View-Projection Animation for 3D Occlusion Management ⋆

Niklas Elmqvist *,1, Philippas Tsigas

Department of Computer Science & Engineering, Chalmers University of Technology, Göteborg, Sweden

Abstract

Inter-object occlusion is inherent to 3D environments and is one of the challenges of using 3D instead of 2D computer graphics for visualization. Based on an analysis of this effect, we present an interaction technique for view-projection animation that reduces inter-object occlusion in 3D environments without modifying the geometrical properties of the objects themselves. The technique allows for smooth on-demand animation between parallel and perspective projection modes as well as online manipulation of view parameters, enabling the user to quickly and easily adapt the view to reduce occlusion. A user study indicates that the technique provides many of the occlusion reduction benefits of traditional camera movement, but without the need to actually change the viewpoint. We have also implemented a prototype of the technique in the Blender 3D modeler.

Key words: occlusion management, occlusion reduction, 3D visualization

1 Introduction

Three-dimensional computer graphics provides considerable potential for information visualization [4]. However, there is an increased overhead associated with using 3D over conventional 2D graphics [6,7]. In general, 3D graphics imposes a high cognitive load on users trying to gain and maintain an overview

⋆ This is an extended version of a paper that previously appeared in the ACM Conference on Advanced Visual Interfaces 2006.

* Corresponding author.
Email addresses: nickelm@acm.org (Niklas Elmqvist), tsigas@cs.chalmers.se (Philippas Tsigas).
[1] Present address: INRIA/LRI, Bât 490, Université Paris-Sud XI, 91405 Orsay Cedex, France. Phone: +33 (0)1 69 15 61 97, Fax: +33 (0)1 69 15 65 86.

of the environment, and often cause disorientation, confusion, and sometimes even nausea (see for example [17,26]). One of the central issues behind this high cognitive load is *occlusion*, the phenomenon that nearby objects hide more distant objects in 3D even if the objects are not overlapping in space [8].

Why is occlusion a problem in 3D visualization environments? There are three basic issues. First, and perhaps most importantly, there is a *discovery* problem if an object is occluded, since then the user may never know that it exists. Secondly, even if the user is aware of the existence of an occluded object, there is an *accessibility* problem, since the user will have to move the viewpoint in some nontrivial way in order to retrieve the information encoded in the object. Finally, even if the user is able to discover and access individual occluded objects in the world, the high-level task of spatially *relating* the objects to each other can be difficult. The latter task is of particular importance to visualization applications, where objects may be nodes in a graph or events on a timeline.

In this paper, we explore the occlusion problem in more detail, attempting to build a theoretical model for its causes and parameters and also to identify possible solution strategies. Using this model, we develop an interaction technique for view-projection animation that aims to reduce inter-object occlusion in 3D environments without modifying the geometrical properties of the environment itself, nor the objects in it. The technique allows for smooth animation between the conventional perspective projection mode, which mimics human vision in the real world and is commonly used for 3D visualizations, and parallel projection mode, where the depth coordinate is ignored and objects are assigned screen space according to their actual geometrical size, regardless of their distance to the viewpoint. While this causes many depth cues to become lost and thus some of the visual realism, the relationships between the objects are more important for the information visualization applications we consider in our work, as discussed above.

A formal user study conducted on our technique in relation to traditional perspective projection shows that it provides the same occlusion reduction capabilities of moving the camera around the environment to disambiguate occlusion, but without the need to actually change the viewpoint. This means that users run a lower risk of becoming disoriented when navigating 3D space. They are also significantly more correct when solving tasks using view-projection animation. The cost for this increased correctness is instead significantly longer task completion times; users essentially trade speed for accuracy when using our technique.

This paper begins with a review of existing work in the area (Section 2). We then launch ourselves into a theoretical treatment of the occlusion problem and its human-computer interaction aspects, mapping out the problem space

and its components (Section 3). After that, we describe the view-projection animation technique itself (Section 4). This is followed by an overview of the formal user study we conducted (Section 5), for both exploring the occlusion problem as well as comparing our new technique to normal perspective views, and the results we collected from it (Section 5). We close the paper with a discussion (Section 7) and some conclusions (Section 8).

2 Related Work

While most work on improving the usability of 3D visualizations attacks the higher-level problem of navigation, there also exists a number of papers dealing more directly with object discovery and access in complex 3D environments. The Worlds-in-Miniature technique [25] uses a miniature 3D map of the environment to support both discovery and access, worldlets [10] provide both global overview maps as well as local views optimized for detail, bird's eye views [12] combine overview and detail views of the world, and balloon force fields [9] inflate to separate occluding objects. None of these makes direct use of the view projection to improve perception; however, temporally controlled non-linear projections [23] have been used to great effect in improving navigation and perception of 3D scenes.

Projections are intrinsic to computer graphics, but are mostly limited to linear perspective projections. CAD programs have traditionally made use of parallel projection, often through multiview orthographic projections where two or more views of the same object are shown on planes parallel to the principal axes. Carlbom and Paciorek [5] (see also [18]) give a complete overview of planar geometric projections and their various advantages and disadvantages in different situations. Baldonado et al. [2] describe design principles and common practice for visualization systems employing multiple views.

Recent developments in the area also include multiprojection rendering, where several perspectives are combined into one, mainly for artistic purposes. Agrawala et al. [1] compose views of multiple cameras, where each object is assigned to a specific camera perspective, allowing for creative scene composition akin to the work of famous painters. Singh [22] uses a similar approach, but smoothly combines the multiple viewpoints into an interpolated virtual camera in view space instead of composing the images of disjoint cameras on an image-level. While only slightly related to our technique, these works give valuable insight into the manipulation of projection transforms. Our technique can also be compared to glances [19], with the perspective view as the base view and the parallel view as the glance.

Our projection animation technique allows for interactive manipulation of

camera properties beyond the traditional position and orientation parameters. Prior work in this area includes the IBar [24] camera widget, which is a comprehensive camera control that provides intuitive ways of manipulating these parameters for the purposes of 3D scene composition. Another approach uses image-space constraints to solve for the camera parameters given a number of control points [13]. Our work is again focused on reducing the impact of objects obscuring other objects rather than direct camera control, and the camera manipulations provided by our technique only give additional means to achieve this.

The family of non-pinhole cameras introduced by Popescu et al. is of particular relevance to the technique described in this paper. Occlusion cameras [16] and depth-discontinuity cameras [20] are examples of such cameras applied to image-based rendering where parts of occluded surfaces that are likely to become visible soon are sampled and integrated into the output. In more recent work, the framework is extended to a general non-pinhole camera model [21] that can be used for both computer graphics as well as visualization applications. However, this approach was not primarily developed for occlusion management for visual tasks and has to our knowledge not yet been formally evaluated with human subjects.

Also, the view-projection animation technique described in this paper bears close resemblance to the *orthotumble* technique presented by Grossman et al. [15] and its predecessor, the animated view transitions described in [14], but the purpose of these are primarily for maintaining and understanding 3D models rather than reducing occlusion. In addition, our approach uses a more correct algorithm for the "dolly-and-zoom" effect, whereas the orthotumble algorithm is based on linear matrix interpolation.

Finally, view-projection animation is just one of many high-level approaches to managing occlusion in 3D visualization; see [8] for a comprehensive survey of 3D occlusion management and a taxonomy describing its design space.

