Multi-scale Manipulation in the World in Miniature Metaphor

paper 1046

Abstract

The World in Miniature Metaphor (WIM) allows users to select, manipulate and navigate efficiently in virtual environments. In addition to the first-person perspective offered by typical VR applications, the WIM offers a second dynamic viewpoint through a hand-held miniature copy of the environment. In this paper we explore different strategies to allow the user to interact with the miniature replica at multiple levels of scale. Unlike competing approaches, we support complex indoor environments by explicitly handling occlusion. We discuss algorithms for selecting the part of the scene to be included in the replica, and for providing a clear view of the region of interest. Key elements of our approach include an algorithm to recompute the active region from a subdivision of the scene into cells, and a view-dependent algorithm to cull-away occluding geometry through a small set of slicing planes roughly oriented along the main occluding surfaces. We present the results of a user-study showing that our technique clearly outperforms competing approaches on spatial tasks performed in densely-occluded scenes.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques

1. Introduction

Virtual Reality applications in areas such as urban management, factory planning, and shipbuilding design, have to deal with increasingly complex scenes with a large number of occluding objects. Efficient 3D interaction in such densely-occluded scenes is still a challenging problem. Occlusion is a big handicap for accomplishing spatial tasks, as most interaction techniques for 3D selection and manipulation require the involved objects to be visible from the user's viewpoint. A common solution for selecting occluded objects is to navigate to a different location so that the targets become unoccluded. However, this navigate-to-select approach is impractical for manipulation-intensive applications.

During the last few years several methods have been proposed to handle 3D occlusion on VEs [ET08]. However, most approaches proposed so far are either limited to scenes with low depth complexity, or have been designed with discovery tasks on mind, being not suitable for performing spatial tasks such as selection and manipulation.

The interaction metaphor this work is based on is the World in Miniature. The World in Miniature (WIM) metaphor [SCP95] complements the first-person perspective offered by typical VR applications with a second dynamic view of a miniature copy of the virtual world. This second exocentric view of the world helps users to understand the spatial relationships of the objects and themselves inside the virtual world. Furthermore, because the WIM is hand-held, it can be quickly explored from different viewpoints without destroying the larger, immersive point of view (Figure 1).

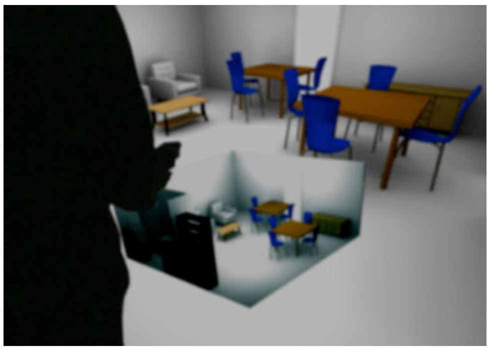


Figure 1: *User interacting with the WIM. The non-dominant hand is used to establish the frame of reference by rotating the WIM, whereas selection and manipulation tasks are performed with the dominant hand.*

The WIM has been used as a unifying metaphor to accomplish many user tasks including object selection and manipulation, navigation and path planning. Object selection can be accomplished either by pointing directly at the object or by pointing at its proxy on the WIM. Once selected, objects can be manipulated either at the scale offered by the WIM or at the one-to-one scale offered by the immersive environment. The WIM can also include an avatar of the user that can be moved to change the user location in the environment, providing camera control from an exocentric point of view.

The WIM partially solves the problem of having to rely to navigation for selecting occluded objects. By rotating the hand-held replica, users can view and pick objects that are obscured from the immersive camera viewpoint.

The classic WIM implementation exhibits two limitations affecting its scalability. One the one hand, the user can only choose among two levels of scale (the scale offered by the WIM and the one-to-one scale offered by the immersive view). On the other hand, basic techniques for occlusion handling such as backface culling are only effective for simple scenes.

As a consequence, two key problems have to be addressed for the WIM metaphor to be applicable to complex, large-scale scenes. First, one must decide which region of the environment should be included in the miniature copy. Early implementations put a replica of the whole environment in the miniature, thus limiting its application to simple models like a single room. A few extensions of the WIM have been proposed to handle more complex models [CGV05, LFKZ01, WHB06]. These approaches use a miniature copy which includes only those objects inside a region of interest, which can be scaled and moved either manually or automatically. These techniques allow the accomplishment of user tasks at different levels of scale. However, current approaches either provide adhoc solutions valid for a limited class of models, or assume the input scene already provides some information about its logical structure, such as the subdivision of a building into floors.

