Starfish: a Selection Technique for Dense Virtual Environments

Jonathan Wonner
ENS Cachan - Bretagne
University of Strasbourg -
LSIIT, CNRS UMR 7005
jonathan.wonner@ens-cachan.org

Jérôme Grosjean
University of Strasbourg -
LSIIT, CNRS UMR 7005
grosjean@unistra.fr

Antonio Capobianco
University of Strasbourg -
LSIIT, CNRS UMR 7005
a.capobianco@unistra.fr

Dominique Bechmann
University of Strasbourg -
LSIIT, CNRS UMR 7005
bechmann@unistra.fr

ABSTRACT

We present Starfish - a new target selection technique for virtual reality (VR) environments. This technique provides a solution to accurately select targets in high-density 3D scenes.

The user controls a 3D pointer surrounded by a starfish-shaped closed surface. The extremity of each branch ends exactly on preselected near targets. The shape is an implicit surface built on the segments going from the pointer to each of these targets. As the pointer moves across the scene, the starfish shape is dynamically rebuilt. When it is locked the pointer is allowed to move inside the volume, slide down the desired branch, reach and select the corresponding target. Since the pointer stays within the shape, targets are easy to reach and select.

Categories and Subject Descriptors

I.3.6 [**Methodology and Techniques**]: Interaction techniques; H.5.1 [**Multimedia Information Systems**]: Artificial, augmented, and virtual realities

Keywords

Virtual reality; 3D interaction; Selection; Implicit surface; Starfish

1. INTRODUCTION

Selecting a target is one of the most fundamental tasks in VR environments [3]. However, this task faces difficulties inherent to such environments, especially at high levels of density. Our work is part of an application focused on editing volumetric meshes. In such a scenario, vertices and edges can partially or totally occlude the target. Moreover, selecting a target closely surrounded by these distractors requires high precision, due to the natural hand tremors [9]. It is also known that the depth perception is biased in VR environments [1]. Thus, precisely locating the depth of the pointer in the scene, or a specific target among others, may be difficult.

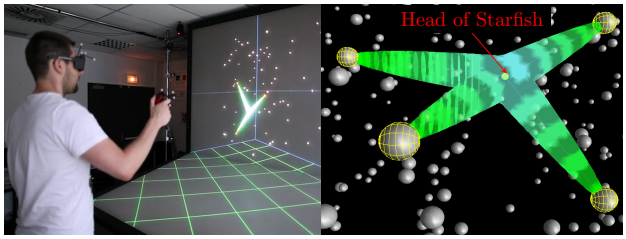

Figure 1: The Starfish selection technique.

The focus of our work is to design a new 3D selection technique, satisfying the following criteria:

- Selection can be performed in sparse or dense scenes.
- The user can effortlessly and quickly select his target.

The technique has to help the user to be precise, and to offer a solution for occluded targets. Basically, our approach was to look for techniques addressing some aspects of these constraints, and to identify their relevant features to design our technique.

Some previous approaches suggest to preselect a subset of targets and then to offer a menu to the user. We think this could be a solution to our first constraint. The SQUAD technique [7] preselects a set of targets using sphere-casting, and then distributes the retained targets into a quad-menu. Since the number of targets is iteratively reduced by intermediate selections, this approach does not require a high level of precision, even in a dense scene, or when the target is initially occulted. However, the SQUAD technique increases the number of necessary selections, and disconnects

VRST'12, December 10–12, 2012, Toronto, Ontario, Canada.

the scene from the menu [4]. We assume that keeping the local context is important not to disturb the user, especially in the case of targets with the same appearance and only distinguishable by their position in space, like the vertices of a mesh.

