could vary (Figure 7a).

A target's size was determined by its angular width, *W* (*4.5°, 9.0°*), relative to the origin. With *W* fixed, the further the targets the larger the radius; but the angle needed to place the ray within its boundaries remained fixed. In each reciprocal task, the angular width of both targets was equal (Figure 7a).

The experiment was done in one, approximately 60 minute, session. The study had 54 blocks for each of the 54 possible combinations of *Depth* (start target), *Depth* (end target), *Theta*, and *W* in random order. Participants could take breaks between blocks to prevent fatigue. For each block, 4 sets of reciprocal trials were performed for the eight *Positions* (i.e., 4 pairs), and consisted of nine clicks (i.e., 8 reciprocal selections between the two targets at opposite positions). This resulted in 54 x 4 x 8 = 1728 trials per participant.

Results and Analysis

Prior to all analysis, outliers (trials where time was more than two standard deviations in length compared to trials with the same *Theta* and *W)* were removed; which was 6% of the data. All remaining trials were analyzed, however, trials where errors occurred (1.7%) were only analyzed to the first click.

Movement Time

Movement time was defined as the duration between selecting the start target and the next subsequent selection, regardless of the success of the selection. A repeated measures analysis of variance showed main effects of *Width* ($F_{1, 16}$ = 581.5, $p < .0001$) and *Theta* ($F_{2, 32}$ = 693.4, $p < .0001$). Also, significant was the interaction between *Width* and *Theta* ($F_{2, 32}$ = 7.7, $p < .005$). Target depth (*Depth*) did not have a significant effect on movement time ($F_{2, 32}$ = 2.8, ns).

Interestingly, across the 6 combinations of *Theta* and *Width*, the performance trend followed an angular derivation of Fitts's Law [17, 27], with an extremely high fit (R^2 = 0.993) (Figure 8):

$$MT = a + b \log_2 \left(\frac{Theta}{Width} + 1 \right)$$

While such an angular derivation has been examined previously for 2D large display interactions, our results contrast prior work. In particular, Jota et al. achieved a fit of only 0.61 for horizontal pointing and 0.33 for vertical pointing on large displays [25]. Kopper et al. achieved a higher fit of 0.96 [29], but used a more complex angular model (ID_{DP}), which resulted in a fit of only 0.74 with our data. While our data provides an initial validation of Fitts' Law for ray pointing in VR environments, this contrast to prior work warrants future investigation.

Head and Controller Angular Movements

When measuring the cumulative angular distance that the controller and HMD travelled, both the HMD Angle ($F_{2, 32}$ = 272.4, $p < .0001$) and Controller Angle ($F_{2, 32}$ = 116673, $p < .0001$) were significantly impacted by *Theta.* For HMD angle, there was a significant effect of both *Width* ($F_{1, 16}$ = 79.1, $p < .0001$) and *Position* ($F_{7, 112}$ = 110.1, $p < .0001$). These effects were also observed for *Width* ($F_{1, 16}$ = 18.2, $p < .005$) and *Position* ($F_{7, 112}$ = 7.6, $p < .0001$) on the Controller Angle. Notably, the HMD moved further for smaller targets – likely, to better see smaller targets. This shows that for a predictive model, head movement may provide information not apparent from just the controller movement. *Depth* did not have a significant effect on HMD or Controller Angle.

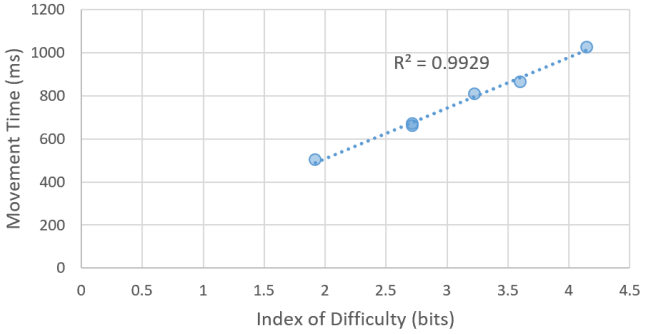

Figure 8. The effect of width and angular distance on movement time results in a high degree of fit with an angular derivation of Fitts' Law.

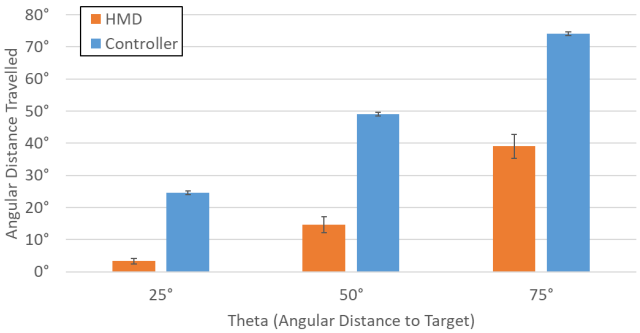

Figure 9. The HMD and Controller angular movements, with respect to the Angular Distance to the target. Error bars represent standard error.

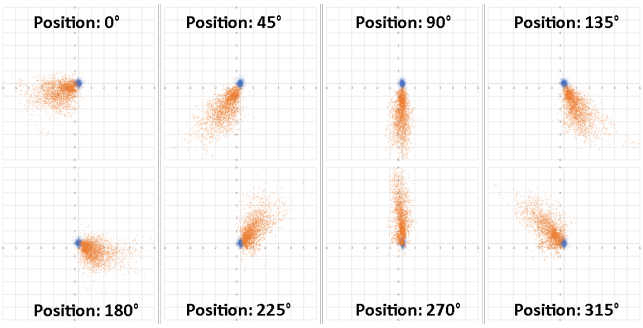

Figure 10. A scatter plot of the final controller (blue) and HMD (orange) locations, for each position. Points are the intersection with a plane centered at the target location.

