

# IntenSelect: Using Dynamic Object Rating for Assisting 3D Object Selection

Gerwin de Haan, Michal Koutek and Frits H. Post

Data Visualization Group, <http://visualization.tudelft.nl>  
Faculty of Electrical Engineering, Mathematics and Computer Science  
Delft University of Technology, The Netherlands

---

## Abstract

*We present IntenSelect, a novel selection technique that dynamically assists the user in the selection of 3D objects in Virtual Environments. Ray-casting selection is commonly used, although it has limited accuracy and can be problematic in more difficult situations where the intended selection object is occluded or moving. Selection-by-volume techniques, which extend normal ray-casting, provide error tolerance to cope with the limited accuracy. However, these extensions generally are not usable in the more complex selection situations. We have devised a new selection-by-volume technique to create a more flexible selection technique which can be used in these situations. To achieve this, we use a new scoring function to calculate the score of objects, which fall within a user controlled, conic selection volume. By accumulating these scores for the objects, we obtain a dynamic, time-dependent, object ranking. The highest ranking object, or active object, is indicated by bending the otherwise straight selection ray towards it. As the selection ray is effectively snapped to the object, the user can now select the object more easily. Our user tests indicate that IntenSelect can improve the selection performance over ray-casting, especially in the more difficult cases of small objects. Furthermore, the introduced time-dependent object ranking proves especially useful when objects are moving, occluded and/or cluttered. Our simple scoring scheme can be easily extended for special purpose interaction such as widget or application specific interaction functionality, which creates new possibilities for complex interaction behavior.*

Categories and Subject Descriptors (according to ACM CCS): H 5.2 [User Interfaces]: Interaction Styles; I 3.6 [Methodology and Techniques]: Interaction Techniques; I 3.7 [Three-Dimensional Graphics and Realism]: Virtual Reality

---

## 1. Introduction

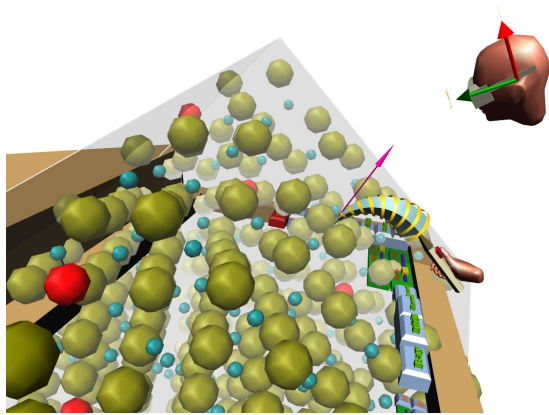
Virtual Environments have the potential to increase the insight into complex multi-dimensional models and processes from a wide range of applications. However, VR applications still are not as widespread as their potential would let us expect. The lack of its general acceptance and use is often attributed to the limited flexibility and practical usefulness of the user interface, even though this is considered as an advantage of VR. One of the most basic elements in the complete taxonomy [BWCL01] of three-dimensional interaction is that of *object selection*. That is, indicating one specific object in the virtual world. When examined very closely, even this basic task of object selection is often surprisingly difficult and can be a source of irritation for the users. We there-

fore believe that the practical usefulness and acceptance of a wide range of interactive VR applications heavily relies on effective selection techniques.

We mainly employ Virtual Reality techniques as a mean for enhancing the interactive process of scientific visualization and data exploration. In addition to rendering techniques for large datasets, we concentrate on one of the main challenges of scientific visualization in VR as described in the overview by Van Dam et. al.: "... to make interaction comfortable, fast, and effective" [DFL\*00]. Currently our main VR applications focus on the visualization and control of (real-time) simulations of physical processes. We often have to deal with objects dynamically moving through the dataset, such as atoms or glyphs (see Figure 1). The researchers in-

volved are interested in the various physical properties, the behavior and context of these objects. To explore these, the object of interest usually first has to be selected.

Nearby objects can be directly selected by manually positioning the spatial interaction device precisely inside its observed graphical representation. This is the so-called *touch* or *direct* selection. For the selection of more distant virtual objects that are beyond arms reach, remote interaction techniques can be employed, of which *ray-casting*, or *laser pointing*, is most commonly used and widely spread. Here, a virtual ray or laser beam emits from the tip of the interaction device, and, by finding an intersection of this ray with an object in the scene, a selection is made. The intuitiveness and low learning curve make the ray-casting technique a very attractive selection technique for many applications. As we also use this ray-casting technique in our applications most of the time, the effectiveness of this selection technique is of great interest to us. Usually, the techniques described above allow effective and fast interaction with objects in the scene. In many cases however, we see the difficulties even very skilled users experience in selecting both nearby and remote objects, leading to confusion or frustration. From casual observations we detected that most difficulties are experienced when attempting to select a certain object that is small, thin or remote; occluded or in a cluttered scene; or showing complex behavior. We have devised a new interaction method designed to cope with these three issues in object selection, with the overall goal of assisting the user in selecting the intended object.