About our second guideline, we think that constraining a pointer into a specific-shape volume, is an elegant solution to guide the user during the selection gesture. The Hard-Borders technique [5] is a pie menu whose items are placed on the vertices of a polyhedron. The border of this shape has been made haptically solid, to guide the user within the polyhedron. The C^3 technique [6] is the VR counterpart of a 2D pie-menu. Items are placed around the user's hand, so that the user can quickly select an item by moving a pointer in one specific direction. Whereas the C^3 items are arranged in a regular grid, we can adapt this idea to a more generic 3D scene, by suggesting to the user the directions of a subset of targets, and constraining the pointer in these directions.

All these observations led us to design the Starfish technique (Selection of TARgets From Implicit Surfaces with simulated Haptics), represented in figure 1. The user controls a pointer, surrounded by a starfish-shaped volume: several branches start from the pointer (called the *head* of Starfish) to near preselected targets. At any moment, the shape is rebuilt according to the position of the head. When the desired target is preselected, the user can lock the shape and move the head inside its volume. We graphically constraint the pointer within the shape, so that the user is guided along the branch, which acts as a bottleneck, to reach and stop on the target. In the next section, we describe the design of Starfish.

2. STARFISH

Starfish is based on a closed surface freely positioned by the user. This surface is built on the union of several branches, starting from a pointer (called *head* of Starfish) to a set of near targets. As a result, the surface has the shape of a starfish. The user controls the head of Starfish like a 3D pointer: a direct mapping is used between the user's hand and the head of Starfish. The branches are dynamically rebuilt while the head is moved. At any time, in particular when the desired target is reached (or *captured*) by one of the branches, the user can lock the shape. The head of Starfish is still controlled by the user, but is now virtually constrained within the closed surface. The head can therefore lean on the surface along the branches, reach the target at the extremity of the branch, and select it.

In this section, we first focus on how the user controls Starfish. We then detail the algorithm behind the creation of the shape with an implicit surface [2]. Finally, we explain how we constraint the head of Starfish within the surface.

2.1 Control of Starfish

In our implementation, Starfish requires a tracker device, equipped with two buttons, called Move button and Select button. As long as no button is pressed, the head of Starfish is stationary. This state, called Idle mode, allows the user to rest his arm without moving Starfish. Moreover, to reach distant targets, the user can drag and drop Starfish in the scene. When the Move button is pressed, the position of the head is mapped to the position of the device. We call this state the Move mode. At any moment, the surface is rebuilt to capture targets near the head. When the Select button is pressed, the surface is locked, and the head of Starfish is constrained within its shape. This state is called the Select mode. When the user releases this button, we first check if a target is selected. If so, the user can interact with it. The target at the extremity of a branch is selected when the head of Starfish has entered deep enough in the branch. In our implementation, the user can select the target when the head has reached 30% of the length of the branch.

2.2 Building of the surface

The goal of Starfish is to locally help the user to select a target. This assistance is provided by preselecting a subset of targets and putting spatial constraints on the head of Starfish, so that these targets can be more easily reached. These two aims are achieved by building a closed surface which captures the preselected targets, and by constraining the head of Starfish inside the volume formed by the surface. We assume that the shape of a starfish (*i.e.* a union of branches), is adequate for that purpose. Thanks to the bottleneck shape of each branch, the head of Starfish slides on the surface, and stops precisely on the target. Moreover, branches indicate to the user in which directions he can move, and help to perceive the relative position of the neighbouring targets.

2.2.1 Preselecting targets

For reasons of visibility and comfort, Starfish should not preselect all the nearest targets, but provides help to the more relevant ones. These targets define the skeleton on which the surface will be built, *i.e.* a set of segments between the head of Starfish, H, and the targets T_i. We follow these intuitive guidelines:

1. **Distance filter:** Starfish is a local assistance, so all targets whose euclidean distance to the pointer is larger than a parameter $R_{\max}$ are rejected. Once the desired target is captured, the selection has to be fast. We suggest that the value of $R_{\max}$ should be chosen so that the user can reach each preselected targets by moving the device less than 50 cm.
2. **Angle filter:** If the angle between two branches is too small, it may be difficult to enter the chosen one without mistake. To address this issue, we check each couple of targets. If the angle between two segments is smaller than a parameter $\Theta_{\min}$, the farthest target is rejected. From preliminary tests, we feel that Starfish is more comfortable when $\Theta_{\min}$ is larger than or equal to $\frac{\pi}{8}$ rad.
3. **Quantity filter:** Finally, we only accept the $N_{\max}$ targets nearest to the head among the remaining, to avoid cluttering the shape with too many branches. In our modelling application, we choose to keep only 3 or 4 branches.