3 The Occlusion Problem

The occlusion problem space in 3D environments is defined by the intrinsic *properties* of the environment, their interaction with *human cognition*, the *visual tasks* involved, and the ensuing effects caused by the occlusion. The environment and its geometrical properties interact with human vision, causing occlusion of objects and leading to loss of correctness and productivity. In this section, we give a general introduction to these issues as a background to the view-projection technique presented in this paper.

See our taxonomy paper for more detail on this theoretical framework as well as the complete design space of occlusion management techniques [8].

3.1 Model

We represent the 3D world U by a Cartesian space $(x, y, z) \in \mathbb{R}^3$. Objects in the set O are volumes within U (i.e. subsets of U) represented by boundary surfaces (typically triangles). The user's viewpoint $v = (M, P)$ is represented by a view matrix M that includes the position and orientation of the user, as well as a projection matrix P that includes view parameters such as viewport dimensions, focal length, far and near clipping plane, etc.

A line segment r is *blocked* by an object o if it intersects any part of o. An object o is said to be *occluded* from a viewpoint v if there exists no line segment r between v and o such that r is not blocked. Analogously, an object o is said to be *visible* from a viewpoint v if there exists a line segment r between v and o such that r is not blocked. An object o is said to be *partially occluded* from viewpoint v if o is visible, but there exists a line segment r between v and o such that r is blocked.

An object can be flagged either as a *target*, an information-carrying entity, or a *distractor*, an object with no intrinsic information value. Importance flags can be dynamically changed. Occluded distractors pose no threat to any analysis tasks performed in the environment, whereas partially or fully occluded targets do, potentially causing decreased performance and correctness.

Figure 1 shows a diagram of this basic model formulation. Here we see three objects A, B, and C, the first of which is a distractor and the other two targets. The shaded area represents areas invisible to the user from the current view. It is easy to see in this diagram that A is fully visible (but is a distractor), B is partially occluded, and C is occluded.

A set of viewpoints V is said to be *complete* if there exists no object that is occluded in all of the viewpoints v_i. For instance, from the figure it is clear that the set $V = \{v_0, v_1\}$ is complete for the simple environment given in the example (in fact, for this simple situation, it is possible to find a single viewpoint from which all objects are visible).

It is possible to introduce a temporal dimension to this model and discuss concepts like transient occlusion and invariant occlusion. We will ignore this aspect in this thesis, however, and consider only temporally invariant situations. Some of the solutions we develop will still be applicable to dynamic situations.

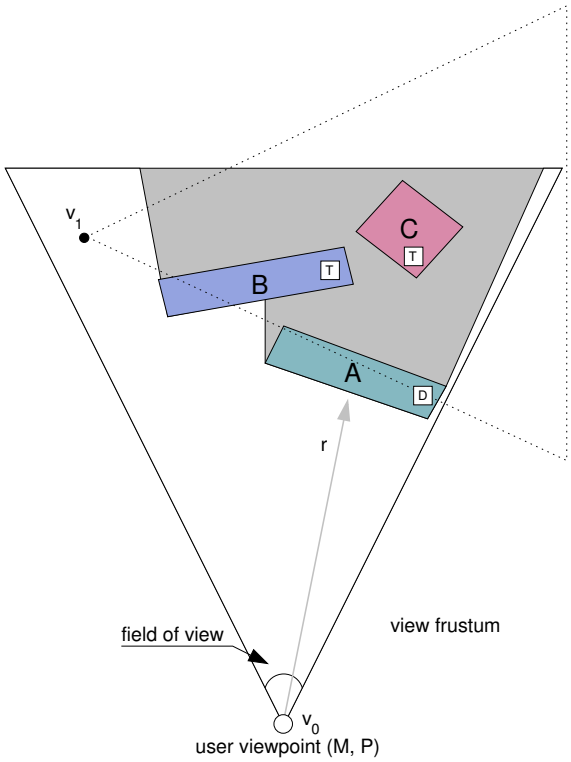

Fig. 1. Basic model for 3D occlusion. Target objects are flagged with "T" and distractors with "D".

3.2 Visual Tasks

The occlusion problem typically occurs in the following three *visual tasks*:

- *object discovery* – finding all targets $t \in O$ in the environment;
- *object access* – retrieving graphically encoded information associated with each target; and
- *spatial relation* – relating the spatial location and orientation of a target with its context.

Other visual tasks that are of relevance include object *creation*, *deletion* and *modification*; in this treatment, however, we consider these to be special cases of discovery and access with regards to inter-object occlusion, and consisting of the same subtasks as these three basic visual tasks.

3.3 Analysis

We can observe that all visual tasks are severely hampered by the existence of fully occluded objects. More specifically, for the purposes of object discovery, a fully occluded object will be impossible to discover without the use of some

occlusion management strategy, and identifying whether the object is a target never becomes an issue. Analogously for object access, the visual search will fail, and so will the perception of the object's visual properties. As a result, both tasks will affect the efficiency and correctness of users solving tasks using a visualization, but clearly, threats to object discovery are the most serious: if the user is unaware of the existence of an object, she will have no motivation to look for it and access never becomes an issue.

Partial occlusion, on the other hand, has a different effect on these tasks. For object discovery, users may have difficulties distinguishing object identity if too large a portion of the object is occluded. In this situation, the user may either miss the object entirely, count the same object multiple times, or believe different objects are part of the same object. Object access, on the other hand, will succeed in the visual search, although the perception of the object may still fail due to important parts of it being occluded.

Spatial relation, necessary for many complex interactions and visualizations, requires overview of the whole world, and is thus severely affected by both partially and fully occluded objects.

3.4 Environment Properties

The geometrical properties of the visualization environment are of special interest in this framework because they allow us to characterize the visualization and determine the nature of the occlusion problems that may arise. These properties can also be used to decide which occlusion management strategies are applicable for a specific situation.

In this treatment, we identify three main geometrical properties of the environment that interact to cause inter-object occlusion and influence the three basic visual tasks associated with the environment:

- *object interaction* – spatial interaction of objects in the environment;
- *object density* – amount of objects in the environment with regard to its size; and
- *object complexity* – detail level of individual objects in the environment.

Obviously, these are high-level properties that only generally describe an environment without going into detail on its actual content. Nevertheless, in the following sections we shall see how these property dimensions can serve as powerful reasoning tools for describing a 3D environment and selecting a suitable solution strategy for it.

3.4.1 Object Interaction

The object interaction property dimension describes how the individual objects in the environment interact spatially with each other, i.e. whether they touch, intersect or merely reside close to each other. There are five ordinal levels to this parameter (see Figure 2 for a visual overview):

- *none* – no spatial interaction between objects (realistically only applicable for singleton objects);
- *proximity* – objects are placed in such close proximity (without intersecting) that they occlude each other from some viewpoint;
- *intersection* – objects intersect in 3D space (without one fully containing another) such that they occlude each other;
- *enclosement* – one or several objects combine to fully enclose objects (without containing them) such that they are occluded from any viewpoint external to the enclosing objects; and
- *containment* – objects are fully contained in other objects such that they are occluded from any viewpoint.

Examples of these interaction levels exist in all kinds of 3D visualizations: proximity for nodes in 3D node-link diagrams, intersection for visualization of constructive solid geometry (CSG), enclosement for 3D objects placed inside larger objects (i.e. the walls of a virtual house), containment for 3D medical CAT scan data, etc.