Comparing the magnitude of the angular movement between the two devices, the HMD moved a fraction of the angle that the controller moved ($F_{1, 16} = 422.0$, $p < .0001$) (Figure 9). This is intuitive – the controller angle is required to move within the bounds of the goal target, while the HMD is required to move just enough so that the target is in the user's field of view. This is further illustrated in Figure 10, which shows a scatterplot of the final focal point of the HMD and Controller when each target was acquired. It can be seen that the controller focus (blue) was within the bounds of the target, while the HMD only travelled a fraction of the required distance, with observable variation between trials.

Head and Controller Velocity Profiles

In addition to looking at the angular movements of the HMD and controller, it is also interesting to observed how velocity profile templates differ across different movement angles. To generate "representative" velocity profiles, the velocity profiles were resampled to 20 Hz, and the average velocity at each interval was computed (Figure 11). Consistent with prior findings [19, 26], the velocity profiles for the handheld controller were not distinguishable in the initial stage of movement across the three movement angles. However, the velocity profiles for the HMD diverge immediately. This suggests that incorporating head movements within a predictive model may allow predictions to be made earlier.

Model Parameters and Performance

To analyze the performance of our model and tune its parameters, intra-participant template libraries were created from each user's trials. With outliers removed, each movement was compared against the other templates in the user's template library. To calculate accuracy for individual predictions, the angular distance between the predicted ray and the ray from the controller origin through the center of the intended target was computed. This angular accuracy informed the precision of the model, similar to pixel distance used in 2D models. Since the data collection experiment only used three discrete angular distances, it served as an initial testbed of the model – if the model worked as expected, candidate motions would match templates with the same angle of movement.

The model had two main factors to manipulate: how many templates to use in the *top-n* matching algorithm and the best values to assign to the weights of the scoring algorithm for the four velocity profiles (Eqn. 2). To determine n for the *top-n* matching templates, four equally weighted components were used (i.e., $a, b, c, d = 1$), and the cumulative accuracy of all trials, of all participants, for different values of n was calculated. Results improved gradually from n=1 to 4, whereas $n = 4$ to 10 were similar, with $n = 7$ providing the best result, which is the value used for our implementation.

To determine the best weighting values for the scoring function (a, b, c and d - Eqn. 2), various values were tried to optimize the model's accuracy at 40% of the movement progress, prioritizing early prediction. The importance of each individual component is shown in Figure 12, showing the average angular accuracy when each component was used individually. The data suggests the HMD angle may provide a better indicator for the first half of the movement, while the

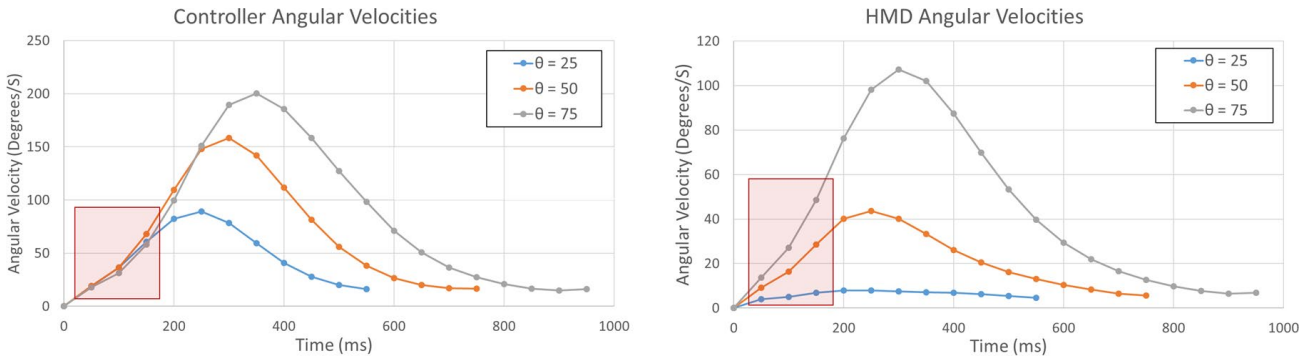

Figure 11. Representative angular velocity profiles for the HMD and Controller by movement angle. The highlighted regions illustrate that in the first 150 ms of movement, profiles only diverge for the HMD, not the controller.

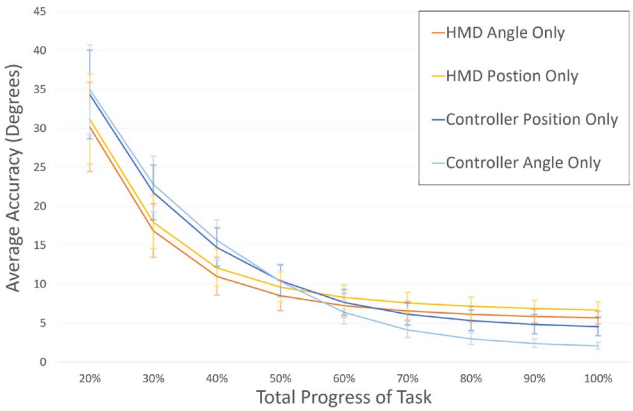

Figure 12. The prediction accuracy derived from each input channel at different stages of the movement.

controller angle may be a more accurate predictor by the end. This may be due to the velocity profiles of the HMD being more distinguishable in the first phase of movement (Figure 11). This is an important result, as it demonstrates the potential benefits of incorporating data from head movements. By considering the relative performance of each input channel, setting the best at 1 and worst at 0.5, and interpolating the remain two values, we derived values of $a = 0.95$, $b = 0.5$, $c = 0.86$, and $d = 1$. We call this specific model HC-KTM-7, for "Head-Coupled KTM" with $n = 7$.