**Figure 1:** Example application: Selecting a moving atom in a heavily cluttered Molecular Dynamics environment.

This paper is constructed as follows: First, we describe the three aspects of the problem in more detail. Then, we discuss previous work in the area of ray-based and selection-by-volume techniques, as well as other assisting selection techniques for VEs. We continue with a general overview of our algorithm and a more detailed description of our scoring metric. After a description of the implementation of our

technique we report on our general impressions on usability. In addition to this, we present the results of an informal user study and compare the performance of our selection technique with other existing techniques. Finally, we discuss our findings and thoughts, and give some pointers for future research and applications.

## 2. Problem Analysis

### 2.1. Selection accuracy: Small and remote objects

The virtual pointer and the attached selection ray are usually controlled by a tracked six-degree of freedom interaction device. The user can position the device to control the origin of direct interaction, while the device rotations control the direction of the ray for remote interaction. For remote interaction the angular control of the virtual pointer dominates over the positional control because angular movements result in amplified movements of an object intersection point at a further distance. In order to accurately pinpoint a distant object using a ray, the angular control needs to be accurate and stable. Although in the last decade tracking quality has increased through improved hardware, filtering and calibration techniques, the interaction accuracy remains negatively influenced by "imperfections" of the human motor functions such as hand jitter and fatigue. Tracker calibration can also be a cause. In non- or poorly calibrated head-tracked environments the misalignment of physical device and virtual selection point hamper intuitive pointing. Even a slight offset can lead to confused or frustrated users, resulting in extra efforts in order to select a virtual object. Often *clutching* is involved, that is, users are repeatedly switching back and forth between navigation, manipulation and selection modes to obtain the desired situation in which they can perform their intended action, such as selecting one specific object. Even if tracking and calibration were perfect, as are in the case of a real physical laser-pointer, it remains a tedious and time consuming task to accurately pinpoint a small object at a large distance. Considering these aspects we believe the previously described discreteness of selection condition should be abandoned for a more error-tolerant approach.

### 2.2. Selection Ambiguity: Occlusion and Cluttering

In many situation where users attempt to indicate or pinpoint something, the object is typically between other objects. In crowded environments there is a high probability that an object is near other objects (*cluttering*). When pointing at a remote object using ray-casting there is also a high chance that another object is in the path of the ray and triggering an un-intended selection (*selection occlusion*). By changing the angle and position of the ray or zooming in on the scene, a free line of selection might be found to avoid this occlusion state, but this can be unsuccessful and difficult in a crowded scene. We consider this clutching an unwanted side-effect of the interface, and we believe that a more elegant alternative should be investigated.

### 2.3. Selection Complexity: Moving Objects

In many selection tasks, the previous two issues can be overcome by spending more time and effort to assure a correct selection. However if the scene is not static and virtual objects or the user show more complex behavior, such as motion, the selection task itself becomes more difficult and exacerbates the previous issues. It is, for example, difficult to accurately pinpoint a small moving target because occlusion and cluttering effects dynamically change. Additionally, there might not be an opportunity to spend more time on the task. Situations where this complex behavior can be seen are, for example, fast moving objects, or objects which are only present for a short period of time as is often the case in real-time simulations or collaborative VR applications. These situations only increase the required effort to correctly select an object, and consequently increase the user's demand for a fast and accurate selection technique. We believe that, in time-dependent situations, we have to provide an interface to cope with the dynamic aspects of the scene.

Both in the real world and in VR it can be difficult to accurately and quickly pinpoint an object, regardless of which selection technique is used. The many degrees of freedom in the interaction allow a myriad of possibilities. The advantage that we have in VR is that the scene-generating software generally has access to many, if not all, aspects of the virtual objects that populate it, as well as all interaction data. When pinpointing in a VE, the degrees of freedom might be strongly reduced by reasoning on this available information, and constraining the selection to only the most likely or intended objects. If possible, the intended selection could be recognized and highlighted to assist the user in the selection task. This *automated assistance* could increase the selection performance over pure manual control, especially in the more complex cases.

### 3. Related Work

Ray-casting selection has been commonly used in all kinds of VR systems, ranging from CAVEs to desktop VR systems [DPRHP99]. Although this technique provides a way to select remote objects, its practical use is often awkward; classic ray-based interaction has proved to be difficult in selecting objects which are small, occluded and/or moving. As discussed in the previous section, ray-casting requires accurate and stable pinpointing. For a useful selection, this implies that the user can accurately point the virtual ray directly at the object and keep the object selected while performing an action such as pressing a button. A selection to an object can easily be lost as soon as you click the button, nicknamed the Heisenberg effect [BWCL01].