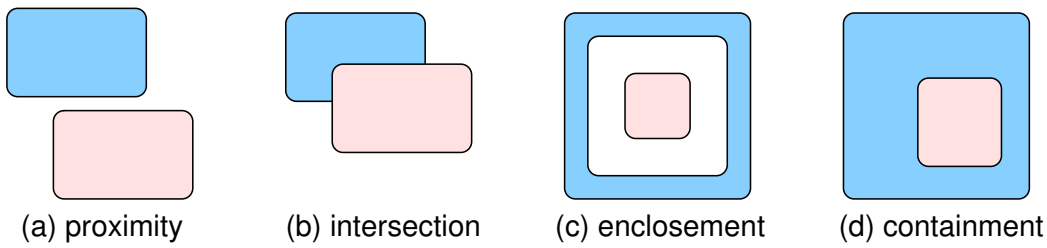

Fig. 2. Object interactions that may cause occlusion in 3D environments.

3.4.2 Object Density

The object density is a measure of the number of objects inhabiting the 3D environment; it follows naturally that the more objects per volume unit we are dealing with, the greater the chance and impact of occlusion will be. For environments containing a singleton object, naturally only self-occlusion can occur.

3.4.3 Object Complexity

The third geometrical property with an impact on the occlusion characteristics of an environment is the complexity of the objects in the environment. For complexity, we refer to the detail level of the 3D objects, i.e. typically the number of triangles (or other 3D primitives, such as quads, lines, and points) that make up the object, but we also include attributes such as color, material, and texture in this parameter. It follows that the more complex an object is, the more information it can potentially encode, and the larger the impact occlusion has on identification and perception of the object.

4 View-Projection Animation

The idea behind our technique for view-projection animation is to combine the major virtues of parallel and perspective projections: that (i) perspective projections offer a realistic view of a 3D environment akin to our perception of the real world, and that (ii) parallel projections offer a more exact view of the environment (in the sense that direct measurements can be made on the screen space). See Figure 5 for a side-by-side comparison. Furthermore, the nature of parallel projection means that inter-object occlusion is reduced in comparison to perspective projection since objects are assigned screen space according to their geometrical size only, regardless of their distance to the camera. Using perspective projection, a tiny object can fill the whole viewport if located sufficiently close to the viewpoint.

By combining these two projection modes into the same interaction technique, we are potentially able to enjoy the best of both worlds: the view defaults to perspective projection when the user is navigating the space normally, but allows for easy switching (glancing) to parallel projection when the user needs to perform object discovery or access. Furthermore, the transition between perspective and parallel projections, and vice versa, is smoothly animated to allow the user to maintain context of the environment and the projection at all times with a minimum of cognitive overhead. The transition also provides additional information on the structure of the 3D scene.

In addition to transitions back and forth between perspective and parallel projections, we augment our technique with functionality to change the center of projection as well as to modify the field-of-view angle in the perspective projection mode. Changing the center of projection gives an additional means for the user to arbitrate between occluding objects, and by gaining control of the field of view, the user can smoothly zoom in and out of the 3D environment at will. See Figure 3 for more details.

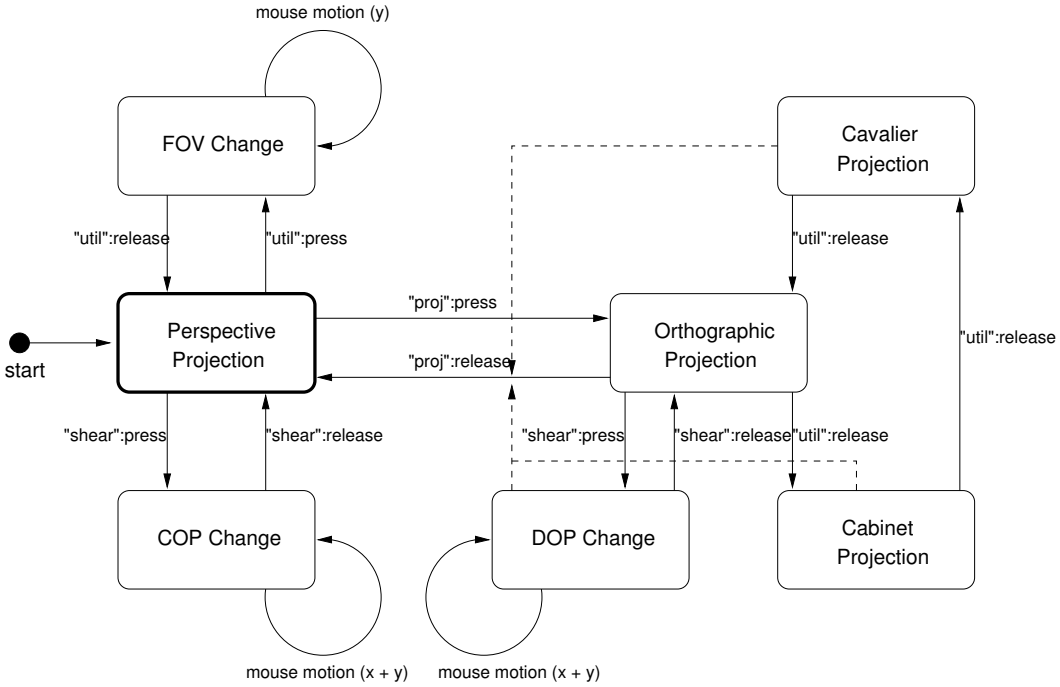

Fig. 3. State diagram for the projection animation interaction technique.

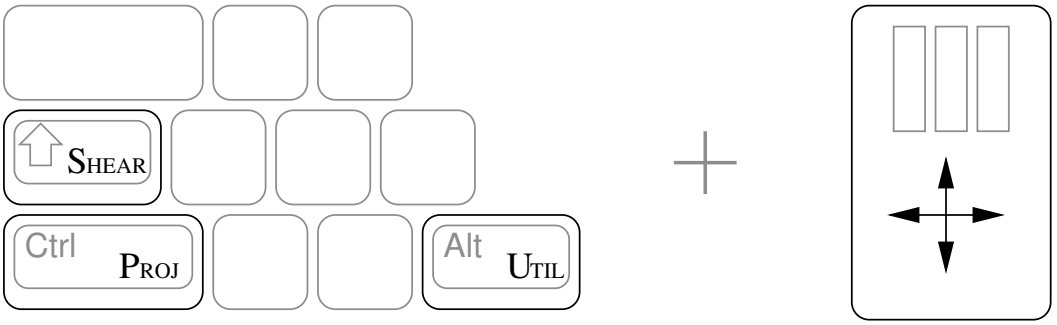

Fig. 4. Example interface mapping for the interaction technique (this setup is used in our prototype implementation).

For our interaction technique we define three input buttons, labeled PROJ, UTIL, and SHEAR, respectively; these can be mouse or keyboard buttons (see Figure 4 for an example input mapping). In addition, the technique also captures mouse motion for some parameter changes, notably the field-of-view and center-of-projection (direction-of-projection for parallel mode) modification states. The parallel projection mode has a number of pre-defined oblique projection modes that the user can cycle between: orthographic (head-on parallel projection) versus cavalier and cabinet projections, where the direction of projection is set at fixed values. Note that for all parallel projection modes, the release of the PROJ input button will smoothly revert the projection back to the default perspective mode. Reverting to the default state will also reset all view parameters, such as centering the center (or direction) of projection and setting the focal length to the default value.

