Hook: Heuristics for Selecting 3D Moving Objects in Dense Target Environments

Michaël Ortega
Laboratoire d'Informatique de Grenoble - Université Grenoble I / CNRS
385, rue de la Bibliotheque, B.P. 53 F-38041 Grenoble cedex 9, France

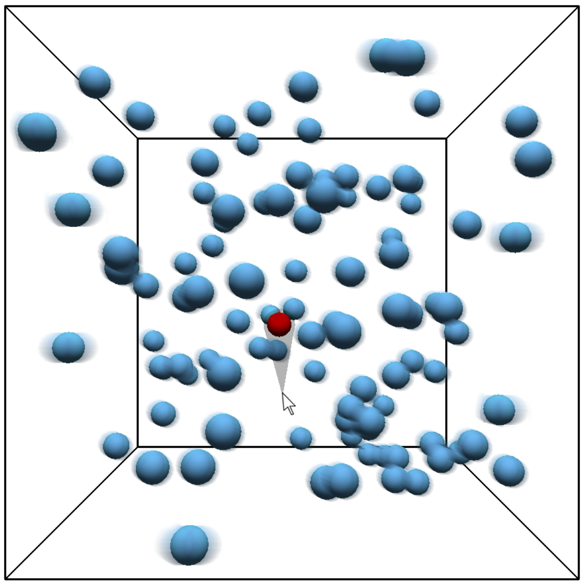


Figure 1: Hook in action in dense environments and fast-moving targets. The visual feedback shows the moving target that, according to cursor behavior, the system detects as being the target-of-interest.

Abstract

This paper presents Hook, a new interaction technique for selecting moving targets. As opposed to existing techniques, Hook uses heuristic methods. It allows pointing in dense 3D environments, and on targets moving with high velocity. Moreover, Hook minimizes the impact of its visual feedback for maintaining user's interaction comfort. Two adaptations of Hook for 2 dof (degrees-of-freedom) and 3 dof interaction for 3D environments have been evaluated. Results clearly show that Hook outperforms the existing methods in pointing time and error rates, for fast and slow targets, in the two configurations. All the participants confirmed the expected feeling with regards to ease of use.

Index Terms: I.3.6 [Computer Graphics]: Methodologie and Techniques—Interaction Technique; I.3.4 [Computer Graphics]: Graphics Utilities—Virtual device interfaces
Additional Keywords: 3D interaction, Selection, Moving Targets

1 Introduction

Moving targets are now found in applications including air traffic control displays, hyperlinked video media, video games. Selecting them is always more challenging than selecting static targets. The difficulty of the selection increases while (1) target size decreases, (2) target velocity increases and (3) target density increases. The selection is even more difficult in 3D environments (filmed or synthetic environments) because a target of interest can be occluded by others targets. Selecting someone in crowds with a surveillance video system is an example of a dense environment with small targets and occlusion. In some cases, like in action sport footage, both the objects of interest and the camera can move. Target movements become unpredictable, and the selection is even more difficult.

In this context, as Hasan et al. [3] wrote, for completing the selection "the user must continually track the target and simultaneously plan to move the cursor over it". This underscores the key point of the technique we propose here. Since the user *follows* the target for selecting it, the Hook technique tracks the cursor behavior for assisting the selection. Indeed, observing the history of cursor displacements, and the history of distances between each target and the cursor, the system can estimate which target is tracked, and then propose a selection to the user who just has to validate it. In other words, the user follows the target of interest, and the system will know which target it is.

This paper presents the implementation of this technique, and two experiments which investigate its performance. Hook has been compared to the basic pointing (non assisted pointing), and to Bubble cursor [2, 8], for the case of 3D object selection. The first evaluation involves a desktop configuration, in which pointing is done on a standard screen with a mouse. The second evaluation involves an immersive configuration in which pointing is done with a 3dof (degrees-of-freedom) device. Both evaluations highlight the benefits of this new interaction technique for selecting moving targets in a 3D environment. More than simply improving the pointing time, Hook drastically decreases the error rate and allows pointing targets in high density environments, with high velocity targets that are not possible to capture with other techniques.