The second problem to be addressed is occlusion management. Regardless of the size and shape of the region included in the miniature copy, bounding geometry such as walls must be conveniently culled away to make interior objects visible. Early implementations relied solely on backface culling techniques [SCP95], but these are suitable only for very simple models; general 3D models require the application of more sophisticated techniques for handling 3D occlusion such as cut-away views [DWE03].

In this paper we aim at facilitating 3D interaction tasks in large-scale, densely-occluded scenes, focusing on applications running on spatially-immersive displays such as CAVEs. We present several strategies to handle complexity and occlusion in the WIM so that it can be used in arbitrarily-complex scenes. The main contributions of the paper are:

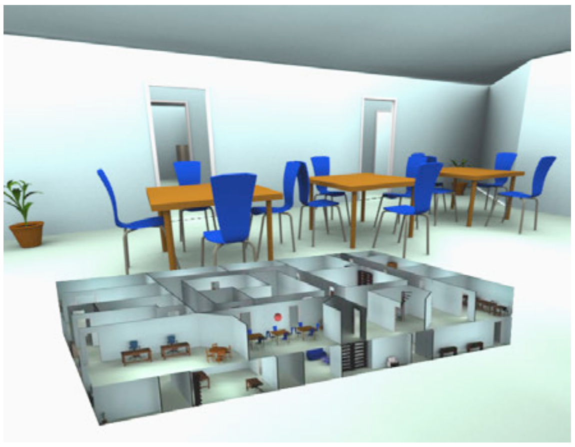


Figure 2: *Example of our improved WIM rendering. The miniature replica provides a cut-away view of a part of the model according to the viewing direction and the user's hand position inside the WIM (shown as a red sphere).*

- An algorithm to automatically recompute the position and size of the axis-aligned box representing the part of the scene to be included in the miniature replica. This algorithm is used at start-up and whenever the WIM has to be updated due to the user navigating to another location. A key ingredient of the algorithm is a subdivision of the scene into *cells*, which are detected during preprocessing. A cell is a region of the scene with no significant occluding geometry on its *interior* (occluding surfaces are only allowed on the cell boundary). This property makes cells particularly suitable for automatically delimitating the extents of the WIM. Similar to [WHB06], the user is allowed to scale up and down the axis-aligned box to perform the task in hand at a suitable level of scale.
- A view-dependent algorithm to cull-away occluding geometry so that the objects the user is willing to manipulate appear unoccluded. The main novelty of our approach is the use of the position of the user's hand inside the WIM to determine a small set of slicing planes. The slicing planes are chosen from a set of precomputed slicing planes roughly oriented along the main occluding surfaces. The rationale here is that the hand position inside the WIM is a good indication of the part of the WIM the user is interested at, and thus it is a natural way to determine which objects must appear unoccluded. By using slicing planes roughly oriented with main occluding surfaces, we reveal interior objects while preserving as much as possible context information that can be useful to perform the task.
- A user study evaluating our approach. We compare our revealing strategy with competing approaches for occlusion management such as transparency and isomorphically-controlled hand-held clipping planes.

2. Previous work

In this section we briefly review previous work related to the problem being addressed (enhancements to the WIM metaphor) and to our adopted solution (cell decomposition and occlusion management).

2.1. Worlds in Miniature

The WIM was proposed originally by Stoakley, Conway and Pausch [SCP95] who discussed their application to selection, manipulation and traveling tasks. They found that users easily understood the mapping between virtual world objects and the proxy WIM objects. Unfortunately, using a replica of the whole environment in the miniature limits its application to simple models like a single room, due to occlusion and level-of-scale problems. Several WIM extensions have been proposed to overcome this limitations. The STEP WIM proposed by La Viola et al. [LFKZ01], puts the miniature replica on the floor screen so that it can be interacted with the feet. The main advantage is the freeing of the hands for other tasks. The method provides several methods for panning and scaling the part of the scene covered by the miniature. The Scaled and Scrolling WIM (SSWIM) [WHB06] supports interaction at multiple levels of scale trough scaling and scrolling functions. SSWIM adds functionally and hence complexity because the user has to scale the model manually. The scrolling is automatic though when the user moves to a position outside of a dead zone. This dead zone is a box centered at the SSWIM. Chittaro et al. [CGV05] propose an extension of the WIM to support user navigation through virtual buildings, allowing users to explore any floor of a building without having to navigate. Trueba et al. [RT09] handle occlusion through a Layered Depth Image built every frame from a set of planes approximating the room surrounding the user. Their approach provide clear views for single rooms, but only one level-of-scale is provided as the WIM is always restricted to match the current room. This fact would force the user to navigate to perform manipulation tasks accross several rooms, such as those analyzed in our user studies.