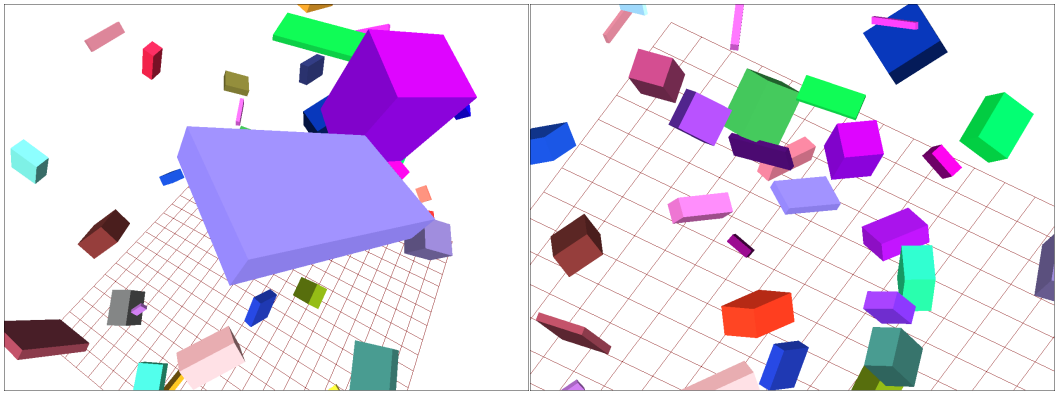

Fig. 5. Two images of the same environment from the same viewpoint using perspective (left) and parallel (right) projection.

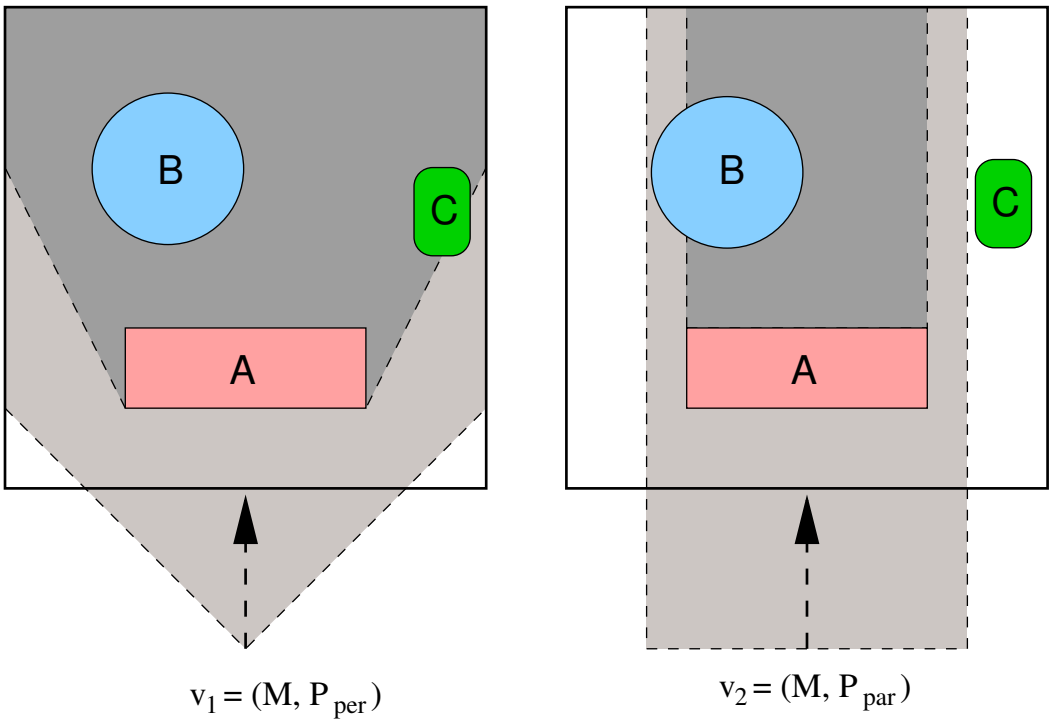

Fig. 6. Comparison of perspective (left) and parallel (right) projection modes in terms of occlusion. Object B is occluded by object A for perspective viewing but is visible in parallel projection mode. On the other hand, object C is visible in perspective mode, yet falls outside the viewing field in parallel mode.

4.1 Analysis

In the terminology of Section 3, view-projection animation dynamically modifies the view projection matrix $V = P(t)$ as a function of a time parameter t and a matrix $P(t)$. This may cause some objects that were previously occluded to become visible. No other properties are modified, implying that the technique is non-invasive. Figure 6 gives an example of the visibility of three

objects for both perspective and parallel modes.

We can categorize the view-projection animation technique as part of a more general solution strategy based on dynamic manipulation of the view projection to favorably present objects and minimize occlusion in the environment. In terms of the taxonomy of occlusion management techniques [8], this technique is a *projection distorter* with the following properties:

Primary purpose: discovery
Disambiguation strength: proximity
Depth cues: somewhat low
View paradigm: twin integrated views
Interaction: active
Target invariances: appearance (location and geometry distorted)

More specifically, the view-projection animation technique is primarily designed as a transient glance intended for discovering the presence of occluded targets, whereas access and relation would be performed after manipulating the view accordingly. Because occlusion management is performed in the view space, enclosement and containment cannot be handled, only occlusion caused by object proximity. Furthermore, due to parallel views discarding the depth coordinate of 3D objects, the technique does not retain very strong depth cues. The interaction is active and in the control of the user and the nature of modifications to the projection matrix means that only the appearance of targets is invariant, not location or geometry.

View-projection animation clearly improves object discovery by providing the user with means to avoid nearby objects hiding more distant ones. Toggling between the projection modes yields two different perspectives on the environment as well as intervening views during the smooth animation between them, strongly facilitating unguided visual search and, to some extent, identification, by disambiguating between occluding objects. Object access benefits much less from the technique; previous knowledge of the target's location is of little use when the view space is non-linearly transformed by the technique; on the other hand, the animation often allows users to track objects during the projection transition, potentially aiding access as well.

The applicability of the technique is limited with respect to intersecting objects: since we do not transform the space itself, enclosed and contained objects will remain occluded even after the projection transformation. As will be seen in the user study at the end of this paper, the technique performs well at low to medium-sized object density.

4.2 Projection Transitions

Transitions between various projection states are performed through simple linear interpolation between the source and destination projection transforms. In the case of the parallel-to-perspective transition (and its inverse), however, a linear interpolation will yield unsatisfactory results due to the non-linear relation between these two projections. For this case, we need to explore the matrix M that relates the two projection matrices P_{par} and P_{per}.

As discussed above, a parallel view transform represents the situation where the focal length of the camera is infinite. The transition from perspective to parallel view can be approximated in a real-life camera by a so-called "dolly and zoom" operation, where the camera is moved backwards at the same time as the focal length is increased (i.e. zoomed in). By keeping these parameters balanced, the focused object in the camera view will maintain the same size and shape, but the rest of the scene will appear to be "flattened". We simulate this effect in our transition between perspective and parallel projection.

Note that the focal point for the animation is placed in the center of the bounding box containing the 3D objects. Objects in the plane centered on this point and parallel to the viewing plane will thus remain the same geometrical screen size during the animation.