For this controlled study, the calculated parameter values resulted in average accuracies of 6.4°, 3.3°, and 2.3° at 50%, 70%, and 90% of the way through the task, respectively. While promising, it is important to note that this should only be considered initial validation, as the test set only used three discrete angular distances. The next study validates the HC-KTM-7 model with a continuous range of target conditions.

EXPERIMENT 2 – MODEL VALIDATION

A second experiment was conducted to further validate the model and compare it to baseline approaches. Experiment 1 only examined three movement angles where this second experiment evaluates the robustness of the model against a continuous range of angles, depths, positions, and widths. Testing the model in a second experiment helps avoid the potential over-fitting of model parameters, which were set using data gathered in Experiment 1. Again, this study collected movement behaviors without any prediction enabled.

Participants

A total of 12 participants (9 female) completed Experiment 2 and were recruited from the pool of 17 participants in Experiment 1. Participants were compensated $30.

Apparatus and Procedure

The same apparatus and procedure that was used in Experiment 1 was used in Experiment 2.

Design

A repeated measures within-participant design was used. As with the first experiment, the task was a reciprocal three-dimensional pointing task with no distractor targets.

The only controlled variable was *Theta*, which tested all angles from 15° to 85° at 1-degree intervals. All other task variables were randomized. The *Depth* of both targets ranged continuously from 3m to 9m, *Position* ranged from 0° to 359°, and angular width (*W*) ranged from 4.5° to 9.0°.

The experiment was performed in one session lasting approximately sixty minutes, with each angle of *Theta* repeated five times in random order. To prevent fatigue and to allow for breaks, the experiment was divided into 50 equal sections; each section presented 7 pairs of targets with randomized *Theta* values, requiring 6 reciprocal selections. This resulted in 50 x 7 x 6 = 2100 trials per participant. Before each session, participants were given practice trials to familiarize themselves with the task, lasting about 2 minutes.

Results

All trials were analyzed, however, trials where errors occurred (2.7%) were only analyzed up until the first click.

To validate our model (*HC-KTM-7*), it was compared to a direct adaptation of the kinematic template matching (*KTM*) model. Recall, our model varies from KTM in two ways: a weighted average of the top 7 matching templates is used, and a head-coupled approach is utilized, using 4 velocity profiles instead of one. As such, we provide a comparison to *KTM-7* (KTM, n=7) and *HC-KTM-1* (KTM, with head coupling), to understand the relative benefit of these two variations. As in prior endpoint prediction work [63], the results are also compared to a *Baseline* that utilizes the current position of the ray without any prediction applied.

The results indicate that HC-KTM-7 performs better than the KTM technique, and the other 2 intermediate variants (KTM-7, HC-KTM-1; Figure 13). This is encouraging, as it shows both of the proposed enhancements can increase the accuracy of predictions. Compared to the Baseline with no prediction, HC-KTM-7 improved accuracy by a factor of 2.7x at 40% progress. However, near the end of the movement, the baseline seems to be most accurate. This is consistent with prior findings [63] and suggests that adaptive techniques that can detect when a movement is ending are promising.

It is interesting that HC-KTM-1 outperforms KTM-7, indicating that the improvement of our technique primarily stems from using a head-coupled approach. The differences in accuracy are most pronounced at 40% progress, with accuracies of 10.0° for HC-KTM-7, 11.6° for HC-KTM-1, 15.0° for KTM-7, 17.9° for KTM, and 26.4° for the Baseline. Overall, our new model, HC-KTM-7 provided promising results. Prediction accuracy was 7.3° at 50% of the way through the task, 4.4° at 70%, and 3.4° at 90%. When looking at the target hit rate, i.e., how often predictions fell within the bounds of the goal target, HC-KTM-7 outperforms the basic KTM condition, achieving an average hit rate of 46.2%, 70% of the way through a movement, compared to 37.6% (Table 1). While these predictions can still be improved, they are on par with prior work which used a similar protocol [33, 53], but are also dependent on the width of the goal targets.

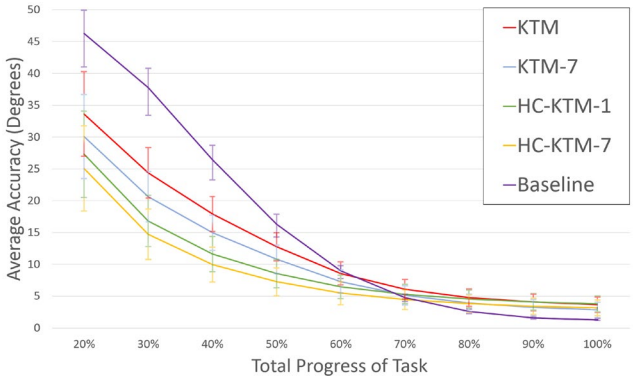

Figure 13. Accuracy curves for the Baseline and 4 variants of the model, i.e., HC-KTM-7, HC-KTM-1, KTM-7, KTM.

Furthermore, to utilize the model, a target hit is only required to predict the intended target if it is in a densely populated area. Thus, there may be a novel opportunity to combine selection refinement techniques [21, 30] with this predictive model in some instances. Alternatively, the model could also be used to inform interface layouts, ensuring that adequate spacing between targets in VR environments is provided.

To examine the necessity of personalized template libraries, accuracies were calculated for each participant when using other user's template libraries (Table 2). In all cases, matching against the participant's own template library, achieved the best accuracies. The average result when comparing to other templates is still within 3° (9.96° vs. 12.94°). It is interesting to note that that personalized data may be more important for some users (e.g. P3) likely depending on the uniqueness of their pointing behaviors. For example, the angular head movements of P3 were only 29.2% of the average across all other participants.