The spotlight selection technique is one of the first extensions of the classic ray-casting selection technique [LG94]. It was presented as a novel selection technique in a desktop-based 3D modeling system with a six-degree of freedom interaction device. To allow a certain level of error in aiming

at small and distant objects the mathematical discreteness of the normal ray is "softened". A cone shaped selection volume is attached with its apex to the origin of the co-ordinate system controlled by the interaction device. Objects that fall within this selection volume are potential candidates for selection and are awarded a score, based on a disambiguation metric. The highest ranking object is highlighted as being the selected object. In the aperture selection technique [FHR96], which is based on spotlight selection, the apex of a conic selection volume is attached to the dominant eye whereas the direction vector of this cone is directed from that eye through the pointer's tip. Attached to the pointer is a circle with fixed radius, which determines the cross-section radius of the cone at the pointer location. By moving the circle towards or away from the eye the user can respectively increase or decrease the size of the selection cone. The eye position and this circle span the cone, and thus interactively determine its size and orientation. For resolving selection ambiguity, a metric similar to the one in the spotlight technique is used. In both techniques the conic selection volume is displayed and the current selection is highlighted. In more recent work, the Improved Virtual Pointer (IVP) is described, using a similar selection-by-volume approach to determine the active object [SRH05]. This technique uses a different scoring metric and uses a flexible ray to visually connect the pointer to the active object.

In order to "automatically" determine the intended object within a selection volume, a balanced metric has to be constructed for an intuitive ranking of the objects. An alternative to using a metric is using additional actions or devices for manual disambiguation, such as twisting the virtual pointer or using a separate button to toggle through selections. Another example is the use of a pop-up dialog containing a list with all the preselected objects [Dan05]. By selecting an item from this menu, any object within the selection volume can be selected. Although this allows a guaranteed selection of an object, we feel that continuously switching from 3D interaction to menu selection can hamper the user's work flow. The nature of the application and moreover the importance of selection correctness ([Dan05] describes air-traffic control) will be decisive on the usefulness of this type of interaction.

Another approach uses artificial intelligence techniques to determine the intended object during selection [WBR01]. Here, machine learning is used to detect users' nuances and preferences and to reason with interface information in an attempt to elicit their intentions. The attempts to create intelligent interfaces have shown the feasibility and effectiveness of this approach, but they are currently limited to simple cases and are not yet ready for practical use. In addition, design requirements are presented for such a system, derived from fields such as gesture recognition and automated user interface design. Although we do not use a similar system, it can be of interest to validate the "dynamic" behavior of our algorithm to these requirements.

In our method we attempt to limit both the previously mentioned explicit actions or clutching movements and the use of extra or specialized interaction devices. We limit our technique to only using only the position and the pitch and heading (tilt and azimuth) of a 6 DOF pointing device, equipped with one button for indicating the selection. In this way, our technique is more generic and can be more easily applied to a wide range of VR systems and interaction devices. Furthermore, we want the selection behavior to be similar to ray selection to reduce the need for extra user training.

## 4. Selection Algorithm

### 4.1. Overview

In this section we will describe our object selection algorithm and the differences between this and existing techniques. Our algorithm consists of the following general stages:

- **Selection volume test:** Determine which objects are inside the conic selection volume.
- **Score contribution:** Objects inside the volume get a scoring contribution, determined by a scoring metric.
- **Score accumulation:** Contributions are accumulated at the objects over time; objects with a score are lowered.
- **User feedback:** The highest ranking object is highlighted and indicated by snapping a bent ray to it.

### 4.2. Selection Volume Test

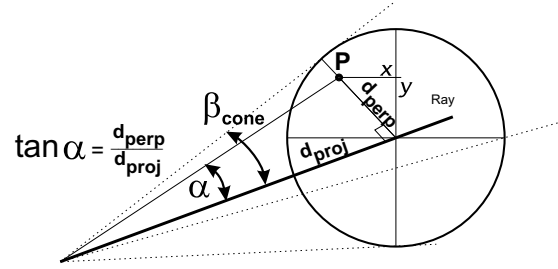
A cone shaped selection volume is attached with its apex to origin of co-ordinate system controlled by the interaction device. The conic volume is determined by the position and orientation of the pointer's tip, its orientation and a spread angle. First, we determine which objects are within the range of this volume. For every object in the scene, every frame, we perform an "inside"-test of the object and the conic volume. In our current proof-of-concept we limit our test to the bounding sphere of an object, which is typical for the objects in our user tests. We transform the object's center point position and radius to the co-ordinate system of the pointer. In this co-ordinate system we can easily derive the projection distance on the ray ( $d_{perp}$ ) and the distance from this projected point to the pointer's position ( $d_{proj}$ ). Figure 2 shows a three-dimensional schematic overview of object P inside the conic volume. Point P lies on a two-dimensional cross section of the cone. This is a circular surface, with the ray defining both its center point and normal, while the radius is determined by the cone spread angle and distance  $d_{proj}$ .

$$d_{perp} = \sqrt{x^2 + y^2} \quad (1)$$

$$d_{proj} = z \quad (2)$$

If the angle  $\alpha$ , defined by these two distances, is smaller than the opening angle of our cone ( $\beta_{cone}$ ), we consider the object

to be inside of the conic volume. For these objects a scoring value will be calculated, while other objects will be ignored in the following stage.



**Figure 2:** Selection volume test: Determining whether point *P* is inside the conic selection volume.

### 4.3. Score Contribution

The scoring contribution of the objects that are inside the selection volume is determined by a scoring metric. A scoring metric is a formula that assigns a scoring value to an object, based on the properties of the object. As described earlier in Section 3, several metrics can be used. For example, in [LG94] and [FHR96] an ellipsoidal distance metric is used. Although this metric is designed to be tolerant to aiming inaccuracies, it is still hard to select occluded objects or far away objects when objects are near. In the IVP technique [SRH05], the absolute projection distance similar to ( $d_{perp}$ ) is used as a metric.