2.2. Cell and portal decompositions

A cell-and-portal graph (CPG) is a structure that encodes the visibility of the scene, where nodes are cells, usually rooms, and edges are portals which represent the openings (doors or windows) that connect the cells. There are few papers that refer to the automatic determination of CPGs, and most of them work under important restrictions [CCSD03]. Very few works provide a solution for general, arbitrarily-oriented models with complex, non-planar walls. Notable exceptions are those approaches based on a distance-field representation of the scene [HDS03, AVF04]. Our approach for cell detection also relies on a distance field and it is based on the scene decomposition presented in [AVF04].

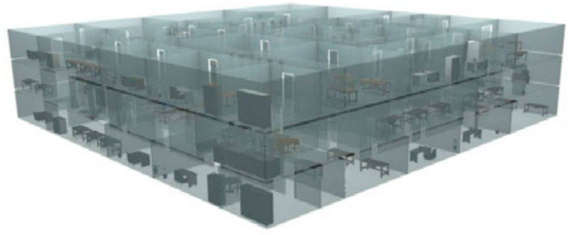
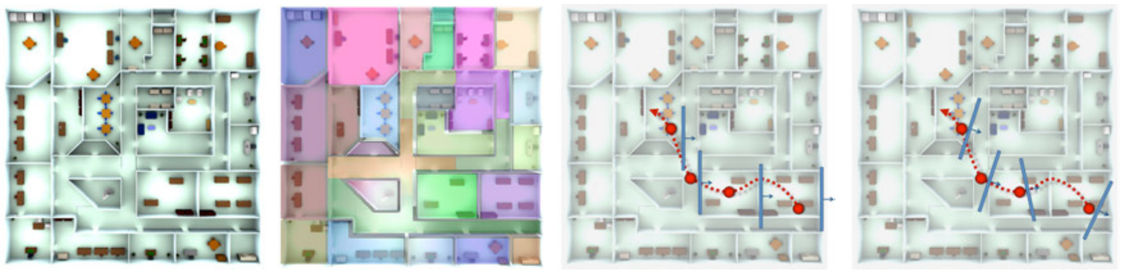


Figure 3: *Test building rendered with alpha blending. Despite furniture pieces are evenly distributed throughout the building, only those closest to the viewpoint can be distinguished.*

2.3. Management of 3D occlusion

Complex geometric models composed of many distinct parts arise in many different applications such as architecture, engineering and industrial manufacturing. Three-dimensional occlusion management techniques are often essential for helping viewers understand the spatial relationships between the constituent parts that make up these datasets. Elmqvist and Tsigas [ET08] analyze a broad range of techniques for occlusion management and identify five main design patterns: Multiple Viewports (using two or more separate views of the scene), Virtual X-Ray (turn occluded objects visible), Tour Planners (a precomputed camera animation reveals the otherwise occluded geometry), Interactive Exploders (adopted for the WIM in [CGV05] through a floor sliding mechanism) and Projection Distorter (nonlinear projections integrate two or more views into a single view).

Virtual X-Ray techniques are particularly relevant to our approach as they facilitate discovery tasks and provide access to potential targets by selectively removing distractors. Distractors can be removed by turning them semi-transparent or invisible [EAT07]. Using transparency is particularly simple to implement as it eliminates the need for identifying distracting objects, but rendering semi-transparent objects increases the visual complexity of the scene and the cognitive load to the user. Distractor removal techniques can be view-independent [CH06] or view-dependent [CGV05, BF08]. Here we only discuss view-dependent techniques as they are more appropriate to reveal geometry on a hand-held miniature. A number of algorithms for generating cut-away views are proposed by Diepstraten et al. [DWE03]. The proposed algorithms are both efficient and easy to implement, although two important assumptions are made: they assume that the classification of objects as interior or exterior is provided by an outside mechanism, and that the cut-out geometry is convex. Fortunately, our decomposition of the scene into cells allows the application of very simple algorithms to generate cut-away views.



(a) (b) (c) (d)

Figure 4: *Overview of our approach: (a) test model (only one floor is shown for clarity); (b) cell decomposition; (c) sample path described by the user's hand during manipulation (in red), and slicing planes (in blue) corresponding to four points along the path; (d) planes that would be defined by the orientation of the hand-held input device (only four samples are shown).*

3. Our approach

3.1. Overview

We aim at improving the WIM metaphor by addressing the following problems:

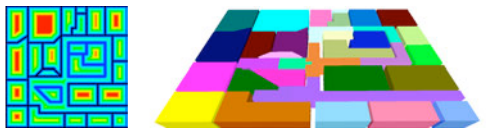
- **WIM delimitation:** given the current user location, compute the part of the scene to be included in the WIM. This means computing the center and size of an axis-aligned box representing the part of the scene to be included in the WIM. Once delimited, users are allowed to manually scale up and down the active box to suit their needs.
- **WIM revealing:** cull-away occluding geometry so that *relevant* interior objects appear unoccluded in the WIM. Here by relevant objects we mean objects which are likely to be involved in the user task.