DISCUSSION AND FUTURE WORK

The presented model shows promise for VR ray-pointer predictions. By modifying the KTM model and introducing head-coupling and considering the *top-n* matches, the HC-KTM-7 model's predictions were 1.8x and 2.7x more accurate than KTM and the baseline respectively, 40% of the way through a user's movements.

Although developed and evaluated within a 3D VR environment, the model should generalize to 2D platforms. Targets were shown in 3D space, but the task itself could be decomposed into the 2D angular movements of the ray pointer. Specifically, it would be interesting to use the model for distant pointing on large, high resolution displays, where 2D angular ray pointing is also used. Within this context, the present work can be seen as building on prior literature which has coupled head and hand movements to divide large display pointing into coarse and precise modes [42]. Notably, while the model can be applied to 2D tasks, it still requires and uses 3D input channels. In particular, the model must estimate the final 3D position of the controller (Figure 5). The 3D movements of the hand can be substantial – on average the hand moved 18.7cm, and as much as 45cm in some trials (average maximum across all participants). If the

	Target Hit Rate			
Distance Travelled	**KTM (n = 1)**	**KTM-7 (n = 7)**	**HC-KTM-1 (n = 1)**	**HC-KTM-7 (n = 7)**
50%	15.1%	14.8%	12.2%	22.7%
60%	26.1%	28.0%	18.4%	34.8%
70%	37.6%	42.9%	23.8%	46.2%
80%	47.8%	56.2%	28.5%	55.6%
90%	55.2%	65.8%	31.8%	61.8%

Table 1. The percentage of predictcions where the predicted ray hits the intended target; for 4 main conditions.

model were only based on 2D angular data of the ray, then the predictions would be less accurate. Fortunately, most virtual ray pointing controllers provide full 6-DOF input.

Of course, HC-KTM-7 is only one possible predictive model. Comparing our results to similar efforts such as regression-based extrapolation and target classification could yield interesting results. Future models could include probability distributions across possible targets to influence predictions.

Limitations

One aspect of 3D VR pointing that was not addressed, is that multiple targets can be located along the same projected path at varying depths, requiring disambiguation. The HC-KTM model only predicts the location of the ray, not the depth of the target. While the study varied the target depth, the analysis did not show any effect from depth on pointing behaviors, making this a challenging problem for prediction. Future work should explore additional input channels, such as gaze [60], to predict object depth and extend the model into a truly 3D prediction. Selection refinement techniques [21, 30] could also be used when multiple targets fall along the ray.

Our study also only examined movements across the center point of the user's field of view. Our initial implementation indicates the captured templates are still applicable (just as templates are applicable across direction of movement). Our hope is that the *top-n* approach is useful in this context. This certainly requires careful evaluation and validation in the future. The alternative, capturing movements for every possible target location, would probably not be feasible, due to the resulting size of the template library.

Another limitation of the study was the lack of distractor targets, which were omitted to simplify the task environment and capture raw target acquisition movements. The visual presence of distractors could influence a user's behavior,

		Template Library Source											
		P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
Tested Participant	P1	8.52	11.72	16.11	14.16	11.90	11.37	11.79	10.71	9.58	11.69	13.41	13.06
	P2	11.69	10.32	17.86	13.63	11.95	11.17	10.39	11.53	11.17	11.73	11.81	12.30
	P3	19.15	17.83	11.80	17.02	14.97	18.36	15.57	20.95	20.57	18.07	16.79	16.51
	P4	14.14	11.80	14.96	13.05	12.00	12.21	10.86	14.19	15.23	12.49	12.00	12.95
	P5	11.32	11.80	18.15	13.77	9.24	12.28	10.71	12.04	10.66	11.72	11.56	11.74
	P6	10.37	12.62	20.28	14.96	12.31	9.35	11.57	11.23	10.58	11.90	13.49	13.52
	P7	11.68	11.12	18.30	13.59	11.04	11.28	8.98	11.44	10.84	11.78	11.67	11.91
	P8	10.00	11.07	20.39	14.68	10.59	10.63	11.15	9.43	9.32	10.56	11.47	12.32
	P9	10.03	12.88	24.46	17.35	13.21	10.93	13.44	11.03	8.13	12.13	14.73	14.89
	P10	10.46	12.06	18.74	14.38	10.94	11.71	11.13	10.89	9.91	10.15	12.59	12.74
	P11	12.57	10.18	16.94	13.60	10.13	10.97	10.44	11.49	11.40	11.80	9.90	10.80
	P12	13.28	10.71	15.93	13.49	10.24	11.64	10.95	12.03	11.38	11.93	10.53	10.58

Table 2. Accuracies (at 40%) for each participant (rows) when using another user's template library (columns).

which could in turn interfere with the model. In particular, a user may duck, lean, or reach around distractors to get a good view of their intended target. Incorporating, or filtering out, such behaviors is an interesting topic for future studies.

Our study may also be limited in that the same participant pool was used for both studies. It is possible that the parameter values which were tuned may have been biased towards this participant pool, and validating across a larger spectrum of users would be worthwhile future work.

Last, our results show predictions may not be as accurate as the actual ray position during the final stage of pointing. This creates an interesting opportunity for a hybrid approach, using our model in the early ballistic phase of pointing, and considers the current ray position in the adjustment phase.

Personalization of Template Libraries

An advantage of the model personalization is that it can be tuned to each user; a drawback is that training data is needed. One solution is to start a new user with a generic template library; slowly replacing that library with the user's own data as movements are collected. There may also be classes of users with similar behaviors, who could share predetermined template libraries. For example, users could be classified based on the extent with which they tend to move their head.