To ensure a seamless and effortless transition from ray-casting selection behavior, we take simple ray-casting as a basis for our scoring metric. To achieve this, we assign the highest score of 1 if pointing is most accurate. That is, the ray is being pointed through the object's center point. Furthermore, we assign the lowest score of 0 if pointing is least accurate, when the object's center point lies on the bounding surface of the conic volume. We achieve this scoring by using the angle  $\alpha$ , defined by the distance perpendicular to the ray ( $d_{perp}$ ) and the distance from this projected point to the pointer's position ( $d_{proj}$ ), as the main variable of the metric.

$$s_{contrib} = 1 - \frac{\alpha}{\beta_{cone}} \quad (3)$$

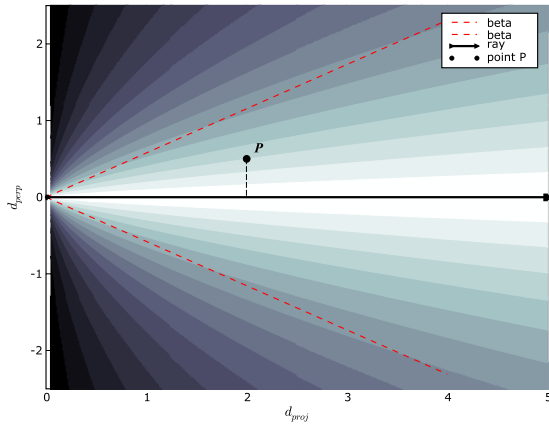
During testing trials it quickly became obvious that, using this scoring metric, it is easier to select distant objects than it is to select nearby objects. It became clear that the use of this angle  $\alpha$  as a direct metric implies larger  $d_{perp}$  at larger  $d_{proj}$ , and thus tolerates large aiming errors. In more detail, as the  $d_{perp}$  is  $\sqrt{x^2 + y^2}$ , the surface that is inside the cone at larger distance allowed is growing exponentially. Although this is in essence desired behavior, it feels counter-intuitive. To compensate for this exponential factor,  $d_{proj}$  is taken to

the power  $k$ , a compensation constant:

$$s_{contrib} = 1 - \frac{\text{atan}\left(\frac{d_{perp}}{(d_{proj})^k}\right)}{\beta_{cone}} \quad (4)$$

Here we typically choose  $k$  to be between  $\frac{1}{2}$  (being a linear relation) and 1 (being the original relation). Currently we have chosen  $k$  to be  $\frac{4}{5}$ , based on initial experiments. We acknowledge that this compensation scheme and the compensation factor is dependent on the type of application and user, and can be of interest in further research.

The resulting scoring metric inside the conic volume is shown in Figure 3. This figure represents a two-dimensional section of the conic volume with the pointer located in the origin. The black line indicates the ray, while the red dotted lines represent  $\beta_{cone}$ . The gray-scale intensity is used to display the scoring value, white being 1 and gray 0.



**Figure 3:** Object Scoring Function: Scoring value of point  $P$  inside the conic selection volume.

#### 4.4. Score Accumulation

In previous work the score determines the disambiguation of the selection directly. That is, the position of the pointer and the state of the objects at a point in time (e.g. a certain frame) directly determine which object is selected. This per-frame static behavior can lead to fast switching between objects when they have similar scores, for example in the case of cluttered or moving objects. Instead of this static behavior we aim at a more dynamic system behavior that can use temporal relation of both objects and interaction. By using this history we hope to create a more meaningful response to the user's actions.

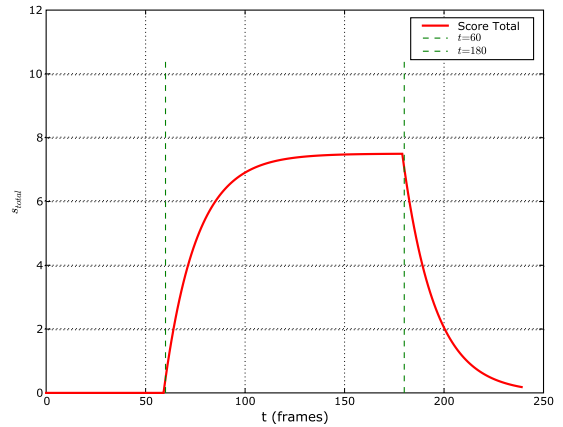
To achieve this temporal effect we don't use the score directly, but the scoring contribution is accumulated over time. Each object maintains its score over several time frames. That is, the score is not reset at every frame, but kept during a sequence of frames. In addition we use a progressive

reduction of the scores of all objects, in effect fading out the scores that have been accumulated in the previous frames. The behavior can be described by the following formulas:

$$s_{contrib}(t) = 1 - \frac{\text{atan}\left(\frac{d_{perp}(t)}{(d_{proj}(t))^k}\right)}{\beta_{cone}} \quad (5)$$