We address the two problems above by using a cell decomposition of the scene computed as preprocessing. Cells are detected by using a region growing algorithm on a discrete distance field representation of the scene. The benefits of the distance field and the cell decomposition for the WIM are manifold. On the one hand, cells are particularly suitable to define the extents of the WIM whenever the user navigates to another location (Figure 4(b)), and the cell center is a good choice as pivot for hand-held manipulation of the WIM. On the other hand, a rough polyhedral approximation of the geometry enclosing the cell can be used to precompute a set of potential slicing planes to cull away occluding geometry, while preserving as much local context information as possible (Figure 4(c)). Instead of allowing the user to establish arbitrary plane orientations on a continuous space (Figure 4(d)), we discretize the set of potential clipping planes by allowing only those planes roughly aligned with the faces of the main occluders (Figure 4(c)). Note that clipping planes located elsewhere (for example, in the middle of a room) are likely to remove important context information (Figure 8(b)).

3.2. Cell decomposition and approximation

As a preprocess, we compute a cell decomposition of the scene and create an rough polygonal approximation for each cell. Our cell decomposition algorithm is strongly based on the volumetric methods described in [HDS03, AVF04] and hence it is not claimed to be a contribution of this work. We first convert the input model into a voxelization and then compute a discrete distance field (Figure 5(a)), where each voxel encodes the distance to the closest geometry. The grouping of voxels into cells is accomplished by a region-growing process starting from the local maxima of the distance field [AVF04].

The next step is to build a polygonal approximation for the geometry enclosing each cell. We considered several ways to get a rough approximation of the geometry enclosing the cell: (a) build a coarse LOD representation via a surface simplification algorithm operating on the input scene, and (b) computing a rough polygonal approximation of the cells (each cell consisting of a collection of voxels) using conventional surface extraction algorithms from an implicit representation. In turns out that the second option is more preferable for our purposes, since it works well at coarse-level approximations and it produces water-tight approximations by automatically filling openings such as doors and windows. Our prototype system uses the surface extraction algorithm described in [AAB02]. Polygonal approximations of the cells of the test building are shown in Figure 5(b).



(a) (b)

Figure 5: *Cell decomposition and approximation: (a) discrete distance field of the test building; (b) polyhedral approximation of the cells (only one floor is shown).*

3.3. WIM Delimitation

Whenever the user navigates to another room, it is important to update the part of the scene covered by the WIM, which we represent as an axis-aligned box. Such an update should be performed in a way that the replicated part provides a clear view of the part of the scene visible from the immersive viewpoint. This problem is somewhat related to best-view selection, but instead of optimizing the view direction (remember that the WIM is hand-held) we want to optimize the part of the scene presented to the user. Our strategy for WIM delimitation is based on the cell decomposition computed during preprocessing. We first identify which cell the user is located at, and then update the active box to match that the bounding box of the cell (Figure 6).

A WIM showing the current cell provides a suitable level of scale for performing local manipulation tasks, but not for tasks such as moving an object to a distant room or quickly changing the viewpoint to another location. That is why users are allowed to manually scale the box to suit their needs. The center of the cell is used as the pivot for hand-held manipulation, and as the reference point for scaling up and down the box.

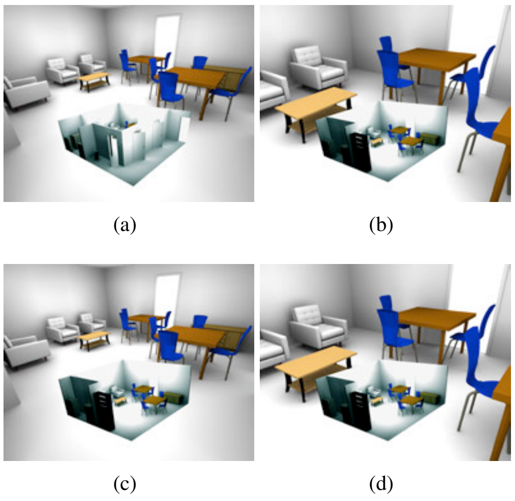

(a) (b) (c) (d)

Figure 6: *Benefits of using cells for initializing the WIM. Top: WIM centered at user's location, before (a) and after (b) the user navigated to approach the room center. Bottom: WIM centered at the current cell (our approach).*

3.4. WIM Revealing

Note that the WIM delimitation strategy described above does not solve the occlusion problem due to enclosing geometry such as walls. WIM revealing deals with culling away occluding geometry so that *relevant* interior objects appear unoccluded in the WIM. Here by relevant objects we mean objects which are likely to be involved in the user task, either actively (e.g. the object to be selected) or passively (e.g. objects helping the user to recognize the object to be selected). WIM revealing is particularly important when the user scales up the active box to include a larger portion of the scene, which in turn increases the depth complexity of the WIM (see e.g. Figure 3).