2 Related Work

From the field of HCI, only a few studies have explored the selection of moving targets. Most of them propose to adapt the studies of static environments, and to improve the "per-frame" static behavior. We classify them into two main categories.

The first one has been illustrated by Ilich et al. [5] with the selection of moving targets on interactive video browsing. The main idea of the proposed selection technique, called "Click-to-Pause", is to temporarily pause the content while selection is in progress. The technique involves two steps for pointing: pause and then selection.They are successively done, with only one mouse button: depressing the button pauses all moving objects and the user moves the cursor to the target of interest. Next, releasing the button selects the target. This two-step technique corresponds to a case of static target selection. Click-to-Pause results in lower pointing time than the classic pointer for small and/or fast targets. However, it implies (1) losing context when screen is in pause and (2) dependence on target density: for the case of 3D environments (filmed or synthetic), in which an object can pass behind another from the user point of view, the pause has to be done in a precise point-in-time for avoiding occlusion. With high target density and/or fast targets this is a serious drawback. Moreover, (3) for the case of successive pointing tasks, stopping the animation at each selection can perturb the fluency of the global task (in games for example).

An improvement of Click-to-Pause has been proposed by Hasan et al. [3] for avoiding the loss of context. It is called "Target Ghost". When the user invokes the Target Ghost technique, proxies are created from each moving target, and placed at the exact position of the target when invocation is done. Behind these static proxies, targets are displayed like ghosts (transparent, less visible) and still move, allowing the user to maintain context. These proxies are selectable. Therefore this technique again corresponds to the selection of static targets but without loosing the context. In the same research study, Hasan et al. proposes to combine Target Ghost with assisted pointing techniques. In Target Ghost, selecting proxies is similar to selecting static targets. So it becomes possible to use assisted methods like Bubble cursor[2] (technique which modifies the pointing space for enhancing the pointing performances). This result shows that Bubble Ghost (the combination of Bubble Cursor with Target Ghost, i.e. the use of Bubble Cursor when content is paused) outperforms other combinations in 2D. However, Target Ghost is also dependent on target density, and even more so when combined with Bubble Cursor. Indeed, duplicating objects creates clutter that can be difficult to manage when target density increases: context becomes too hidden by proxies, and is finally lost. Moreover, users can be overloaded with the increasing amount of visual information.

The second category of techniques has been recently illustrated by Hasan et al. [3], who proposed an interaction technique called "Comet". The technique is based on the concept of target expansion used in static environments. Each target has an appended tail, as present in astronomical comets, which corresponds to a representation of its past positions. The tail's size is based on the speed and the size of the target itself. Fastest targets have longer tails than slower targets. This technique enlarges the selecting surface of the target and, according to Fitts'law, the target is therefore easier to select. This technique outperforms both unassisted and assisted techniques (respectively Basic pointer and Bubble cursor[2]). However, in scenes with a high density of targets, Comet adds clutter, and targets' tail can overlap. Moreover, in games for instance, adding visual elements may have no sense relative to the graphical chart, or the aesthetic of the displayed 3D environment [7].

About using the dynamic behavior of the environment, a first study has been proposed by De Haan et al. [1]: "Intenselect". This technique takes into account the correlation of past targets and pointer movements in order to predict which target the user is trying to select. However, the method is highly correlated to the "Ray-Casting" technique, and only proposed for virtual environments.

The technique that we describe in the following section is generic. It works with low and high target densities, with low and high target velocities, and minimizes the amount of visual feedback and therefore the context perturbation because (1) no display pause is done, (2) the target appearance is not modified and (3) the visual feedback solely linked to the cursor is minimized.

3 Hook

The Hook technique is inspired by the following observation: if a target has unpredictable, but non-brownian movements (like a fly flying, for instance), the user cannot anticipate future positions and has no choice but to follow the target and try to catch it. Hook suggests that this tracking behavior can be detected, i.e. the system can observe the relation between the moving targets and the cursor over time. Based on the observation, the system can estimate which target is tracked, and it can assist the user in pointing at it.

Compared to existing methods, Hook uses time in the computation of pointing assistance, and it uses an heuristic method for computing *scores* of each target during the selection. These scores change during the pointing task, and their value increases or decreases depending on the distance between the cursor and each target. By sorting the targets by score, the system provides a list of the closest targets over time. A target which has been regularly proximal to the cursor will have a high score. The system considers the target with the highest score as "hooked" and provides a visual feedback to inform the user that the target is selectable.