Complexity

A related factor is the number of templates that are in an individual's template library. Our implementation used approximately 2000 templates for each user. The model was able to run in real-time, with minimal optimization, with no perceivable impact on performance, at an input rate of 90 Hz. With each incoming input event frame, a prediction occurs in just under 11ms. However, any technique that requires additional resources must be used with caution, given the nature of computation-heavy virtual environments. Limits should be placed on the template library size, and the predictions could be performed on a separate thread.

Selection Facilitation Techniques

An exciting line of future research is developing new VR pointing enabling techniques that use movement predictions.

Techniques that dynamically adapt the CD ratio [28] could benefit particularly from early prediction. As the user initially moves the cursor, it could accelerate towards the predicted region, and decelerate when it arrives. As our model performs best at the early stages of movement (e.g. 40% mark), this type of facilitation may be particularly suitable for our model. In this case, the predicted landing position would not need to be precisely located at the intended target, as the technique would benefit from accelerating towards the general target region.

Alternatively, target snapping (e.g. [21, 57]) could benefit from early predictions – instead of just snapping to the closest target to the cursor, the technique could snap to the closest target in the predicted region. This could support faster access to targets, when predictions are made in the early stages of the movement. For a snapping technique, the required prediction accuracy would depend on target layout. In a dense environment, the prediction could be drawn to a nearby, inaccurate target. Based on our study results, we believe our model could work well for target snapping when there is an average of approximately 7° between targets. This would be large enough for larger UI buttons (e.g. the Oculus Quest Home screen buttons, which range from around 10° - 20°), but may be inadequate when selecting small and dense scene content in a VR environment.

It is important to note that any facilitation technique is likely to change the user's behaviors due to the perception/action loop in target acquisition [37]. This in turn may impact the model's performance. This work collected data when prediction was inactive but future studies should collect data and determine user behaviors when predictions are active.

Other Potential Applications

Predictive modelling could improve other aspects of VR user experiences beyond target selection. Recent work has shown the potential of haptic retargeting in VR [6], where a user's movements are biased towards a physical proxy. The success of such techniques required the ability to predict what region a user is moving towards [12]. Similarly, it has been recently shown that foveated rendering [2], which provides a higher resolution rendering at the user's point of focus, can benefit from predicting where a user is going to gaze next. By leveraging our technique, such predictions could be made early enough to be unnoticeable by the user. Finally, by predicting earlier what a user will do, associated processing could begin proactively, reducing latency for associated intended actions. This could be especially useful in VR where the reduction of latency is particularly important.

It is important to note that our technique is target agnostic, and as such predicts regions and not targets. In some use cases, (e.g. foveated rendering, CD gain adjustment), the technique could be useful on its own, without making a specific target prediction. In other cases, the model could be paired with facilitation techniques (e.g. target snapping) that are target-aware. In such circumstances, the predicted regions could be used to identify target likelihoods (e.g. [63]). This would be a useful area of future research.

CONCLUSION

This work has demonstrated that by using an adapted kinematic template matching technique (HC-KTM-7), ray-pointer landing positions can be predicted early in a user's movement. A key insight is that it is beneficial to integrate the movements of the head into the model. The presented approach increases landing position prediction accuracy by a factor of 1.8x at 40% of the way through a movement, when compared to traditional KTM, and by a factor of 2.7x when compared to a baseline using the current ray position. This work has provided a first exploration into pointer prediction in VR, opening the door for future enhancements to 3D user experiences.

REFERENCES

[1] Bashar I. Ahmad, Patrick M. Langdon, Simon J. Godsill, Richard Donkor, Rebecca Wilde, and Lee Skrypchuk. 2016. You Do Not Have to Touch to Select: A Study on Predictive In-car Touchscreen with Mid-air Selection. In *Proceedings of the 8th International Conference on Automotive User Interfaces and Interactive Vehicular Applications* (Automotive'UI 16). ACM, New York, NY, USA, 113-120. DOI: https://doi.org/10.1145/3003715.3005461

[2] Elena Arabadzhiyska, Okan Tarhan Tursun, Karol Myszkowski, Hans-Peter Seidel, and Piotr Didyk. 2017. Saccade landing position prediction for gaze-contingent rendering. *ACM Trans. Graph*. 36, 4, Article 50 (July 2017), 12 pages. DOI: https://doi.org/10.1145/3072959.3073642

[3] Ferran Argelaguet, Carlos Andujar. 2013. A survey of 3D object selection techniques for virtual environments. *Computers & Graphics*, *37,*(3), 121-136. DOI: https://doi.org/10.1016/j.cag.2012.12.003

[4] Takeshi Asano, Ehud Sharlin, Yoshifumi Kitamura, Kazuki Takashima, and Fumio Kishino. 2005. Predictive interaction using the delphian desktop. In *Proceedings of the 18th annual ACM symposium on User interface software and technology* (UIST '05). ACM, New York, NY, USA, 133-141. DOI: http://dx.doi.org/10.1145/1095034.1095058

[5] Gökçen Aslan Aydemir, Patrick M. Langdon, Simon Godsil. 2013. User target intention recognition from cursor position using kalman filter. In *International Conference on Universal Access in Human-Computer Interaction*. 419-426. Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-642-39188-0_45

[6] Mahdi Azmandian, Mark Hancock, Hrvoje Benko, Eyal Ofek, and Andrew D. Wilson. 2016. Haptic Retargeting: Dynamic Repurposing of Passive Haptics for Enhanced Virtual Reality Experiences. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16). ACM, New York, NY, USA, 1968-1979. DOI: https://doi.org/10.1145/2858036.2858226