Our approach for revealing relies on the rough polygonal approximation of each cell computed during preprocessing. The reconstructed geometry is used to precompute a set of potential slicing planes to cull away occluding geometry, while preserving as much local context information as possible (Figure 4(c)).

Our revealing strategy can be seen as an extension of a hand-held clipping plane, with some key modifications affecting its behavior. Instead of allowing the user to establish arbitrary plane orientations on a continuous space (such orientations could be given isomorphically by a hand-held 6-DOF device), we discretize the set of potential clipping planes by allowing only those planes roughly aligned with the faces of the main occluders (Figure 4(c)). For example, in the case of a building, the only candidates for horizontal slicing planes will be those defined by the building floors. Discretizing the plane orientations in such a way allows users to cull away only occluding geometry such as walls while preserving the visibility of interior objects. Note that clipping planes located elsewhere (for example, in the middle of a room) are likely to remove important context information (Figure 4(d)). A second benefit of the above discretization is the higher stability of clipping planes against local movements (and thus higher stability of the part of the WIM shown to the user). For path shown in Figure 4, we would switch the slicing plane only four times (once for each time the hand enters a new room), whereas the hand-held clipping plane option would produce a different slicing plane for every frame.

Another key aspect of our approach is that we use only the position (and not the orientation) of the user's dominant hand to choose the slicing planes. We assume that the position of the user's hand inside the WIM is a good indication of the part of the scene the user needs to see to perform selection and manipulation tasks. When using a virtual hand metaphor to manipulate the WIM, this assumption appears to be quite reasonable. However, coupling the orientation of the slicing plane to the orientation of the dominant hand would interfere with manipulation tasks involving object rotation. Instead, we automatically choose a properly-oriented slicing plane according to the viewing direction of the WIM.

The algorithm for WIM revealing is as follows. Let H be the position of the user's dominant hand (the one used to select and manipulate objects in the WIM) w.r.t. the miniature replica, and let E be the position of the user's head (Figure 7). First we setup an auxiliary camera centered at H and oriented towards the viewer E with a user-defined field-of-view (we used a field-of-view of 30 degrees for the experiments). Then we determine the set F of faces of the rough polygonal approximation of the cell containing H that are visible from the newly defined camera. This can be implemented easily by rendering the cell approximation in OpenGL's selection mode. Finally, we choose as slicing planes up to three distinct planes corresponding to the faces in F with largest screen-projected area. Figure 7 shows an example where the auxiliary frustum intersects two walls and the ceiling (shown as semitransparent faces for clarity).

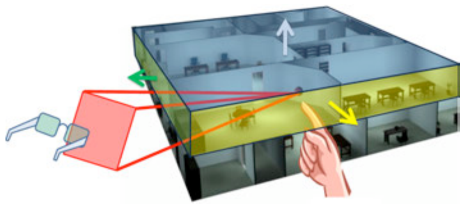


Figure 7: *Elements involved in the choice of the slicing planes to cull away occluding surfaces. In this case the frustum intersects three faces resulting in three slicing planes.*

The chosen slicing planes need to be slightly offset in the opposite direction of their normals to guarantee they cull away the original faces they come from. In our implementation the offset was set to roughly the length of a voxel. Figure 8 compares our approach with a hand-held clipping plane and a disk-based cut-away view. Note that the hand-held clipping plane often culls away objects in the room containing the user hand (Figure 8(b)). The same applies to the cut-away option, which also makes it harder to gather the spatial relationship of the revealed part with respect to the rest of the scene. The quality of our results can also be observed in the accompanying videos.

3.5. Time evolution of the slicing planes

As mentioned above, we choose up to three slicing planes to keep relevant objects visible. These planes will be referred to as the *working set*. The orientation of the slicing planes have a large impact on the part of the WIM shown to the user. Therefore, it is important to minimize both the frequency and the magnitude of the changes in the orientation of the planes in the working set, as continuous or abrupt changes might distract the user. In particular, it is important to keep tracking errors and hand trembling from producing abrupt changes on the display.