4.3 Implementation

We have implemented our interaction technique in a C++ application called PMorph. This application consists of a $100 \times 100 \times 100$ unit-sized cube populated with n boxes with randomized geometrical and graphical properties. The 3D rendering is performed using OpenGL. The application provides mouse-driven view controls with a camera that can be orbited and zoomed in and out around a focus point in the center of the environment (see Figure 4 for the controls for the prototype). The implementation of the interaction technique itself hooks seamlessly into the input handlers of the windowing system and requires no additional modification to the implementation of the 3D environment or the 3D objects.

4.4 Case Study: Blender Implementation

In order to study the feasibility and flexibility of our projection animation technique, we also implemented it inside the Blender [3] 3D modeling package. Blender is a very powerful and widely used 3D software suite that is

freely available as Open Source under the GPL license. Our implementation integrates seamlessly into Blender and allows modelers to animate between parallel and perspective projections in different 3D windows. The software naturally already supported these projection modes prior to our modifications, so we changed the projection code to perform a smooth animation between the two matrices. In addition to this, we introduced the capability for users to change the center of projection while in orthographic mode, providing an additional way to reduce occlusion.

Figure 7 shows a screenshot of the modified Blender software. There are no actual user interface changes to the application except a text in the currently selected viewport that indicates whenever the view is being animated or when the user is changing the center of projection. In general, a projection-control widget such as the IBar [24] would be a useful addition to any 3D modeling software like Blender, and state information about the view-projection animation technique could then easily be implemented into it.

While a 3D modeler is not the primary target platform for our technique (even though Grossman et al. [14,15] use the effect for this very purpose), this case study shows that the technique can indeed be implemented seamlessly inside existing 3D applications with only small modifications to the old code.

5 User Study

We have conducted a formal user study with two main motivations: (i) to empirically investigate the impact of occlusion on object discovery efficiency in 3D environments, and (ii) to study the performance of users given access to our view-projection animation technique in comparison to users with a normal perspective view.

5.1 Subjects

We recruited 26 subjects, six of which were female, from the undergraduate engineering programs at our university. No previous experience of 3D applications was required. Ages ranged from 20 to 40 years of age, and all subjects had normal or corrected-to-normal vision.

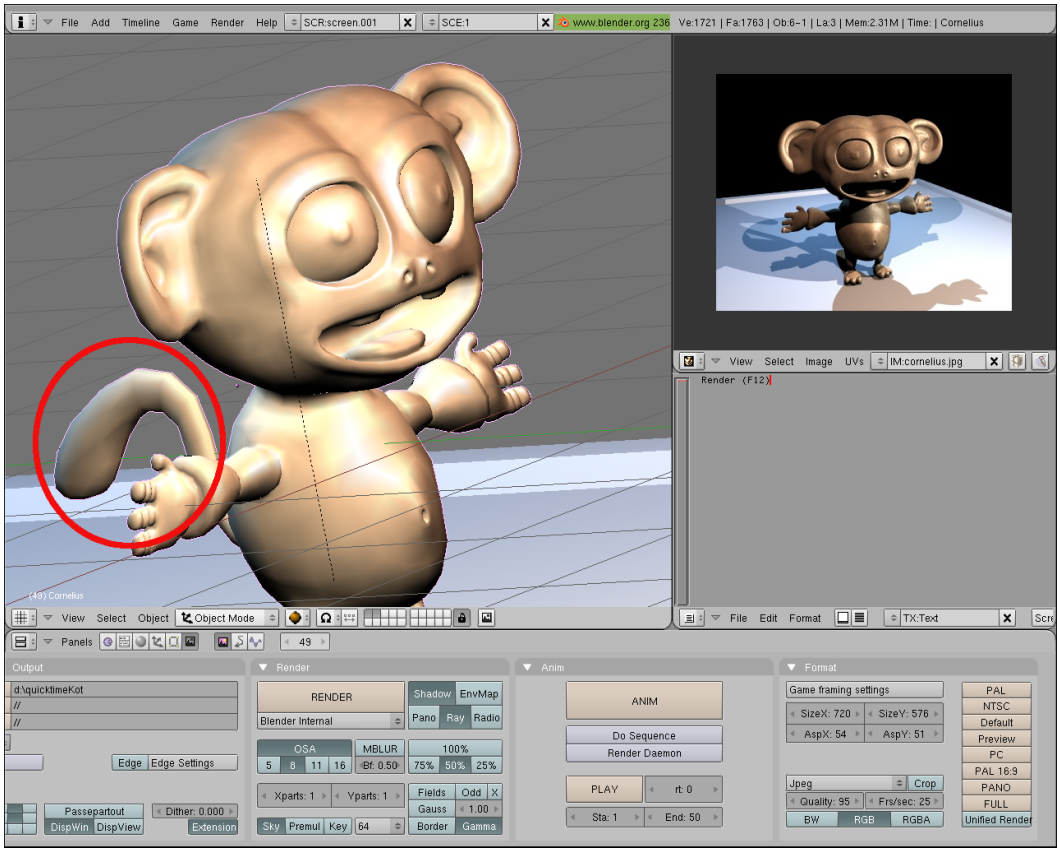

Fig. 7. Blender implementation screenshot. Note the parallel view (left) showing geometric features not visible in the perspective view (right).

5.2 Equipment

The experiment was conducted on a Pentium III 1 GHz desktop PC with 512 MB of memory and running the Linux operating system. All tasks were carried out using our prototype implementation. The display was a 19" monitor with the main visualization window fixed at 640×480 size.

5.3 Task

The view-projection animation technique is primarily aimed at 3D information visualization applications, such as 3D scatterplots and similar. Thus, we designed the task to model this kind of visualization. Subjects were asked to perform object discovery in a simple $100 \times 100 \times 100$ environment filled with 3D boxes by counting the number of boxes of a given color. Target colors were restricted to one of the primary RGB colors (i.e. red, green, or blue), and all distracting objects were configured to contain no elements of that color component. Each task instance was fully randomized, including the position,

orientation, and size of the distractors. At least 1% and at most 10% of the total number of objects were targets. Box dimensions (both targets and distractors) ranged from 1% to 12.5% of the environment dimensions. Intersection but no enclosement or containment was allowed. A simple 20×20 line grid was rendered at the bottom of the environment to facilitate user orientation.

The camera focus point was fixed at the center of the environment and the orientation was randomized within 60° from the horizontal. In addition, the camera position was also randomized and offset sufficiently from the focus point so that all objects in the scene were visible. Field-of-view angle for the perspective view was fixed at 60°. For the dynamic camera, the users could freely orbit (rotate) and dolly (move closer or further away) the camera around the focus point.

5.4 Design

The experiment was designed as a repeated-measures factorial analysis of variance (ANOVA), with the independent variables Density (two levels, "low" or "high"), Camera ("static" or "dynamic", i.e. a fixed or a user-controlled camera), and PMorph ("on" or "off", i.e. whether the projection animation technique was available or not), all of them within-subjects. The dependent variables were the number of found target objects and the completion time for each task. Subjects received the PMorph and Camera conditions in randomized order to avoid systematic effects of practice; for the Density condition, the ordering was low to high.

Users performed the test in sets of 10 tasks for each condition. Each task scenario was completely randomized, with either 50 or 200 total objects in the environment depending on the density, and up to 10% of them being targets. See Table 1 for the complete experimental design.