[7] Marc Baloup, Thomas Pietrzak, and Géry Casiez. 2019. RayCursor: A 3D Pointing Facilitation Technique based on Raycasting. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (CHI '19). ACM, New York, NY, USA, Paper 101, 12 pages. DOI: https://doi.org/10.1145/3290605.3300331

[8] Pradipta Biswas, Gokcen Aslan Aydemir, Pat Langdon, Simon Godsill. 2013. Intent recognition using neural networks and Kalman filters. In *International Workshop on Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data*. 112-123. Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-642-39146-0_11

[9] Renaud Blanch, Yves Guiard, and Michel Beaudouin-Lafon. 2004. Semantic pointing: improving target acquisition with control-display ratio adaptation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '04). ACM, New York, NY, USA, 519-526. DOI: https://doi.org/10.1145/985692.985758

[10] Juan Sebastian Casallas, James H. Oliver, Jonathan W. Kelly, Frederic Merienne, Samir Garbaya. 2014. Using relative head and hand-target features to predict intention in 3D moving-target selection. In *2014 IEEE Virtual Reality (VR)*. 51-56. IEEE. DOI: 10.1109/VR.2014.6802050

[11] Elie Cattan, Amélie Rochet-Capellan, Pascal Perrier, and François Bérard. 2015. Reducing Latency with a Continuous Prediction: Effects on Users' Performance in Direct-Touch Target Acquisitions. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces* (ITS '15). ACM, New York, NY, USA, 205-214. DOI: https://doi.org/10.1145/2817721.2817736

[12] Lung-Pan Cheng, Eyal Ofek, Christian Holz, Hrvoje Benko, and Andrew D. Wilson. 2017. Sparse Haptic Proxy: Touch Feedback in Virtual Environments Using a General Passive Prop. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (CHI '17). ACM, New York, NY, USA, 3718-3728. DOI: https://doi.org/10.1145/3025453.3025753

[13] Nathan Cournia, John D. Smith, and Andrew T. Duchowski. 2003. Gaze- vs. hand-based pointing in virtual environments. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '03). ACM, New York, NY, USA, 772-773. DOI: http://dx.doi.org/10.1145/765891.765982

[14] Haan, Gerwin de and Koutek, Michal and Post, Frits H. 2005. IntenSelect: Using Dynamic Object Rating for Assisting 3D Object Selection. In *IPT/EGVE 2005*. 201-209. The Eurographics Association. DOI: 10.2312/EGVE/IPT_EGVE2005/201-209

[15] Augusto Esteves, Eduardo Velloso, Andreas Bulling, and Hans Gellersen. 2015. Orbits: Gaze Interaction for Smart Watches using Smooth Pursuit Eye Movements. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (UIST '15). ACM, New York, NY, USA, 457-466. DOI: https://doi.org/10.1145/2807442.2807499

[16] Ribel Fares, Shaomin Fang, and Oleg Komogortsev. 2013. Can we beat the mouse with MAGIC?. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '13). ACM, New York, NY, USA, 1387-1390. DOI: https://doi.org/10.1145/2470654.2466183

[17] Paul M. Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology*, *47*(6). 381-391. DOI: http://dx.doi.org/10.1037/h0055392

[18] Andrew Forsberg, Kenneth Herndon, and Robert Zeleznik. 1996. Aperture based selection for immersive virtual environments. In *Proceedings of the 9th annual ACM symposium on User interface software and technology* (UIST '96). ACM, New York, NY, USA, 95-96. DOI: http://dx.doi.org/10.1145/237091.237105

[19] C. C. A. M. Gielen, K. van den Oosten, F. Pull ter Gunne. 1985. Relation Between EMG Activation Patterns and Kinematic Properties of Aimed Arm Movements, *Journal of Motor Behavior*, 17(4). 421-442. DOI: 10.1080/00222895.1985.10735359

[20] Tovi Grossman and Ravin Balakrishnan. 2004. Pointing at trivariate targets in 3D environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '04). ACM, New York, NY, USA, 447-454. DOI: https://doi.org/10.1145/985692.985749

[21] Tovi Grossman and Ravin Balakrishnan. 2005. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '05). ACM, New York, NY, USA, 281-290. DOI: https://doi.org/10.1145/1054972.1055012

[22] Tovi Grossman and Ravin Balakrishnan. 2006. The design and evaluation of selection techniques for 3D volumetric displays. In *Proceedings of the 19th annual ACM symposium on User interface software and technology* (UIST '06). ACM, New York, NY, USA, 3-12. DOI: https://doi.org/10.1145/1166253.1166257

[23] Qi Guo and Eugene Agichtein. 2010. Ready to buy or just browsing?: detecting web searcher goals from interaction data. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval* (SIGIR '10). ACM, New York, NY, USA, 130-137. DOI: https://doi.org/10.1145/1835449.1835473

[24] Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassell. 1994. A survey of design issues in spatial input. In *Proceedings of the 7th annual ACM symposium on User interface software and technology* (UIST '94). ACM, New York, NY, USA, 213-222. DOI: http://dx.doi.org/10.1145/192426.192501

[25] Ricardo Jota, Miguel A. Nacenta, Joaquim A. Jorge, Sheelagh Carpendale, and Saul Greenberg. 2010. A comparison of ray pointing techniques for very large displays. In *Proceedings of Graphics Interface 2010* (GI '10). Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 269-276.

[26] Hilde Keuning-van Oirschot and Adrian J. M. Houtsma. 2001. Cursor displacement and velocity profiles for targets in various locations. In *Proceedings of Eurohaptics*. 108-112.