$$s_{total}(t) = s_{total}(t-1)c_s + s_{contrib}(t)c_g \quad (6)$$

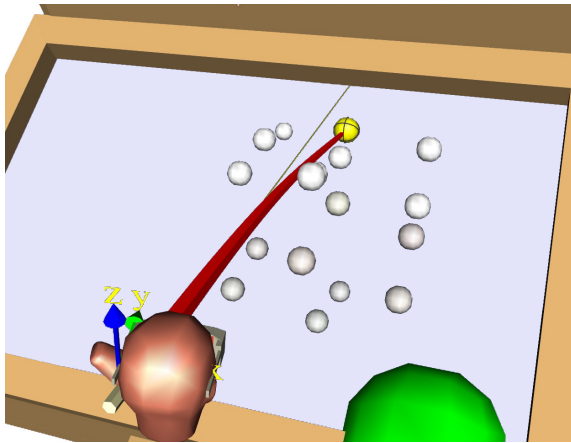
The contribution score is left unaltered but denotes the score at the current time step (or frame)  $t$ . The total accumulated score  $s_{total}$  is defined by the total score of the previous time step ( $t-1$ ) and the current contribution score. Both variables are multiplied by constant scaling factors,  $c_s$  and  $c_g$ . The constant  $c_s$  defines the rate of decay or shrinking of the total score. The constant  $c_g$  defines the rate of growth of the score for a selected object. These constants typically define the *stickiness* and *snappiness* of the selection, respectively. If we choose  $c_s = 0$  and  $c_g = 1$  we get the regular static volume selection without the time-dependent behavior. As these values typically describe a complex trade-off between snappiness and stickiness, their correlation and their influence on the time-dependent selection behavior is not straightforward. Currently, we have chosen these parameters manually based on our initial experiments, balancing both parameters to a comfortable response. In Figure 4 the scoring response of a single object in time is shown. Until frame  $t = 60$  the object is not inside the selection volume, thus receiving no score. From frame  $t = 60$  and later, the object is accurately selected causing the per-frame scoring values to accumulate at the total score. At frame  $t = 180$ , the object is again outside the selection volume. Here the score is only influenced by the decay defined by  $c_s$ . It must be noted that in practical use the selection will gradually enter and leave the conic selection volume, resulting in a more complex scoring response than shown in the figure.



**Figure 4:** Score Accumulation: Time-dependent scoring behavior for a single object.

#### 4.5. User Feedback

The object that has the highest accumulated score  $s_{total}(t)$  at the current frame  $t$  is marked as the *intended* or *active* object. If no object has a score higher than zero, no active object is defined. In this case the normal ray is displayed, allowing users to fine-tune their pointing. If an object is marked as the intended object, the end of the selection ray is bent towards it. The bending ray is an extension of the family of our Virtual SpringTools [KP01], and uses a Bézier curve to determine the geometric shape. The bending is done in such a way that the ray emits from the pointer and the endpoint *snaps* to the object, see Figure 5. As long as this object remains the currently activated, the connecting bent ray between it and the virtual pointer is maintained. To achieve this, the shape of the ray is continuously updated to accommodate movements of both pointer and the active object. If another object is activated, the ray will snap immediately to it.



**Figure 5:** *IntenSelect* in action: The user is selecting an occluded, moving and far-away object. A thin line indicates the pointing direction, while the bending ray remains snapped to the highest ranking object.

Although we have used the coloring of objects as well as displaying the conic selection volume and several transparent rays deforming towards a number of lower ranking selection objects, we have the impression that these all lead to unnecessary visual clutter and distract the user's attention. Therefore we limit our visual feedback to the bending of the ray and simultaneously show a thin ray that indicates the pointer's aiming direction.

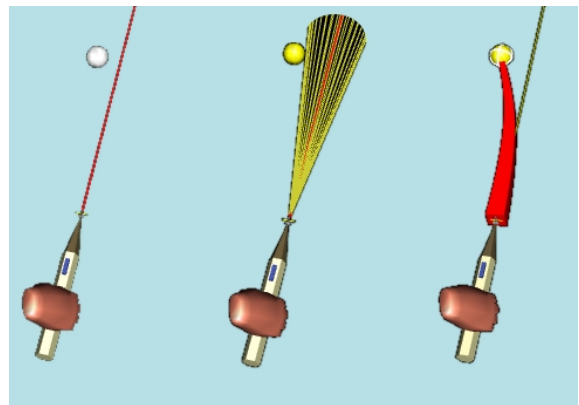
The resulting behavior feels similar to that of ray-casting, only with snapping in less accurate selections. A good effect of the temporal relation can be seen in Figure 6. We have recorded a user session (see Section 5) where ray-casting is used to select a moving object as accurately as possible. Using this technique, the user frequently *loses* the object. When the same session is replayed and the volume-based selection is used, the object is selected most of the time. Only when

the object falls just outside the conic selection volume the selection is lost for a moment. When replaying the session with *IntenSelect*, the temporal filtering maintains the selection the entire movement.