If the user decides to change its target of interest (*TI*), s/he will follows a new one. Thus, the latter score will increase while that of the previous *TI* will decrease. After a while, the score of the new *TI* will be higher and the user will be able to select it. The computation of the scores must then be efficient. Indeed a tradeoff must be defined in order to (1) provide a good stability on "Hooked" target, i.e. the system will not easily swap between targets in order to avoid perturbing the user interaction; (2) minimize the inertia in changing the TI, i.e. when the user decides to change the *TI*, the time needed will not be too long.

3.1 Scores Computation

At each time step, Hook computes the distance from each target to the cursor, and so classifies the targets in an ordered list of increasing distance values. According to that list, each target's score will be increased or decreased. It is critical to avoid system inertia, so only a limited number of closest targets (NCT) will have their scores increasing. All the others will have their scores decreasing. It is also critical to maintain stability on Hooking, so NCT cannot be too small. For instance, if $NCT = 1$, only the closest target will be pre-selected at each time step, and this corresponds to the Bubble cursor technique, which is highly dependent on distractors in dense environments of moving targets.

NCT computation has been established from observations of users behavior during the selection of fast targets with a basic pointer (non-assisted pointing). We observed that the cursor was following the *TI* at a roughly constant distance, which increases if target velocity increases. Considering the highest distance, i.e. the distance observed with the highest velocities that we propose in the evaluation, as the radius of the surrounding of the cursor, NCT becomes the mean number of targets that could be in this surrounding. It is then dependent on: (1) the total number of targets (TT), (2) the spherical volume of the surrounding (SV) and (3) the global frame volume (FV) in which targets move. So: $NCT = TT * SV/FV$.

The amount of score, added or substracted to one target, is time dependent and proportional to the rank of the target in the ordered list of closest targets. For NCT closest targets (T), the formula at time t is *eq (1)*, where i is the current rank of the target in the list of closest targets. All other target scores decrease with the same value (*eq (2)*).

$$T_i score(t) = T_i score(t-1) + (NCT - i) * \Delta t \tag{1}$$

$$T_i score(t) = T_i score(t-1) - NCT/2 * \Delta t \tag{2}$$

A score cannot decrease below 0. There is no maximum score, because it is always necessary to discriminate NCT closest targets with different scores.

This computation ensures that (1) a regularly close target has a high potential of being the one the user wants to select; (2) the more user tracks a target, the more the hooking will be strong and free from distractors and occlusion; (3) if the user tracks a target during N seconds, the time needed to change to the desired one is below $N/2$ seconds ; (4) Even with high target density and fast targets, hooking a target is just a question of time and no random exists.

3.2 Interaction and Feedback

Like other assisted pointing techniques [2, 8, 5], Hook is invoked by depressing a button. At this point-in-time, all the targets' scores are initialized to 0, so that all the targets have an equivalent potential of being the *TI*. When the user tracks the *TI*, the system computes the targets' scores, as described in the previous section. Validation of the selection is done by releasing the button.

During the tracking, a visual feedback informs the user about the currently hooked target. This feedback has to display the link between the cursor and the target, and has to highlight the target. For minimizing the impact on the global graphical aspect of the environment, this feedback is then comprised of (1) a semi-transparent cone from the cursor to the *TI*, (2) a semi-transparent enlarged bounding box of the target (3) an highlighting of the target. The cone base is fixed to the target center while its top corresponds to the cursor position.

3.3 2D and 3D interaction

The underlying principle of the Hook technique for pointing in 3D environments is similar for both 2D and 3D interaction. By 2D interaction we mean the use of a two degrees-of-freedom device, like a mouse, which manipulates the cursor on a projection plan of the 3D environment (classical desktop configuration). By 3D interaction we mean the use of a three degrees-of-freedom device, which manipulates the cursor along the 3 axes defining the 3D environment.