[27] George V. Kondrask. 1994. An Angular Motion Fitts' Law for Human Performance Modeling and Prediction. In *Proceedings of Engineering in Medicine and Biology Society.* 307.

[28] Werner A. König, Jens Gerken, Stefan Dierdorf, and Harald Reiterer. 2009. Adaptive Pointing --- Design and Evaluation of a Precision Enhancing Technique for Absolute Pointing Devices. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part I(INTERACT '09)*, Tom Gross, Jan Gulliksen, Paula Kotzé, Lars Oestreicher, Philippe Palanque, Raquel Oliveira Prates, and Marco Winckler (Eds.). Springer-Verlag, Berlin, Heidelberg, 658-671. DOI: https://doi.org/10.1007/978-3-642-03655-2_73

[29] Regis Kopper, Doug A.Bowman, Mara G.Silva, Ryan P.McMahan. 2010. A human motor behavior model for distal pointing tasks. *International journal of human-computer studies*, *68*(10). 603-615. Elsevier. DOI: 10.1016/j.ijhcs.2010.05.001

[30] Regis Kopper, Felipe Bacim, Doug A. Bowman. 2011. Rapid and accurate 3D selection by progressive refinement. In *2011 IEEE Symposium on 3D User Interfaces (3DUI)*. 67-74. IEEE. DOI: 10.1109/3DUI.2011.5759219

[31] Mikko Kytö, Barrett Ens, Thammathip Piumsomboon, Gun A. Lee, and Mark Billinghurst. 2018. Pinpointing: Precise Head- and Eye-Based Target Selection for Augmented Reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (CHI '18). ACM, New York, NY, USA, Paper 81, 14 pages. DOI: https://doi.org/10.1145/3173574.3173655

[32] David M. Lane, S. Camille Peres, Aniko Sandor, H. Albert Napier. 2005. A process for anticipating and executing icon selection in graphical user interfaces. *International Journal of Human-Computer Interaction*, *19*(2). 241-252. DOI: 10.1207/s15327590ijhc1902_5

[33] Edward Lank, Yi-Chun Nikko Cheng, and Jaime Ruiz. 2007. Endpoint prediction using motion kinematics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '07). ACM, New York, NY, USA, 637-646. DOI: https://doi.org/10.1145/1240624.1240724

[34] Joseph J. LaViola. 2003. Double exponential smoothing: an alternative to Kalman filter-based predictive tracking. In *Proceedings of the workshop on Virtual environments 2003* (EGVE '03). ACM, New York, NY, USA, 199-206. DOI: http://dx.doi.org/10.1145/769953.769976

[35] Jiandong Liang, Mark Green. 1994. JDCAD: A highly interactive 3D modeling system. *Computers and Graphics*. 18(4). 499-506. Elsevier. DOI: 10.1016/0097-8493(94)90062-0

[36] Michael McGuffin and Ravin Balakrishnan. 2002. Acquisition of expanding targets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '02). ACM, New York, NY, USA, 57-64. DOI: https://doi.org/10.1145/503376.503388

[37] David E. Meyer, Richard A. Abrams, Sylvan Kornblum, Charles E. Wright, JE Keith Smith. 1988. Optimality in human motor performance: ideal control of rapid aimed movements. *Psychological review*, 95(3), 340.

[38] Mark R. Mine. 1995. Virtual environment interaction techniques. *UNC Technical Report.* TR95-018.

[39] Atsuo Murata. 1998. Improvement of pointing time by predicting targets in pointing with a PC mouse. *International Journal of Human-Computer Interaction*, *10*(1). 23-32. DOI: https://doi.org/10.1207/s15327590ijhc1001_2

[40] Brad A. Myers, Rishi Bhatnagar, Jeffrey Nichols, Choon Hong Peck, Dave Kong, Robert Miller, and A. Chris Long. 2002. Interacting at a distance: measuring the performance of laser pointers and other devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '02).* ACM, New York, NY, USA, 33-40. DOI: https://doi.org/10.1145/503376.503383

[41] Mathieu Nancel, Olivier Chapuis, Emmanuel Pietriga, Xing-Dong Yang, Pourang P. Irani, and Michel Beaudouin-Lafon. 2013. High-precision pointing on large wall displays using small handheld devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '13). ACM, New York, NY, USA, 831-840. DOI: https://doi.org/10.1145/2470654.2470773

[42] Mathieu Nancel, Emmanuel Pietriga, Olivier Chapuis, and Michel Beaudouin-Lafon. 2015. Mid-Air Pointing on Ultra-Walls. *ACM Trans. Comput.-Hum. Interact.* 22, 5, Article 21 (August 2015), 62 pages. DOI: https://doi.org/10.1145/2766448

[43] Mathieu Nancel, Daniel Vogel, Bruno De Araujo, Ricardo Jota, and Géry Casiez. 2016. Next-Point Prediction Metrics for Perceived Spatial Errors. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (UIST '16). ACM, New York, NY, USA, 271-285. DOI: https://doi.org/10.1145/2984511.2984590

[44] Mathieu Nancel, Stanislav Aranovskiy, Rosane Ushirobira, Denis Efimov, Sebastien Poulmane, Nicolas Roussel, and Géry Casiez. 2018. Next-Point Prediction for Direct Touch Using Finite-Time Derivative Estimation. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (UIST '18). ACM, New York, NY, USA, 793-807. DOI: https://doi.org/10.1145/3242587.3242646

[45] Phillip T. Pasqual and Jacob O. Wobbrock. 2014. Mouse pointing endpoint prediction using kinematic template matching. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '14). ACM, New York, NY, USA, 743-752. DOI: https://doi.org/10.1145/2556288.2557406