For each specific condition, subjects were instructed in which features (dynamic or static camera, projection animation on or off) were available to them. Tasks were given automatically by a testing framework implemented in the software and answers were input by the user directly back into the framework, thus requiring no intervention by the test administrator. The software silently recorded the completion time, total target number, and found target number for each task. Trial timing started as soon as each new task was presented, and ended upon the subject giving an answer.

Each session lasted approximately thirty to forty minutes. Subjects were given a training phase of up to five minutes to familiarize themselves with the controls of the application.

With 26 participants and 10 search tasks for each of the 8 conditions, there were 2080 trials recorded in total. After having completed the full test, subjects were asked to respond to a post-test questionnaire (see Table 2) where they were asked to select their combination of projection mode (view-projection or normal perspective) and camera mode (static or dynamic) of preference for a number of different aspects.

Density	**Camera**	**PMorph**	**Objects**	**Targets**
low	static	off	50	1-5
low	static	on	50	1-5
low	dynamic	off	50	1-5
low	dynamic	on	50	1-5
high	static	off	200	1-20
high	static	on	200	1-20
high	dynamic	off	200	1-20
high	dynamic	on	200	1-20

Table 1
Experimental design. All three factors were within-subjects. The order of the "PMorph" and "camera" conditions were randomized to counterbalance learning effects.

Task	**Description**
Q1	Which modes did you prefer with respect to ease of use?
Q2	Which modes did you prefer with respect to efficiency of solving the tasks?
Q3	Which modes did you prefer with respect to enjoyment?
Q4	Which modes helped you feel the most confident about having discovered all objects in the scene?
Q5	Which modes did you feel were the fastest to use?
Q6	Overall, which modes would you choose for performing this task in your daily work?

Table 2
Post-test questionnaire. For each question, participants were asked to rank both view-projection animation versus normal perspective mode, as well as static versus dynamic camera.

6 Results

We divide the results from the user study into completion times, correctness, and subjective ranking categories. Note that for the correctness measure, we derive the cumulative errors for each task set from the sum of the differences

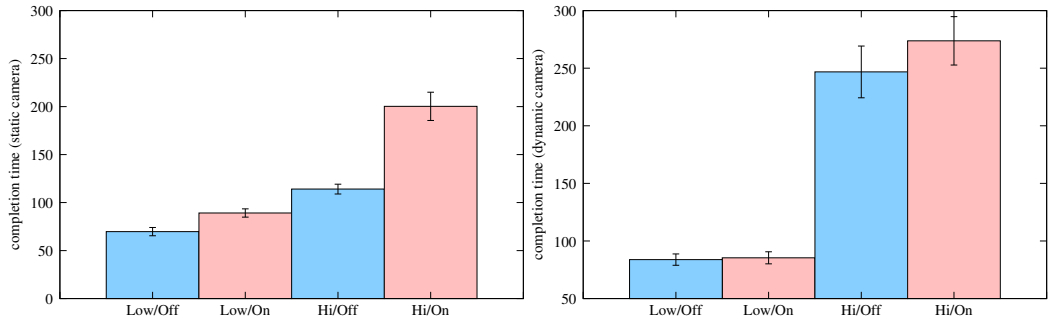

Fig. 8. Mean completion times for solving a full task set for static (left) and dynamic (right) camera types (standard deviations shown as error bars). Participants with standard perspective projection completed their tasks significantly faster than those using view-projection animation for both low and high density environments.

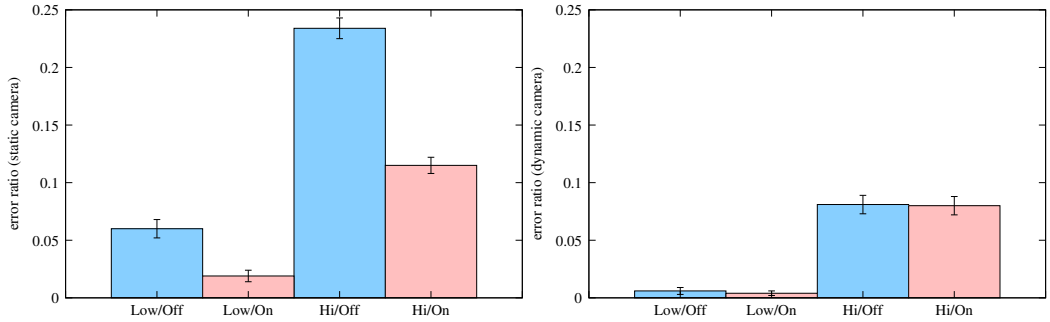

Fig. 9. Mean error ratios for solving a full task set for static (left) and dynamic (right) camera types (standard deviations shown as error bars). Participants using view-projection animation were significantly more accurate than those with standard perspective projection for both low and high density environments.

between the total number of targets and the found targets for each task. The error ratio is defined as the cumulative error divided by the sum of the total number of targets for the task set, i.e. the number of errors per target.

6.1 Time

Overall, the view-projection animation technique caused participants to take longer time to solve tasks when using a static camera, but did not significantly affect completion times for a dynamic camera. Not surprisingly, participants took significantly more time when using a dynamic camera than a static one, as well as more time for high than low object density.

Averaging results from the static and dynamic camera modes, the mean completion time for a full task set (10 tasks) using normal perspective projection was 128.093 (s.d. 7.803) seconds, as opposed to 162.311 (s.d. 9.697) seconds for projection animation. This is also a significant difference ($F(1, 25) = 38.752, p < 0.001$). Not surprisingly, the main effect for density was significant

($F(1,25) = 108.887, p < 0.001$), with mean completion times for low and high conditions of 81.384 (s.d. 4.025) and 209.20 (s.d. 14.086) seconds, respectively.

Figure 8 summarizes the results for the individual density conditions. For the low density condition, the mean completion time was 75.796 (s.d. 4.012) and 86.972 (s.d. 4.420) seconds for the normal projection and the projection animation technique. For the high density, the completion times were instead 180.414 (s.d. 12.680) and 236.995 (s.d. 16.053) seconds, respectively. Both of these differences are significant ($F(1,25) = 19.363, p < 0.001$ versus $F(1,25) = 30.797, p < 0.001$).

Furthermore, for the low density case using a static camera, the mean completion time was 69.737 (s.d. 4.298) versus 89.152 (s.d. 4.339) for normal projection versus projection animation; also a significant difference ($F(1,25) = 36.638, p < 0.001$). Similarly, for the low density case using a dynamic camera, the completion time was 83.817 (s.d. 4.933) versus 85.387 (s.d. 5.256) seconds; this is not a significant difference, however ($F(1,25) = 0.226, p = 0.639$). In the case of the high density condition with a static camera, the mean completion time was 114.049 (s.d. 5.114) for normal perspective projection as opposed to 200.245 (s.d. 14.683) for projection animation, a clearly significant difference ($F(1,25) = 53.824, p < 0.001$). Finally, for the high density case using a dynamic camera, the completion times were 246.778 (s.d. 22.461) versus 273.745 (s.d. 20.998), a nonsignificant difference ($F(1,25) = 2.504, p = 0.126$).

Analogously to other factors, the main effect on completion time for the camera mode was significant ($F(1,25) = 46.874, p < 0.001$); static camera completion time averaged at 117.267 (s.d. 6.266) seconds, whereas the dynamic camera completion time average was 173.137 (s.d. 11.569).