In 2D interaction, the distance computation is placed in the projection plan, between the 2D cursor position and the 2D projection of the 3D targets. The feedback is then displayed in the projection plan, and overlaps the displayed scene. The enlarged bounding box is projected. The cone is reduced to a 2D triangle whose vertices are: the cursor position, and two points on the projected bounding box. These two points are diametrically opposed, and the line they draw on the projection plan is orthogonal to the line defined by the cursor position and the target center (see figure 1).

For the case of 3D interaction, Hook is very similar to 2D interaction. We use the principle of a virtual hand for moving the cursor in three dimensions, then the computation of the distance is in three dimensions, as is that of the bounding box and the cone as well.

4 EVALUATIONS

An evaluation has been done for each version of Hook: 2D and 3D interaction. In both techniques we compare Hook with the two existing techniques which allow pointing moving targets in dense environments, and which minimize both the loss of context and their impact on the visual aspect of the scene (see previous sections of this paper for more details). Therefore, the three techniques we evaluated are : Basic pointer (unassisted), Bubble cursor (2D and 3D version with its original feedback [6]) and Hook.

4.1 Participants

10 participants were recruited from the local University. Two are left-handed. All were frequent users of classical WIMP interaction ("Windows, Icons, Menus, Pointer"), but none of them were accustomed to interact in 3D environments.

4.2 Apparatus

For 2D interaction, a traditional desktop configuration is used. A perspective openGL projection, with a 60 degrees field-of-view, displays the scene on a 19 inch screen with a resolution of 1600x1200 pixels. The projection of the front face of the cubic frame measures 850x850 pixels (21.7cm), and the back face around 450x450 pixels (11.48cm). The pointing device was a standard 2dof mouse.

For 3D interaction, a perspective openGL projection displays the scene on a large projection screen, using passive stereoscopic technology for improving depth perception. An optical tracking system is used to track the 3D hand position of the user. The wand was equipped with a single button. The size of the virtual cubic frame measures 90cm, so the target diameter measures 6cm. The cursor is a 3D cross, mapped to the user's hand with a simple virtual hand technique. The mapping is 1/1, and there is a translation of 50 cm on depth axis between the real hand and the 3D cursor for avoiding occlusion and accommodation issues.

4.3 Experimental design

For this controlled experiment, the 3D environment has been computed, and not filmed as it could be in a video tracking software. The two evaluations involve the same density of moving targets: 100. This high density has been chosen for stressing the techniques. All of the targets have the same size, the same color and the same shape. They move in a fixed cubic 3D frame (see figure 1), and $NCT = 20$. Targets are spheres, and their diameter is equal to $1/15$ the frame size. Since the difficulty of a task is highly correlated with the moving targets velocity [4], the latter is the only parameter that changes during each evaluation. Five velocities were tested, from 1 to 5. At velocity 1, a target can cross the cubic frame in $6s$ (seconds). At velocity 5, a target can cross the cubic frame in $6/5 = 1.2s$. The latter velocity is high and stresses the techniques, but also the participants. Each of the 5 possible velocities is used 20 times in each evaluation. So, the participants have to undertake $3x5x20 = 300$ trails for each version (2D and 3D) of the evaluation.

About target movements, we wanted to avoid predictable behaviors. So, each target has a movement direction that can randomly change at each animation step. For having unpredictable movements, but not brownian ones, the direction vector can only vary within a cone, centered on the previous direction vector (see figure 2). The cone angle was fixed to 10 degrees during these evaluations. When targets reach the cubic frame border, the direction vector is simply inverted.

Each trial is decomposed as follows. First, the user brings the cursor to a determined zone on the display border: a 2D square or a 3D cube for respectively 2D or 3D interaction. A click-and-release on this zone randomly highlights a TI to reach. Then user executes the pointing task, by trying to select the TI with the best compromise between velocity and precision. Only movement time (from the TI highlighting to the selection) and errors (target selected or clicked beside) are captured.