[46] Ken Pfeuffer, Benedikt Mayer, Diako Mardanbegi, and Hans Gellersen. 2017. Gaze + pinch interaction in virtual reality. In *Proceedings of the 5th Symposium on Spatial User Interaction* (SUI '17). ACM, New York, NY, USA, 99-108. DOI: https://doi.org/10.1145/3131277.3132180

[47] Jeffrey S. Pierce, Andrew S. Forsberg, Matthew J. Conway, Seung Hong, Robert C. Zeleznik, and Mark R. Mine. 1997. Image plane interaction techniques in 3D immersive environments. In *Proceedings of the 1997 symposium on Interactive 3D graphics* (I3D '97). ACM, New York, NY, USA, 39-ff.. DOI: http://dx.doi.org/10.1145/253284.253303

[48] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. 1996. The go-go interaction technique: non-linear mapping for direct manipulation in VR. In *Proceedings of the 9th annual ACM symposium on User interface software and technology* (UIST '96). ACM, New York, NY, USA, 79-80. DOI: http://dx.doi.org/10.1145/237091.237102

[49] Kai Riege, Thorsten Holtkamper, Gerold Wesche, and Bernd Frohlich. 2006. The Bent Pick Ray: An Extended Pointing Technique for Multi-User Interaction. In *Proceedings of the 3D User Interfaces* (3DUI '06). IEEE Computer Society, Washington, DC, USA, 62-65. DOI: https://doi.org/10.1109/VR.2006.127

[50] Gang Ren, Eamonn O'Neill. 2013. 3D selection with freehand gesture. *Computers & Graphics,* 37(3). 101–120. DOI: https://doi.org/10.1016/j.cag.2012.12.006

[51] Hyocheol Ro, Seungho Chae, Inhwan Kim, Junghyun Byun, Yoonsik Yang, Yoonjung Park, Tackdon Han. 2017. A dynamic depth-variable ray-casting interface for object manipulation in AR environments. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2873-2878. IEEE. DOI: 10.1109/SMC.2017.8123063

[52] Florian Roider and Tom Gross. 2018. I See Your Point: Integrating Gaze to Enhance Pointing Gesture Accuracy While Driving. In *Proceedings of the 10th International Conference on Automotive User Interfaces and Interactive Vehicular Applications* (AutomotiveUI '18). ACM, New York, NY, USA, 351-358. DOI: https://doi.org/10.1145/3239060.3239084

[53] Jaime Ruiz, Edward Lank. 2009. Effects of target size and distance on kinematic endpoint prediction. *Technical Report CS-2009-25*, University of Waterloo.

[54] Jaime Ruiz and Edward Lank. 2010. Speeding pointing in tiled widgets: understanding the effects of target expansion and misprediction. In *Proceedings of the 15th international conference on Intelligent user interfaces* (IUI '10). ACM, New York, NY, USA, 229-238. DOI: https://doi.org/10.1145/1719970.1720002

[55] G. Schmidt, Y. Baillot, D.G. Brown, E.B. Tomlin, J.E. Swan. 2006. Toward disambiguating multiple selections for frustum-based pointing. In *3D User Interfaces (3DUI'06)*. 87-94. IEEE. DOI: 10.1109/VR.2006.133

[56] Frank Steinicke, Timo Ropinski, Klaus Hinrichs. 2006. Object selection in virtual environments using an improved virtual pointer metaphor. In *Computer Vision and Graphics.* 320-326. Springer, Dordrecht. DOI: 10.1007/1-4020-4179-9_46

[57] Lode Vanacken, Tovi Grossman, Karin Coninx. 2007. March. Exploring the effects of environment density and target visibility on object selection in 3D virtual environments. In *IEEE 3D user interfaces*. 117-124. IEEE. DOI: 10.1109/3DUI.2007.340783

[58] Eduardo Velloso, Jayson Turner, Jason Alexander, Andreas Bulling, Hans Gellersen. 2015. An empirical investigation of gaze selection in mid-air gestural 3D manipulation. In *Human-Computer Interaction*. 315-330. Springer. DOI: 10.1007/978-3-319-22668-2_25

[59] Daniel Vogel and Ravin Balakrishnan. 2005. Distant freehand pointing and clicking on very large, high resolution displays. In *Proceedings of the 18th annual ACM symposium on User interface software and technology* (UIST '05). ACM, New York, NY, USA, 33-42. DOI: http://dx.doi.org/10.1145/1095034.1095041

[60] Martin Weier, Thorsten Roth, André Hinkenjann, and Philipp Slusallek. 2018. Predicting the gaze depth in head-mounted displays using multiple feature regression. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications (ETRA '18)*. ACM, New York, NY, USA, Article 19, 9 pages. DOI: https://doi.org/10.1145/3204493.3204547

[61] Haijun Xia, Ricardo Jota, Benjamin McCanny, Zhe Yu, Clifton Forlines, Karan Singh, and Daniel Wigdor. 2014. Zero-latency tapping: using hover information to predict touch locations and eliminate touchdown latency. In *Proceedings of the 27th annual ACM symposium on User interface software and technology* (UIST '14). ACM, New York, NY, USA, 205-214. DOI: https://doi.org/10.1145/2642918.2647348

[62] Shumin Zhai, Carlos Morimoto, and Steven Ihde. 1999. Manual and gaze input cascaded (MAGIC) pointing. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (CHI '99). ACM, New York, NY, USA, 246-253. DOI: http://dx.doi.org/10.1145/302979.303053

[63] Brian Ziebart, Anind Dey, and J. Andrew Bagnell. 2012. Probabilistic pointing target prediction via inverse optimal control. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces* (IUI '12). ACM, New York, NY, USA, 1-10. DOI: https://doi.org/10.1145/2166966.2166968