6.2 Correctness

The view-projection animation technique helped participants to be more correct for a static camera, but did not significantly improve correctness for a dynamic camera. In total, participants were more accurate for dynamic over static camera, as well as for low over high object density.

Averaging static and dynamic camera modes, the mean error ratio for a full task set (10 tasks) using normal perspective projection compared to projection animation was 0.095 (s.d. 0.003) versus 0.055 (s.d. 0.003), respectively. This is a statistically significant difference ($F(1,25) = 75.757, p < 0.001$). Again, density had a significant impact on correctness ($F(1,25) = 407.290, p < 0.001$); the average error ratio for the low density condition was 0.022 (s.d. 0.002), to contrast with 0.127 (s.d. 0.005) for the high density condition. This suggests that occlusion does negatively affect object discovery efficiency.

See Figure 9 for an overview of additional correctness results. For the low density, the mean error ratio was 0.033 (s.d. 0.004) for normal projection and 0.012 (s.d. 0.003) for projection animation, versus 0.157 (s.d. 0.006) and 0.097 (s.d. 0.006) for the high density case. Both of these pair-wise differences are significant ($F(1,25) = 13.493, p = 0.001$ and $F(1,25) = 55.176, p < 0.001$, respectively).

Furthermore, in the low density case using a static camera, the ratio was 0.60 (s.d. 0.008) versus 0.019 (s.d. 0.005) for normal projection versus projection animation, respectively; this was also a significant difference ($F(1,25) = 14.595, p = 0.001$). Analogously, for the low density case using a dynamic camera, the ratio was 0.006 (s.d. 0.003) versus 0.004 (s.d. 0.002); this, however, was not a significant difference ($F(1,25) = 0.240, p = 0.629$). On the other hand, in the high density case using a static camera, the average error ratio was 0.234 (s.d. 0.009) for normal perspective and 0.115 (s.d. 0.007) for projection animation. This is again a significant difference ($F(1,25) = 183.217, p < 0.001$). In comparison, for the high density case using a dynamic camera, the ratio was 0.081 (s.d. 0.008) versus 0.080 (s.d. 0.008), also not a significant difference ($F(1,25) = 0.006, p = 0.941$).

Finally, the camera factor had considerable bearing on correctness; error ratios for static camera averaged at 0.107 (s.d. 0.004), whereas dynamic camera error ratios averaged at 0.043 (s.d. 0.003), a significant difference ($F(1,25) = 199.284, p < 0.001$).

6.3 Subjective Rankings

The rankings given by the participants in the post-test questionnaire are overall positive and in favor of our projection animation technique; see Table 3 for an overview. The Q2 and Q6 questions, relating to perceived efficiency and preference, are of special interest, and are significantly in favor of our technique (65% and 73%, respectively). Subjects also consistently ranked a dynamic camera over a static camera.

7 Discussion

This paper presents two main contributions: (i) an analysis of the space of the occlusion problem in 3D environments, and (ii) the view-projection animation technique used to reduce inter-object occlusion for any 3D visualization. Figure 10 summarizes the results from the user study in a scatter plot showing both completion times and correctness. As shown by the figure, both of

Task	Factor	(a) Projection Mode			(b) Camera Mode		
		Normal	PMorph	Undec.	Static	Dynamic	Undec.
Q1	Ease-of-use	58%	35%	7%	15%	85%	0%
Q2	Efficiency	23%	65%	12%	12%	88%	0%
Q3	Enjoyment	19%	69%	12%	0%	100%	0%
Q4	Confidence	23%	69%	8%	15%	85%	0%
Q5	Speed	50%	46%	4%	39%	54%	7%
Q6	Overall	15%	73%	12%	0%	96%	4%

Table 3
Post-test ranking results of normal perspective projection versus view-projection animation as well as static over dynamic cameras.

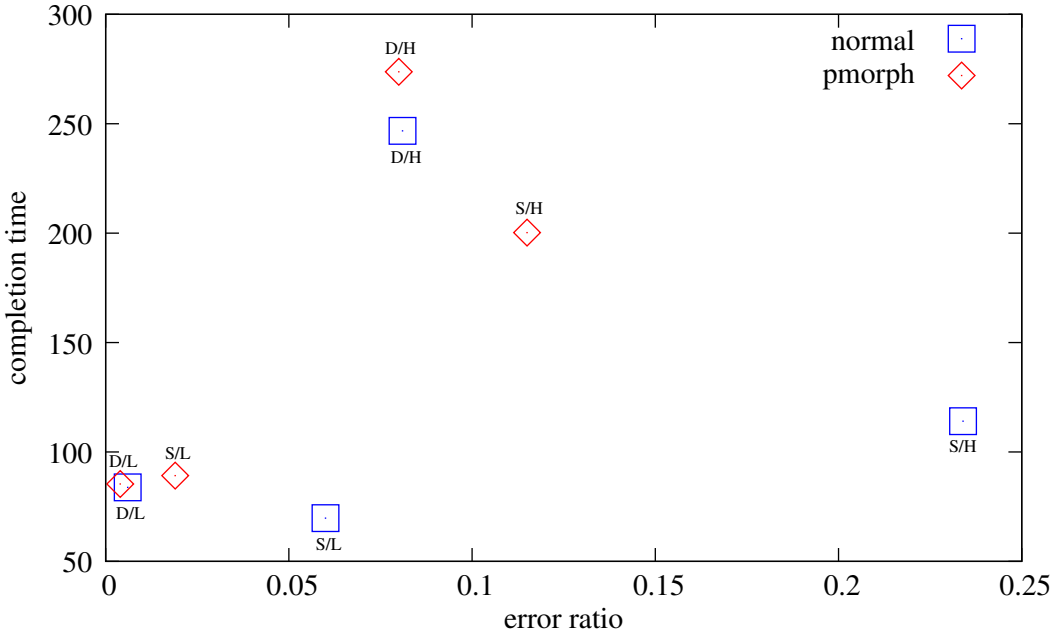

Fig. 10. Summary of correctness (horizontal) and completion time (vertical) results for both normal perspective (blue squares) and view-projection animation (red diamonds). (S = static camera; D = dynamic camera; L = low density; H = high density)

these contributions are validated by our results; we see that increasing object occlusion leads to significantly reduced discovery efficiency, and that the availability of projection animation significantly boosts efficiency in all object density conditions, respectively. In addition, by giving users control over the viewpoint, the impact of the occlusion problem is significantly diminished. On the other hand, this comes at the cost of longer completion times; the participants spent much more time solving tasks when having access to projection animation or a controllable camera, essentially trading speed for accuracy.

This last finding is typical of many advanced interaction techniques—giving users access to a new tool for solving a task more accurately often means that more time is spent taking advantage of the new information the tool provides. Completion time may increase, but so will accuracy (Fitts' law [11] is a classic example of this for pointing-like tasks, but similar models can be applied to more complex interaction techniques and tasks). How to balance this tradeoff depends on the context and the user task.

It is particularly interesting to study whether a user-controlled camera is sufficient to negate the occlusion problem, and whether the projection animation technique presented here is necessary. There is no clear benefit of projection animation over a traditional dynamic camera. However, we claim that projection animation is orthogonal to controllable cameras, and that they complement each other. Furthermore, our informal observations during the user study indicated that users with access only to a controllable camera performed significantly more view changes than when having access to both a controllable camera and projection animation. All 3D view changes incur a risk of loss of context and orientation, especially for high object densities, and so it is in our best interest to keep the amount of such changes low. The advantage of view-projection animation is that it can give some of the benefits of a movable camera, yet without actually having to change the viewpoint. Nevertheless, we suggest that a combination of the two conditions will work best for practical applications.