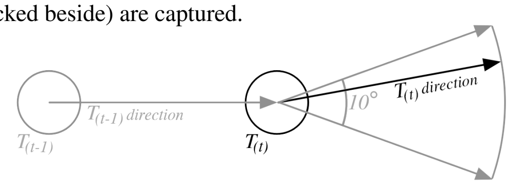


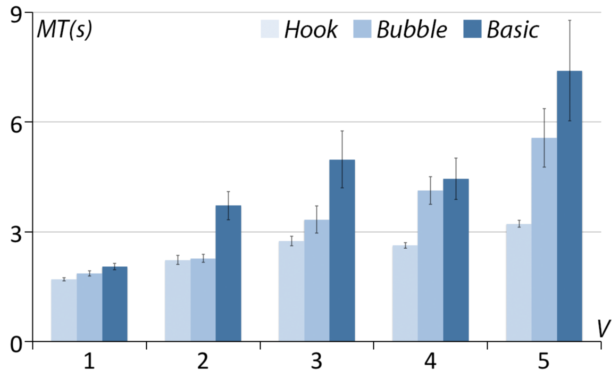
Figure 2: For having unpredictable movements, but not brownian ones, at each time step the new direction of the target randomly varies within a cone of ten degree vertical angle.

4.4 2D Interaction Results

The left of Figure 3 shows the mean of movement time. We discarded trials in which errors occurred, resulting on a high percentage of trials for Bubble and basic pointer on high velocities (see Figure 3 *right*). Repeated measures analysis of variance (ANOVA) showed a significant effect of the technique ($F(2,1693) = 112,67, p < 0.0001$) and of the velocity ($F(4,1693) = 94,09, p < 0.0001$). Indeed, as observed on the figures, increasing velocity increases movement time for the three techniques. A similar result is given by a chi^2 test for error rate: technique and velocity have a significant effect on error rate (respectively $chi^2(2) = 899, p < 0.0001$ and $chi^2(4) = 182, p < 0.0001$). Hook has the smallest error rate: 7.1%.

4.5 3D Interaction Results

The left of Figure 4 shows the mean of movement time. As in 2D interaction, we discarded trials in which errors occurred. Here, the task was more difficult than in 2D because the users had to manage the depth dimension. We observed that the task when using basic pointing is beyond the users capacities when the velocity is up to 2. Repeated measures analysis of variance are only done with Bubble and Hook data, and showed a significant effect



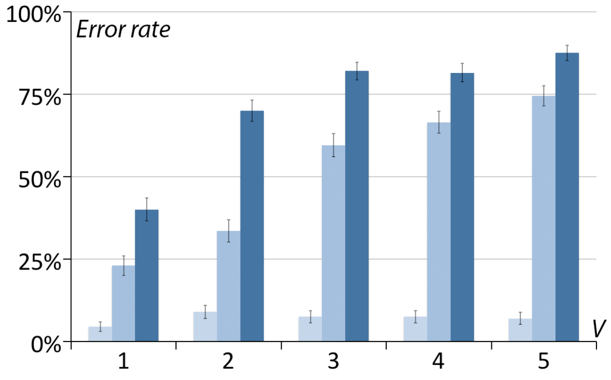
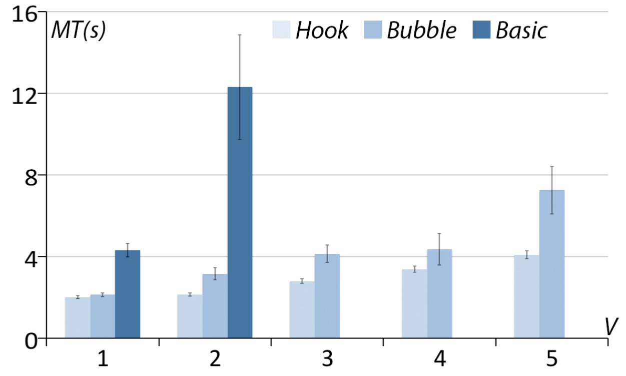


Figure 3: 2D Interaction results: movement times and error rates , across 5 velocities for the 3 compared techniques (Error Bars: +/- 1SE).