We can distinguish three kinds of events which might cause the working pair to be updated: (a) a change of the position of the user's hand inside the WIM, (b) a movement of the non-dominant hand, which controls the location and orientation of the WIM, and (c) a change of the user's viewpoint. Events of type (a) are easily filtered by using as reference point H the local maximum of the distance field closest to the hand position rather than the hand position itself. This makes the working set less sensitive to noise or involuntary movements. Events of types (b) and (c) basically involve a change in the relative position of the WIM w.r.t. the viewer. We considered the option of restricting the clipping plane to 45 degree discrete cuts (as suggested in [Bro03]), but this option was found to be a bit unnatural. Instead, we propose three orthogonal approaches to minimize the impact of abrupt changes on the display (see accompanying videos):

Lazy updates For a new plane Π' to replace a plane Π already in the working set, we require the importance value of Π' to exceed that of Π by a given user-defined percentage. We found a percentage of a 25% to be addequate for preventing approximate ties between two slicing planes to cause unnecessary updates of the working set.

Update filter Instead of executing the updates of the working set immediately, we can delay the update for a few milliseconds to prevent involuntary movements to cause an update. This is particularly useful to prevent unnecessary updates when the user is manipulating objects near a wall or other occluding surfaces whose support planes are potential slicing planes.

Smooth transitions between slicing planes Suppose that a given plane Π_1 has to replaced with Π_2. We define a smooth transition Π_t between planes Π_1 and Π_2 with $t \in [0,1]$ as follows. If Π_1 and Π_2 are parallel, then we define Π_t by simple linear interpolation, i.e. $\Pi_t = [a_1\ b_1\ c_1\ d_t]^{\mathbf{T}}$ with $d_t = t \cdot d_1 + (1-t) \cdot d_2$. This has the effect of a plane moving at constant speed from Π_1 to Π_2. If Π_1 and Π_2 are not parallel, we use their intersection line $\mathbf{A}$ as the axis for the rotation of Π_1 onto Π_2, i.e. $\Pi_t = [a_t\ b_t\ c_t\ d_t]^{\mathrm{T}} = R_A(t \cdot \theta)[a_1\ b_1\ c_1\ d_1]^{\mathrm{T}}$, where $R_{\mathbf{A}}(t \cdot \theta)$ is the 4x4 rotation matrix of $t \cdot \theta$ degrees about axis $\mathbf{A}$. In our experiments the animation took only about 100ms to complete.

4. Evaluation

We conducted a user-study to evaluate potential advantages of our algorithms in comparison with competing approaches. We focused on discovery, selection and manipulation tasks performed in spatially-immersive displays such as CAVEs. The test model was a fully-equipped three-storey building (Figure 4(a)) with about 80 rooms and 1.1M polygons. Figures 4(b) and 5(b) show the results of the cell decomposition step on the test model, using a 256^3 voxelization. The running time for the cell decomposition was 30 seconds, including the distance transform and polyhedral approximation steps. The resulting polyhedral approximation had 962 faces and a maximum approximation error of 15cm.

Regarding WIM revealing, we considered three options for distractor removal: our approach (object-oriented slicing planes), a hand-held clipping plane controlled isomorphically by a 6-DOF input device [Bro03], and transparency (using unsorted alpha blending with write operations on depth buffer disabled). Users were able to manually scale up and down the active box at constant speed (2.5 m/s for the

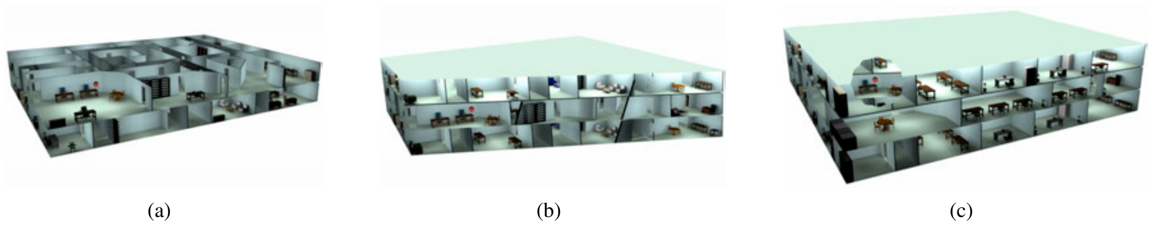


Figure 8: *Comparison of different strategies for WIM revealing: (a) slicing planes oriented along main occluders (our approach); (b) hand-held clipping plane; (c) cut-away view. The viewer and hand positions were the same in all cases.*

experiments), using two buttons. The adjustment of the active box only affected the WIM coverage; the apparent size of the WIM remained constant during the experiment (about 40 cm^3).