After applying these filters, if the desired target is not the nearest one, it may be rejected. However, since the shape is constantly rebuilt, adjusting the Starfish position to capture the correct target does not require much effort, and is rather fast and intuitive. The set of segments between the head of Starfish and the retained targets is called the *skeleton*.

2.2.2 *Building the shape*

Each segment starts from the head to a preselected target. To obtain the desired Starfish shape, shown on figure 2A, an implicit surface is used. The surface of Starfish can be seen as the set of points which share a same potential value, in a potential field generated by the skeleton. Each segment of the skeleton generates a branch-shaped potential field, as illustrated in figure 2B. The fields of all the branches are then merged. Finally, the isopotential surface S_0 crossing all the targets at the extremities of the segments of the skeleton is chosen, and displayed as explained in section 2.2.3.

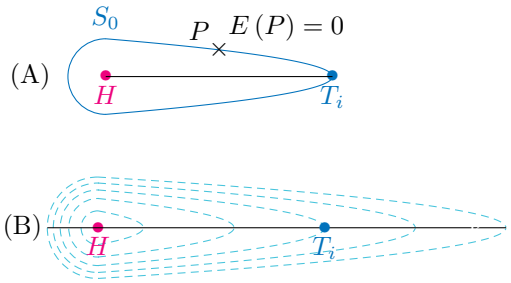

Figure 2: (A): 2D section of the isopotential surface S_0. This surface crosses the target T_i, and surrounds the head H. (B): The potential field generated by segment $[HT_i]$.

To generate such an implicit surface, we need a potential function E, and a specific distance function d to the skeleton.

The function E computes the potential of a point P according to its distance to the skeleton. In our implementation, the following function is used:

$$E(P) = \begin{cases} 0 & \text{if } d(P) \geqslant D_{\max}, \\ \left(1 - \dfrac{d(P)^2}{{D_{\max}}^2}\right)^2 & \text{otherwise.} \end{cases} \quad (1)$$

$D_{\max}$ is a distance which value is the radius of the branch at its base, and sets the thickness of the branch. So that the technique scales when the density increases, we recommend to vary this parameter according to the length of the branch. In our implementation, $D_{\max}$ quadratically depends on this length. All points of the surface S_0 have a distance to the skeleton equal to $D_{\max}$. Therefore, due to the particular desired shape, an euclidean distance is not relevant.

The distance function d is carefully designed so that the isopotential surface S_0 crosses all the targets retained by the filters defined in section 2.2.1, and surrounds the head of Starfish. We define a distance function d_i to each segment $[HT_i]$. These functions are then merged to obtain a distance function to the complete skeleton, as follows:

$$d(P) = \min_{i=1\ldots n} (d_i(P)). \quad (2)$$

The desired appearance of the surface for one single segment, shown on figure 3A, matches the isodistance surface $d_i = D_{\max}$. The head of Starfish is located at the intersection of all branches, and therefore should have a null distance to each segment. Consequently, our functions d_i must verify the following constraints:

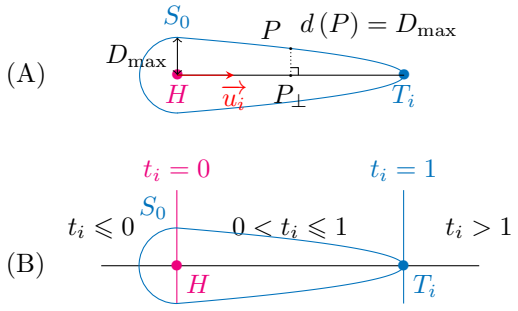

Figure 3: (A): 2D section of the isodistance surface S_0. (B): Representation of t_i, with $t_i(P)$ between 0 and 1 if $P_\perp$, projection of P on the line (HT_i), is in the segment $[HT_i]$.