We must note that many of the parameters and techniques shown here can have many variations and can be optimized in various ways to adjust behavior and maybe improve performance. In our current applications we have not noticed significant performance penalties. Nevertheless, if performance issues would arise in more complex scenes, various optimization techniques can be employed.

#### 4.6. Flexibility and Extensibility

In addition to the *default scoring metric* for simple virtual objects, we have the possibility to assign *special-purpose scoring functions* to customize the behavior of the selection interface. By using user-defined object callbacks in the scoring function we can enhance or redefine specific scoring distributions resulting in different selection behavior. This specialized selection behavior can be customized for special virtual objects or on a per-application basis, allowing application developers to tune the interaction performance for specific purposes. In this way we can facilitate improved selection behavior of items of special interest such as buttons and sliders in the user interface. Another example would be the selection of a door in an architectural model. The door itself could redirect its scoring contribution to the door handle (which itself is more difficult to ray-cast), directing the selection and the user's attention to the more interesting or functional virtual objects.



**Figure 6:** *Selecting a moving object:* Comparing a frame from a recorded session of (a) ray-casting (b) volume selection, with visible conic selection volume, and (c) *IntenSelect*. Moving objects can easily fall just outside the selection volume for a short period of time.; see animation.

## 5. Implementation

We run our applications on our Responsive Workbench (RWB), a tabletop projection-based VR system. We use dual DLP projectors to display a stereoscopic image of 1400 x 860 pixels at the glass plate with 180 x 110 cm display dimensions. A Polhemus Fastrak electromagnetic tracking system is used to track our interaction devices.

We use our RWB-Library [Kou03], a custom VR software library built on top of the OpenGL Performer scene graph API. The library provides basic interaction functions for selection and manipulation and contains a set of 3D widgets such as buttons and sliders, and handles all necessary tracking and display functionality. The library uses a default RWB-interactor class to allow selection and manipulation of these RWB-objects directly with the stylus or by using ray-casting. It checks for bounding volume or ray-casting selections, providing the intersection and interaction points (stylus/ray with objects). This class works on the principle of an interaction state machine, driven by interaction events.

The IntenSelect technique is currently implemented as a special RWB-interactor class, and allows easy integration with existing RWB-applications. Instead of checking for bounding volume or ray-casting hits, the interactor determines the ranking of RWB-objects in the scene at every frame. If a new highest ranking object is found, its touch callback function is called. The previously winning object is deselected by calling its un-touch callback. The selection ray is bent to snap to the center of the bounding volume of the highest ranking RWB-object. If the user presses the stylus button, the object is selected and the pick callback is called. As long as the button is pressed, the object ranking is not updated. The interactor will be in the manipulation state. To maintain the connection to the object being manipulated, the bending ray is being continuously updated, as described in Section 4.5.

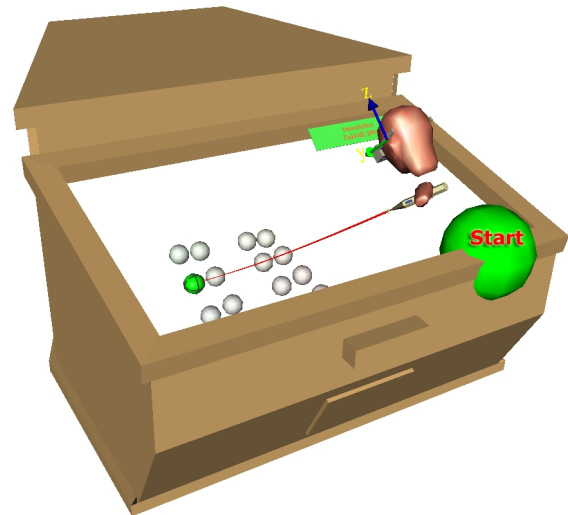
## 6. User study

To verify our technique we have conducted a small informal user study. In this setup we compare IntenSelect with regular ray-casting technique and our volume-based selection without temporal accumulation. We use the same display and interaction hardware without re-calibration between tests. Furthermore we have fixed the frame rate to 60Hz, leading to a timing resolution of 16.67 ms. After a small introduction to the VR system, the user starts out with a 2-minute training session to get acquainted with the interaction styles and the display environment. This is followed by a series of small tests in which selection time is measured.

### 6.1. Test setup

In our test scene the user can use the selection technique to select single objects in the scene. A test scene consists

of a pre-generated set of simple virtual objects which are both similarly shaped and colored. First, the subject holds the pointer inside the *starting sphere*, which is a large object located at a comfortable position on the right-hand side of the scene, see Figure 7. Once this starting sphere is selected, one of the objects from the set is highlighted by a distinctive color. As long as the stylus is inside the starting sphere, no ray information is displayed nor the task time has started. We instruct the user to first find the object and only then initiate the selection. In this way we exclude the finding activity of the subject from the timing, and prevent coincidental hidden objects. As soon as the user has visually located the highlighted object, the pointer can be moved from the starting box. At this moment, the used selection technique is activated and the timer starts ticking. As soon as the user selects the highlighted object by pointing at it and pressing the stylus button, the timer is stopped. The user is then instructed to return the pointer into the starting box. Now, another object is highlighted and the sequence starts from the beginning until the test is complete.