Given that our WIM revealing strategy uses slicing planes oriented along the main occluding surfaces, in the best case scenario one could expect our approach to have a positive impact on selection performance. In practice, however, this may not be the case. On the one hand, the discretization of the potential slicing planes reduces flexibility. On the other hand, the part of the scene shown in the WIM might change too abruptly and distract the user. In any case, rendering semi-transparent objects increases the visual complexity so we expected the transparency option to have a poor behavior.

Apparatus All the experiments were conducted on a four-sided CAVE at 1280x1280 resolution. The input device was a 6-DOF Ascension Wanda and a tracking system with two receivers providing 60 updates/s. The experiment was driven by 2.66GHz QuadCore PCs with Nvidia Quadro 5500 cards.

Participants Nine volunteers (2 female, 7 male), aged from 24 to 32, participated in the experiment, all with some experience with VR applications.

Procedure We designed three tasks involving selection and manipulation of 3D objects with increasing distances from the user's position to the object to be selected, and from the initial object position to its final destination, thus requiring varying levels of scale and thus increasing size of the active box. Starting from a given room A, the user was required to select an object lying on a different room B and move it to a target position located on a third room C. Rooms A, B and C where adjacent to each other for task 1, 20m apart for task 2, and on different floors for task 3.

Design A repeated-measures within-subjects design was used. The independent variable was the revealing strategy (transparency, hand-held clipping plane, and our approach). Each participant performed the experiment in one session lasting approximately 15 min. Before each task users were provided with a short training session.

Results Results of the experiment are shown in Figure 9(a). For task 1 (involving neighboring rooms), the hand-held clipping plane option was found to be significantly slower than our approach and transparency ($p < 0.04$), users performing a 50% slower with the hand-held plane. We found no significant differences between using our approach and semi-transparent objects. The good behavior of the transparency option was expected due to the low depth complexity of the WIM required to accomplish the task. We did not expected such a poor behavior of the hand-held clipping plane; this point will be discussed later. For task 2 (involving rooms far apart), a few pilots were enough to prove that the alpha blending option was completely unusable to the extent of making impossible the completion of the task. As we expected, transparency only works well with low depth complexity. Again, the hand-held clipping plane option was found to be significantly slower than our approach ($p < 0.01$), users performing much faster with object-oriented slicing planes. For task 3 (involving rooms on different floors), our approach was again found to be significantly faster that hand-held clipping ($p < 0.01$) and transparency ($p < 0.05$). To summarize, our approach was significantly better than the hand-held clipping plane for all three tasks, and clearly outperformed transparency except for the task requiring a WIM with low depth complexity, where both techniques had a similar behavior.

Concerning user preferences, all users preferred our approach, although a few users reported that the sudden changes on the part of the WIM shown were a bit confusing, until they noticed that the revealing was dependent on their hand position. Regarding the hand-held clipping plane, most users complained about the frequency objects were appearing and disappearing from the WIM. Users had difficulties in placing the clipping plane in such away that distracting objects were removed while preserving the visibility of the involved objects.

We also conducted an additional informal user study by designing a discovery task where users were requested to count the number of occurrences of a given object unevenly distributed throughout the first and second floor (the number of occurrences varied from 4 to 6). Users had 20 seconds to scale up the active box and explore the WIM to identify and count the occurrences of the requested object. We counted the number of missing objects. Results are shown in Figure 9(b). All users gave the correct answer when using our approach (no one object was missed), whereas the trans-

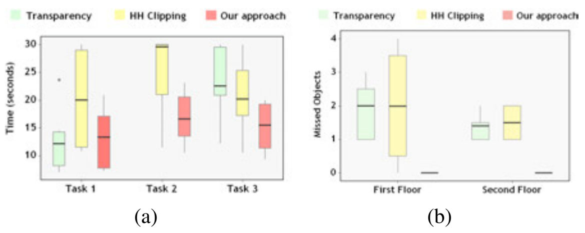


Figure 9: *Box plots for the first (left) and second (right) experiments.*

parency and hand-help clipping options exhibited a much poor behavior.

Runtime overhead We also measured the performance overhead of our technique. The overall overhead was found to be less than two milliseconds on the test hardware, which had no noticeable impact on the application frame rate. Since cells are detected during preprocessing, the WIM delimitation step introduces no noticeable runtime overhead in comparison with classic WIM implementations. WIM revealing requires drawing a low-polygon approximation of the current cell in select mode from an auxiliary camera, and using a fragment shader to discard fragments according to their classification w.r.t. the slicing planes. This per-fragment overhead affects only the rendering of the WIM, which usually covers a small portion of the screen.