- $d_i(T_i) = D_{\max}$,
- $d_i(H) = 0$,
- the surface has the shape of a bottleneck.

The shape of the surface implies that the distance d_i to the segment $[HT_i]$ is piecewise-defined, and that $d_i(P)$ depends on the coordinate of P along the axis (HT_i). Thus, for each point P, we define a scalar $t_i(P)$ as follows:

$$t_i(P) = \frac{\overrightarrow{HP_\perp} \cdot \overrightarrow{u_i}}{HT_i}, \quad (3)$$

where $\overrightarrow{u_i}$ is a unit direction vector of the line (HT_i), and $P_\perp$ is the orthogonal projection of P on this line. This function verifies $t_i(P) = 0$ if $P_\perp = H$, and $t_i(P) = 1$ if $P_\perp = T_i$. These conditions are represented in figure 3B. For $t_i \leqslant 0$, all points of the surface S_0 have the same euclidean distance to H. In this area, the surface has the shape of an half sphere. For $0 < t_i \leqslant 1$, the surface has the shape of a bottleneck.

From the definition of t_i results the function d_i, representing the distance to the segment $[HT_i]$, or *branch* B_i:

$$d_i(P) = \begin{cases} \dfrac{(HP)^2}{D_{\max}} & \text{if } t_i(P) < 0, \\ \dfrac{(PP_\perp)^2}{D_{\max}} + D_{\max} \cdot t_i(P) & \text{otherwise.} \end{cases} \quad (4)$$

When the skeleton has more than one branch, the distance to the skeleton of a point is defined as its smallest distance-to-branch value, as explained in equation 2. As a result, it looks like branches are merged. Figure 4 shows a representation of this surface for a multi-segment skeleton.

2.2.3 *Displaying the surface*

In Move mode, the user must ensure that the target is captured by Starfish. We therefore need to display the surface S_0. A Marching Cube algorithm[8] is first applied on the potential field to produce a triangle mesh. Then, a simple implementation consists in displaying the triangles of the mesh with only one color, semi-transparent to see the head. However, we believe that this may be not sufficient to distinguish between branches, and their directions. Some

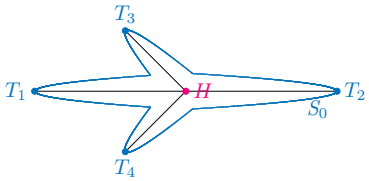

Figure 4: 2D section of the surface generated by Starfish from a skeleton with four segments.

preliminary tests led us to another procedure to color each vertex of the mesh. Basically, the idea is to display a fixed number of stripes for each branch, with a color gradation, as shown in figure 5. We assume that the color gradation helps to disambiguate branches, and the stripes help to understand their orientations. In Select mode, a different color function is used to visually inform the user that the mode has changed. Moreover, the branch containing the captive pointer is highlighted.

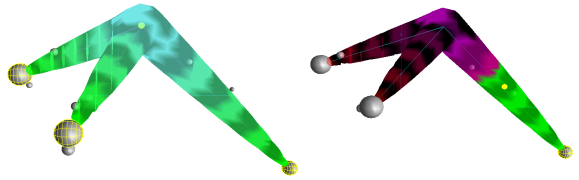

Figure 5: Starfish in Move mode (left) and in Select mode (right).