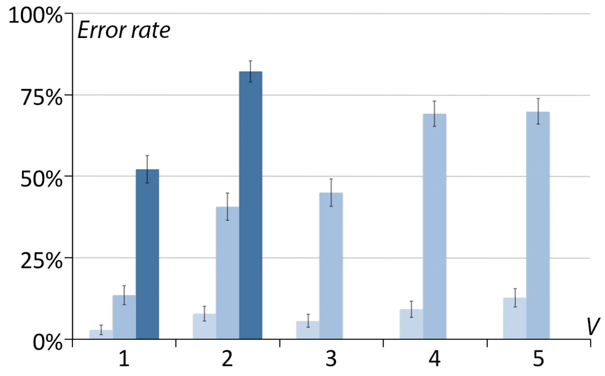


Figure 4: 3D Interaction results: movement times and error rates (Error Bars: +/- 1SE). With Basic, only velocity 1 and 2 were within users skills.

of the technique ($F(2,1821) = 125,35, p < 0.0001$) and of the velocity ($F(4,1821) = 32,9, p < 0.0001$). A similar result is given by a chi^2 test for error rate: technique and velocity have a significant effect on error rate (respectively $chi^2(2) = 565, p < 0.0001$ and $chi^2(4) = 62.4, p < 0.0001$). Hook has again the smallest error rate: 7.71%.

5 Discussion and Conclusion

Hook outperforms both techniques, Basic and Bubble cursors, in high velocities in 2D and 3D interactions. Looking at the results, and especially the error rates, Hook seems to be the only one technique really adapted to select moving targets in dense environments. Indeed, the error rate is within an acceptable range for all velocities. The participants felt it, and expressed the feeling of usage comfort provided by Hook. A participant said: "With this technique, the user is sure that the more he waits, and the more the Hook is stabilized, the more the selection will be a success".

A surprising result is the outperforming of Hook with small velocities. Indeed, because of the time the users usually wait for having a stabilized Hook, we expected that, for small velocities, Basic and bubble techniques are faster. Even users expected this result. In 2D interaction, one of them said that "despite the usage comfort, Hook seems to be slower than the other two techniques for targets which move slowly". Results show that this feeling was not correct for the velocities tested and, beyond the performances in movement time, error rate is also the best with Hook.

In conclusion, Hook is the first interaction technique truly adapted for selecting moving targets in 3D dense environments. Results of the two evaluations are encouraging. In future works, we will investigate a finer tuning of NCT, for possibly integrating target velocity in an adaptive computation of the cursor surrounding volume SV. Next, we will integrate Hook into a real existing environment: a game, for doing a longitudinal experiment, and for studying its graphical impact, and its real usability *in-vivo*. The Hook visual feedback will also be studied. Indeed, some participants proposed the idea of having a kind of visual *progress bar* in order to inform the user on the "hooking quality", i.e. the level of stabilization on the target. This will be investigated for knowing the effect on performance and usage comfort.

Acknowledgements

Thanks to Laurence Nigay (LIG) for the fruitful first reviews, and to Nadine Mandran (LIG) for helping with the statistics.

References

[1] G. De Haan, M. Koutek, and F. H. Post. *IntenSelect: Using Dynamic Object Rating for Assisting 3D Object Selection*. Proc. of EGVE (2005), pp. 201-209.

[2] T. Grossman and R. Balakrishnan. *The Bubble Cursor: Enhamcing Target Acquisition by Dynamic Resizing of the Cursor's Activation Area*. Proc. of CHI (2005), pp. 281-290.

[3] K. Hasan, T. Grossman, and P. Irani. *Comet and Target Ghost: Techniques for Selecting Moving Targets*. Proc. of CHI (2011), pp. 839-848.

[4] E. R. Hoffman. *Capture of Moving targets: a modification of Fitts'Law*. Ergonomics 34 (1991), pp. 211-220.

[5] M. Ilich. *Movin Target Selection in Interactive video*. M.Sc Thesis, University of British Columbia.

[6] J. Laukkanen, P. Isokoski, and K.-J. RUaiha. *The Cone and the lazy Bubble: Two Efficient Alternatives between the Point Cursor and the Bubble Cursor*. Proc. of CHI (2008), pp. 309-312.

[7] D. Mould and C. Gutwin. *The effects of Feedback on Targeting with Multiple Moving Targets*. Proc. of GI (2004), pp. 25-32.

[8] L. Vanacken, T. Grossman, and K. Coninx. *Exploring the Effects of Environment Density and Target Visibility on Object Selection in 3D Virtual Environments*. Proc. of 3DUI (2007), pp. 117-124.