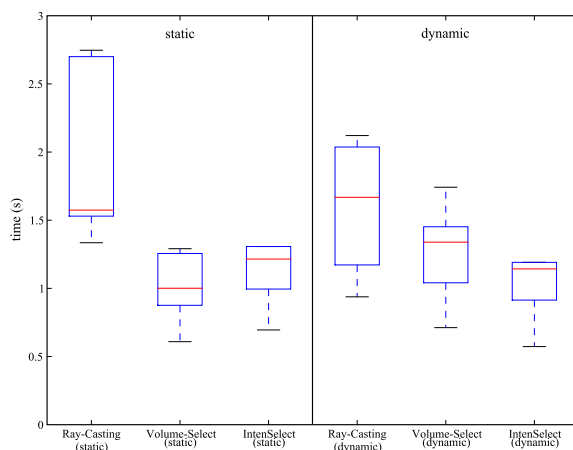
**Figure 7:** *User Test Scene: Using IntenSelect to select a moving object amongst other moving objects. The spheres make fast periodic movements during which they continuously intersect, clutter and occlude each other.*

We have created two scenes, one with small static objects and one with larger moving objects. The static scene is created to test only the selection performance of small, distant objects, of which some problems were described in Section 2.1. It consists of an array of 8x8 small spheres at a distance of approximately two meters inside the workbench. The moving scene consists of 16 larger spheres which quickly follow a predefined periodic path through the scene. During these movements, the objects continuously intersect, clutter and occlude each other. The moving scene tries to mimic real applications with complex moving objects and occlusion, see Sections 2.2 and 2.3.

The generated scenes are created off-line and are accompanied by a predefined sequence of objects that are to be selected. Each subject uses the three different selection techniques, ray-casting, selection-by-volume and IntenSelect, on the same selection sequence in both scenes. This leads to a total of 6 small tests per user, each consisting of 10 object selections. We start out with the three interaction techniques on the static scene, followed by the same techniques on the dynamic scene. To avoid some of the training effects in favor of our technique, we start out with the IntenSelect technique, followed by the volume selection and finally the ray-casting technique. If these training effects do occur, they mainly work in favor of the regular ray-casting and volume-based selection most. Nevertheless, users might also reduce their accuracy in pointing in the first trials, reducing the performance in ray-casting.

## 6.2. Test Results

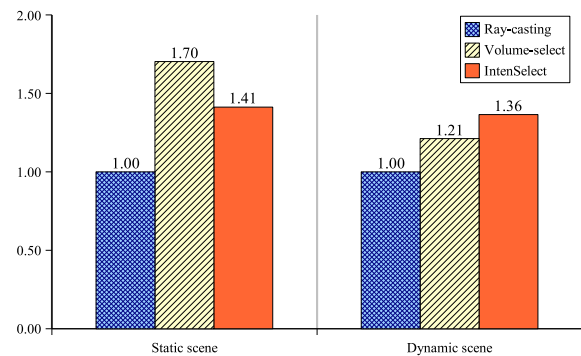
A total of eight subjects have successfully completed all of the six tests, resulting in 480 time values. The test group mainly consisted of male colleagues in the age range of 25 to 30, of which only three had hands-on experience with interaction on Virtual Reality. In Figures 8 and 9 the quantitative results of the tests are presented. In Figure 8, the leftmost three boxes indicate the selection speed in the static scene, while the remaining three indicate the selection speed in the dynamic scene. The y-axis indicates the object selection time, measured in seconds starting from the leaving of the starting sphere to the selection of the object with the pointer. For each box, eight samples are used, each representing the user's averaged object selection time for that session. Each box extends from the lower to the upper quartile value of the samples, while the median value is indicated by the middle red line in the box.



**Figure 8:** Test Results: Box plot of averaged object selection times for the three techniques in static scene (left) and dynamic scene (right).

In Figure 9, a relative comparison is made between the three selection techniques. Here we take the ray-casting technique as the reference technique, and express the performance of the other techniques as a ratio of this reference. First, we average all selection times of all users per session, leaving out the best and worst timing value of each user. Then, we normalize these average timings with respect to ray-casting to obtain relative timing values, and take their inverse to obtain relative speed-ups.

First, it becomes clear that, in both scenes, generally, ray-casting is out-performed by the two improved selection techniques. All users appreciated the helpful assistance in pointing and found the improved selection methods a relief in contrast to regular ray-casting.



**Figure 9:** Test Results: Relative average selection speed-up of volume-selection and IntenSelect compared to the ray-casting technique.

In the static scene, somewhat to our surprise, the volume-based selection technique seems to perform better than the IntenSelect technique with temporal accumulation. The more experienced users reported that, in the case of IntenSelect, the ray felt too sticky. This forced them to wait a small amount of time for the ray to update its selection to the intended object. The less skilled users however did not report any noticeable differences in the two latter techniques, but nevertheless they performed best with IntenSelect. We believe that their generally slower interactions allow the temporal scoring to update quickly enough, effectively showing similar behavior to the normal volume selection. In addition, their inaccurate and jittery aiming probably led to more incorrect selections using the basic volume selection. Furthermore, we believe that if cluttering increases (effectively decreasing error tolerance), the selection with only volume-selection technique becomes more difficult.