5. Conclusions and future work

In this paper we have presented a few strategies for handling 3D occlusion while interacting with densely-occluded scenes. We have shown that a decomposition of the scene into cells with approximately constant visibility provides an adequate framework for both recomputing the part of the scene to be shown to the user, and for detecting and classifying the main occluding surfaces of the model. A key element of our approach is the algorithm for culling away distracting objects through a small set slicing planes aligned along the main occluding objects. The set of slicing planes is chosen in a view-dependent and hand-dependent manner to preserve critical context information and minimize view instability. We have shown empirically that our approach facilitates discovery, selection and manipulation tasks of 3D objects through their WIM proxies, providing a clear view automatically adapted to the user's location, and clearly outperforming competing approaches such as transparency and isomorphically-controlled hand-held clipping planes. There are several directions that can be pursued to extend the current work. Our revealing algorithm works better for culling away planar geometry, being less efficient for removing curved occluding objects. This might have a negative impact on performance when curved occluders surround relevant objects from multiple view directions. It would be useful to complement our slicing plane technique to better handle these cases. This could be accomplished by using the stencil-buffer in combination of the low-polygon approximation of the cells. Although we have focused on its implementation for the WIM, it may be interesting to explore the application of our revealing strategy to manipulate directly the objects shown in the immersive view.

References

[AAB02] ANDÚJAR C., AYALA D., BRUNET P.: Topology simplification through discrete models. *ACM Transactions on Graphics 20,6* (2002), 88–105.

[AVF04] ANDÚJAR C., VÁZQUEZ P., FAIRÉN M.: Way-finder: Guided tours through complex walkthrough models. *Computer Graphics Forum 23,3* (2004), 499–508.

[BF08] BURNS M., FINKELSTEIN A.: Adaptive cutaways for comprehensible rendering of polygonal scenes. *ACM Transactions on Graphics 27*, 5 (2008), 1–7.

[Bro03] BROOKS A.: Aids for training real-world spatial knowledge using virtual environments. In *Research Papers of the Link Foundation Fellows* (2003), B.J. T., (Ed.).

[CCSD03] COHENOR D., CHRYSANTHOU Y., SILVA C., DURAND F.: A survey of visibility for walkthrough applications. *IEEE TVCG 9,3*, 3 (July-Sept. 2003), 412–431.

[CGV05] CHITTARO L., GATLA V. K., VENKATARAMAN S.: The interactive 3d breakaway map: A navigation and examination aid for multi-floor 3d worlds. In *CW '05: Proceedings of the 2005 International Conference on Cyberworlds* (Washington, DC, USA, Nov. 2005), IEEE Computer Society, pp. 59–66.

[CH06] COFFIN C., HOLLERER T.: Interactive perspective cutaway views for general 3d scenes. In *3DUI '06: IEEE Symposium on 3D User Interfaces* (2006), pp. 25–28.

[DWE03] DIEPSTRATEN J., WEISKOPF D., ERTL T.: Interactive cutaway illustrations. *Proceedings of Eurographics 2003* (2003), 523–532.

[EAT07] ELMQVIST N., ASSARSSON U., TSIGAS P.: Employing dynamic transparency for 3d occlusion management: Design issues and evaluation. In *Proceedings of INTERACT 2007* (2007), pp. 532–545.

[ET08] ELMQVIST N., TSIGAS P.: A taxonomy of 3d occlusion management for visualization. *IEEE Transactions on Visualization and Computer Graphics 14*, 5 (2008), 1095–1109.

[HDS03] HAUMONT D., DEBEIR O., SILLION F.: Volumetric cell-and-portal generation. *Computer Graphics Forum 3-22* (2003).

[LFKZ01] LAVIOLA JR. J. J., FELIZ D. A., KEEFE D. F., ZELEZNIK R. C.: Hands-free multi-scale navigation in virtual environments. *Proceedings of the Symposium on Interactive 3D Graphics'01* (2001), 9–15.

[RT09] R. TRUEBA C. ANDUJAR F. A.: *Smart Graphics*, vol. 5531/2009 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2009, ch. Complexity and Occlusion Management for the World-in-Miniature Metaphor, pp. 155–166.

[SCP95] STOAKLEY R., CONWAY M. J., PAUSCH Y.: Virtual reality on a wim: interactive worlds in miniature. *SIGCHI'95: SIG on Human factors in computing systems* (1995), 265–272.

[WHB06] WINGRAVE C. A., HACIAHMETOGLU Y., BOWMAN D. A.: Overcoming world in miniature limitations by a scaled and scrolling wim. *3D User Interfaces* (2006), 11–16.