2.3 Constraints on the pointer

In Select mode, the surface is locked, and we want the head of Starfish to be constrained within the surface, so that it can slide on it, and stop on the desired target (we say that the head becomes a *captive* pointer). Obviously, we cannot prevent the user's hand to cross the surface. To get around this, the position of the user's hand is associated with a second pointer, called *free* pointer, which stays invisible. Initially, when Starfish switches to Select mode, the free pointer is at the same position than the captive one. While both are inside the surface, their positions are identical. When the free pointer moves outside, the captive pointer has to slide along the surface to follow the free pointer.

A simple way to obtain this behaviour is to place the captive pointer on the point of the surface nearest to the free pointer. However, this may involve involuntary "jumps" between two branches, that should not be allowed. The user chooses to enter a branch, then the bottleneck shape ensures that either the captive pointer ends on the target, or the user changes his mind and has to move back to the center to choose another branch. The captive pointer should never jump from branch to branch. We propose a method to follow the surface, based on small successive movements in directions tangential to the surface.

Let P_C be the position of the captive pointer, P_F the position of the free pointer, and $\overrightarrow{D} = \overrightarrow{P_C P_F}$ the direction of the desired movement for the captive pointer. Let also $\overrightarrow{N_s}$ be the normal to the surface at P_C. $\overrightarrow{N_s}$ is easily computed thanks to the gradient of the potential field.

Algorithm: Following the surface

While $\overrightarrow{D}$ and $\overrightarrow{N_s}$ are not collinear:

- The captive pointer is translated by a vector $\delta \cdot \overrightarrow{N_s} \wedge \left(\overrightarrow{D} \wedge \overrightarrow{N_s}\right)$, where δ is an arbitrary small factor.
- The captive pointer is then moved back to the surface, iteratively adjusting its position in the direction of the gradient of the potential field.

EndWhile

3. CONCLUSION AND FUTURE WORK

In this paper, we proposed a new interaction technique for target selection in VR environments, called Starfish. This technique builds a closed implicit surface around the pointer to preselect near targets. Once locked, the surface constrains the pointer, so it can easily slide towards and select the desired target without requiring much precision.

From a preliminary study, Starfish has been deemed comfortable and "fun" to control by the participants, while its performance in terms of selection time and error rate seems promising. Thus, a more formal and complete experience is being developed, to compare Starfish with SQUAD [7].

4. REFERENCES

[1] C. Armbruster, M. Wolter, T. Kuhlen, W. Spijkers, and B. Fimm. Depth perception in virtual reality: distance estimations in peri- and extrapersonal space. *Cyberpsychol Behav.*, 11(1):9–15, 2008.

[2] J. F. Blinn. A generalization of algebraic surface drawing. *SIGGRAPH Comput. Graph.*, 16(3):273–, 1982.

[3] D. A. Bowman, E. Kruijff, J. J. LaViola, and I. Poupyrev. *3D User Interfaces: Theory and Practice.* Addison Wesley, 2004.

[4] J. Cashion, C. Wingrave, and J. J. LaViola Jr. Dense and dynamic 3d selection for game-based virtual environments. *IEEE Trans. on Visualization and Computer Graphics*, 18(4):634–642, 2012.

[5] C. Essert-Villard and A. Capobianco. Hardborders: a new haptic approach for selection tasks in 3d menus. In *Proc. of ACM Symp. on VR Software and Technology*, VRST '09, pages 243–244, 2009.

[6] J. Grosjean, J.-M. Burkhardt, S. Coquillart, and P. Richard. Evaluation of the command and control cube. In *Proc. of Int. Conf. on Multimodal Interfaces*, ICMI '02, pages 473–, 2002.

[7] R. Kopper, F. Bacim, and D. A. Bowman. Rapid and accurate 3d selection by progressive refinement. In *Proc. of IEEE Symp. on 3D User Interfaces*, 3DUI '11, pages 67–74, 2011.

[8] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, Aug. 1987.

[9] L. Vanacken, T. Grossman, and K. Coninx. Multimodal selection techniques for dense and occluded 3d virtual environments. *Int. J. Human-Computer Studies*, 67(3):237–255, 2009.