In the dynamic scene the IntenSelect was found the most effective technique, which is confirmed by the performance results. The continuous occlusion and cluttering effects hamper the use of the normal volume-selection, and lead to many wrong selections. Individual results show that, in only in one

or two cases, very experienced users performed best using simple ray-casting. These users reported that they had gained experience from the previous sessions. Typically, they had learned to wait for the right moment in the periodic movement of the objects.

Although our initial tests show a significant increase in average performance, it is too early to make decisive quantitative statements on this. A larger and more formal test setup is required to achieve a more reliable results and to perform statistical evaluation. For a more reliable test and an elimination of any unwanted learning effects, we should, for example, increase the number of samples, the number of users, and use only one technique per user. Furthermore, we need to gain more insight in the complex influence of the individual object and scene characteristics on the performance of our technique.

## 7. Discussion and Future Work

In this paper we have discussed the design of an improved ray-selection technique for object selection in VR. We have tested this technique in comparison with others, on the Responsive Workbench. The results of the user test indicate that our technique is more effective in selecting small objects, especially when these are moving and when clutter or occlusion effects occur. Moreover, all users appreciated the helpful assistance in pointing. Before a more definite statement can be made though on the value of this technique, a number of factors will have to be considered. First, we have used a simplified testing set of virtual objects, consisting of only spheres. Second, we have to compare the technique to other, also non ray-casting based selection techniques. Finally we have to extend our current user test to a larger set of test-subjects and more narrow tasks to gain more detailed insight in the various parameters, their correlation and their impact on user-performance. In addition, we will have to evaluate our technique on a wider range of systems with various display and tracking hardware, including our PowerWall, desktop VR systems such as the Personal Space Station [MvL02], and the CAVE.

As we briefly described in our introduction, our main VR applications focus on the domain of data visualization and control of physical simulations. Our next goal is to apply the IntenSelect technique in these applications. We plan to enhance general application control by applying a special scoring function for 3D widgets such as buttons. In addition, we will apply the IntenSelect technique and required application specific optimizations to the selection of application objects, such as atoms or glyphs.

We have observed the powerful effect of *hinting* gestures when users follow a moving object or try to disambiguate between two remote, adjacent objects. The temporal effects of the scoring function allow these more advanced, time-dependent user interactions. The flexibility of the various

scoring functions and their intricate time-dependent relations open up a myriad of opportunities for specialized interaction behavior.

## References

- [BWCL01] BOWMAN D., WINGRAVE C., CAMPBELL J., LY V.: Using Pinch Gloves for both Natural and Abstract Interaction Techniques in Virtual Environments. In *Proc. HCI International* (2001), pp. 629–633.
- [DFL\*00] VAN DAM A., FORSBERG A., LAIDLAW D., LAVIOLA J., SIMPSON R.: Immersive VR for Scientific Visualization: A Progress Report. *IEEE Computer Graphics and Applications* (Nov/Dec 2000), 26–52.
- [Dan05] DANG N.-T.: The Selection-By-Volume Approach: Using Geometric Shape and 2D Menu System for 3D Object Selection. In *Proc. of the Workshop on New Directions in 3D User Interfaces, IEEE VR Conference* (2005).
- [FHR96] FORSBERG A., HERNDON K., R. Z.: Aperture based selection for immersive virtual environments. In *Proc. of the Symposium on User Interface Software and Technology* (1996), pp. 95 – 96.
- [Kou03] KOUTEK M.: *Scientific Visualization in Virtual Reality: Interaction Techniques and Application Development*. PhD thesis, Delft University of Technology, 2003.
- [KP01] KOUTEK M., POST F.: Spring-Based Manipulation Tools for Virtual Environments. In *Proc. Immersive Projection Technology and Eurographics Virtual Environments '01* (2001), pp. 61–70.
- [LG94] LIANG J., GREEN M.: JDCAD: A highly interactive 3D modeling system. *Computers & Graphics* 18, 4 (1994), 499–506.
- [MvL02] MULDER J., VAN LIERE R.: Personal Space Station. In *Proc. VRIC '02* (2002), Laval Virtual, pp. 73–81.
- [dPRHP99] VAN DE POL R., RIBARSKY W., HODGES L., POST F.: Interaction Techniques on the Virtual Workbench. In *Proc. Eurographics Virtual Environments '99* (1999), pp. 157–168.
- [SRH05] STEINICKE F., ROPINSKI T., HINRICHS K.: VR and Laser-Based Interaction in Virtual Environments Using a Dual-Purpose Interaction Metaphor. In *IEEE VR 2005 Workshop Proceedings on New Directions in 3D User Interfaces* (2005), pp. 61–64.
- [WBR01] WINGRAVE C., BOWMAN D., RAMAKRISHNAN N.: A First Step Towards Nuance-Oriented Interfaces for Virtual Environments. In *Proc. of the Virtual Reality International Conference* (2001), pp. 181–188.