Parallel projection assigns screen space to objects proportional to their geometrical size regardless of the distance to the camera, but the drawback is that the viewing volume is a box instead of a pyramidal frustum as for perspective projection. This means that peripheral objects will be lost in parallel mode, essentially rendering these objects impossible to discover. By smoothly combining *both* parallel and perspective projection into a single interaction technique, we are able to sidestep this problem and get the best of both worlds from the two projections.

A potential drawback of the technique is that the use of parallel projection leads to a loss of some depth cues in a 2D image of the environment (more specifically relative and familiar size as well as relative height). However, the spring-loaded nature of the interaction allows users to switch easily back and forth between projection modes to disambiguate potential depth conflicts between objects. Also, for the information visualization applications we primarily consider for our technique, depth cues are less important than the relationships between the 3D objects.

As indicated by the analysis of the presented technique (see Section 4.1), projection animation has a rather low disambiguation strength and can realistically only handle proximity-based occlusion. Indoor or outdoor scenery, such

as for Virtual Reality 3D walkthroughs, will be less tractable to this approach. On the other hand, the technique is designed primarily for information visualization applications, which are more akin to the 3D scatterplot task employed in the user study.

Finally, another weakness of view-projection animation is that it only merges two separate projections of the same 3D scene from the same viewpoint. Needless to say, two projections are far from enough to manage occlusion in the general case, and approaches like worldlets [10] recognize this by providing for a large, unspecified number of simultaneous views of a 3D world. However, each new view has diminishing returns in terms of new visible objects, and also entails users having to allocate a part of their attention to yet another view. The solution used in the view-projection animation technique is a good two-view compromise optimized for higher speed yet retaining good accuracy.

8 Conclusions

We have presented an interaction technique for the seamless integration of perspective and parallel projection modes, allowing users to combine realism with accuracy, as well as reducing inter-object occlusion in 3D environment views. Results from a user study conducted on a prototype version of the technique show that occlusion in 3D environments has a major impact on efficiency, but that our technique allows for significant improvements in both object discovery and object access. Our technique treats 3D objects as immutable entities and requires no changes to the implementation or representation of the 3D environment, and should thus be possible to integrate with almost any 3D visualization.

Acknowledgments

Many thanks to the developers of the Blender project for their help on integrating the technique into the Blender3D modeller. Thanks to the anonymous reviewers for their many constructive comments.

References

[1] M. Agrawala, D. Zorin, T. Munzner, Artistic multiprojection rendering, in: Proceedings of the Eurographics Workshop on Rendering Techniques, 2000.

[2] M. Q. W. Baldonado, A. Woodruff, A. Kuchinsky, Guidelines for using multiple views in information visualization, in: Proceedings of the ACM Conference on Advanced Visual Interfaces, 2000.

[3] Blender, see *http://www.blender3d.org* (Aug. 2007).

[4] D. A. Bowman, C. North, J. Chen, N. F. Polys, P. S. Pyla, U. Yilmaz, Information-rich virtual environments: theory, tools, and research agenda, in: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, 2003.

[5] I. Carlbom, J. Paciorek, Planar geometric projections and viewing transformations, ACM Computing Surveys 10 (4) (1978) 465–502.

[6] A. Cockburn, Revisiting 2D vs 3D implications on spatial memory, in: Proceedings of the Australasian User Interface Conference, 2004.

[7] A. Cockburn, B. McKenzie, 3D or not 3D?: Evaluating the effect of the third dimension in a document management system, in: Proceedings of the ACM CHI 2001 Conference on Human Factors in Computing Systems, 2001.

[8] N. Elmqvist, P. Tsigas, A taxonomy of 3D occlusion management techniques, in: Proceedings of the IEEE Conference on Virtual Reality, 2007.

[9] N. Elmqvist, M. E. Tudoreanu, Evaluating the effectiveness of occlusion reduction techniques for 3D virtual environments, in: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, 2006.

[10] T. T. Elvins, D. R. Nadeau, D. Kirsh, Worldlets – 3D thumbnails for wayfinding in virtual environments, in: Proceedings of the ACM Symposium on User Interface Software and Technology, 1997.

[11] P. M. Fitts, The information capacity of the human motor system in controlling the amplitude of movement, Journal of Experimental Psychology 47 (1954) 381–391.

[12] S. Fukatsu, Y. Kitamura, T. Masaki, F. Kishino, Intuitive control of "bird's eye" overview images for navigation in an enormous virtual environment, in: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, 1998.

[13] M. Gleicher, A. Witkin, Through-the-lens camera control, in: Computer Graphics (SIGGRAPH '92 Proceedings), 1992.

[14] T. Grossman, R. Balakrishnan, G. Kurtenbach, G. W. Fitzmaurice, A. Khan, W. Buxton, Interaction techniques for 3D modeling on large displays, in: Proceedings of the ACM Symposium on Interactive 3D Graphics, 2001.

[15] T. Grossman, R. Balakrishnan, G. Kurtenbach, G. W. Fitzmaurice, A. Khan, W. Buxton, Creating principal 3D curves with digital tape drawing, in: Proceedings of the ACM CHI 2002 Conference on Human Factors in Computing Systems, 2002.

[16] C. Mei, V. Popescu, E. Sacks, The occlusion camera, Computer Graphics Forum 24 (3) (2005) 335–342.

[17] O. Merhi, E. Faugloire, M. Flanagan, T. A. Stoffregen, Motion sickness, console video games, and head mounted displays, Human Factors(in press).

[18] J. C. Michener, I. B. Carlbom, Natural and efficient viewing parameters, in: Computer Graphics (SIGGRAPH '80 Proceedings), 1980.

[19] J. S. Pierce, M. Conway, M. van Dantzich, G. Robertson, Tool spaces and glances: Storing, accessing, and retrieving objects in 3D desktop applications, in: Proceedings of the ACM Symposium on Interactive 3D Graphics, 1999.

[20] V. Popescu, D. G. Aliaga, The depth discontinuity occlusion camera, in: Proceedings of the ACM Symposium on Interactive 3D Graphics, 2006.

[21] V. Popescu, J. Dauble, C. Mei, E. Sacks, An efficient error-bounded general camera model, in: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission, 2006.

[22] K. Singh, A fresh perspective, in: Proceedings of Graphics Interface, 2002.

[23] K. Singh, R. Balakrishnan, Visualizing 3D scenes using non-linear projections and data mining of previous camera movements, in: Proceedings of AFRIGRAPH, 2004.

[24] K. Singh, C. Grimm, N. Sudarsanam, The IBar: a perspective-based camera widget, in: Proceedings of the ACM Symposium on User Interface Software and Technology, 2004.

[25] R. Stoakley, M. J. Conway, R. Pausch, Virtual Reality on a WIM: Interactive worlds in miniature, in: Proceedings of the ACM CHI'95 Conference on Human Factors in Computing Systems, 1995.

[26] T. A. Stoffregen, L. J. Hettinger, M. W. Haas, M. M. Roe, L. J. Smart, Postural instability and motion sickness in a fixed-base flight simulator., Human Factors 42 (2000) 458–469.