Interactive Tools for Virtual X-Ray Vision in Mobile Augmented Reality

Ryan Bane, Tobias Höllerer
University of California, Santa Barbara
{sporky, holl}@cs.ucsb.edu

Abstract

This paper presents a set of interactive tools designed to give users Virtual X-Ray vision. These tools address a common problem in depicting occluded infrastructure: either too much information is displayed, confusing users, or too little information is displayed, depriving users of important depth cues. Four tools are presented: The Tunnel Tool and Room Selector Tool directly augment the user's view of the environment, allowing them to explore the scene in direct, first person view. The Room in Miniature Tool allows the user to select and interact with a room from a third person perspective, allowing users to view the contents of the room from points of view that would normally be difficult or impossible to achieve. The Room Slicer Tool aids users in exploring volumetric data displayed within the Room in Miniature tool. Used together, the tools presented in this paper can be used to achieve the virtual x-ray vision effect. We test our prototype system in a far-field mobile augmented reality setup, visualizing the interiors of a small set of buildings on the UCSB campus.

1. Introduction

This paper presents a set of interactive tools aimed at giving users of mobile augmented reality systems new interfaces to make sense of the world around them. In particular, this paper examines the possibility of giving users virtual x-ray vision—the ability to see through walls or other solid objects.

For the purposes of this paper, we consider x-ray vision to be the act of visualizing some target through at least one layer of occluding structure. Fundamental work on direct visualization of such targets using Augmented Reality (AR) has been reported, and the "Superman's X-Ray Vision" problem has been defined, in which showing too much information about occluding structure confuses users, but not showing any deprives the user of important depth cues, rendering the visualization ambiguous [16]. The tools presented in this paper allow users to interactively select the information to be displayed in an attempt to provide the most useful visualization.

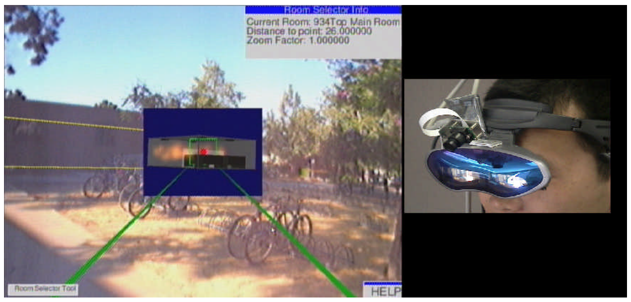


Figure 1, Virtual x-ray vision, using an Augmented Reality research system.

While the general concepts of our interactive tools are applicable to visualize occluded objects of any kinds, we particularly focus on x-ray vision as applied to buildings and general volumetric data. Buildings are large, complex, three-dimensional structures with easily viewable real world counterparts. Because of their size and internal complexity, objects in buildings may be occluded from view, and existing visualizations may simply confuse users. Volumetric data may represent heat distributions, smoke concentration, wireless network strength, or other data that may not be easily visible to users. Important features in one part of a volume may be occluded or obscured by data nearer to the viewer.

Potential applications of virtual x-ray vision are numerous. With an accurate model of a building's floor plan and wiring or pipes, electricians or plumbers could more easily plan their work or examine the layout of the infrastructure. Combined with sensor data, such as heat and smoke detectors, emergency response teams could use such systems to plan routes through buildings, view important data about rooms before entering them, and in general be more prepared for what they may find. Combined with security cameras and different kinds of sensors, security guards could examine rooms without physically entering them. NASA's Johnson Space Center has sketched an application of an AR "X-Ray Window" that may one day offer crewmembers of the International Space Station interior views of different station modules and storage bins [23].

In the previously mentioned user study on the influence of occlusion representations in direct AR overlays [16], even with the best graphical representation, users misjudged the occlusion relationship of a target object with respect to a series of three occluding test objects in about 10% of all the trials. On average (across all tried representations), the correct occlusion relationship was misjudged in 21% of the trials. Also, as intuitive as the direct visualization method is, depicting occluded infrastructure in a single view is not scalable with the number and complexity of the occluded objects. With more layers of occlusion, the confusion rises dramatically. The direct overlay visualizations for occlusion do not scale beyond a small number (3-4) of layers [16].

The main contribution of this paper is a toolset aimed at making x-ray vision through mobile augmented reality feasible. This toolset provides interactive methods of generating visualizations of objects or features behind multiple layers of occluding infrastructure, and allows users to selectively display the parts of the overall environment that give them the best view of the occluded spaces. By interacting with the environment over time, more information is passed to the user and a better understanding of the environment is imparted than with static display methods.

The system reported in this paper makes use of video see-through augmented reality, using a small camera and orientation tracker mounted on an HMD (shown in Figure 1). The software runs from a laptop computer, carried with the system in a backpack. At present, the system uses no positional tracking device, instead relying on users standing in specific, measured positions. The addition of such a tracker is an obvious necessity for real-life deployment, but not imperative for the testing and validation of the techniques presented in this paper.

2. Related Work

Our interaction techniques that aim to facilitate seeing through walls and occluding infrastructure are fairly typical examples of outdoor mobile augmented reality applications [8][13][18][19][16], which operate at medium to long range. This particular sub-field of AR has been referred to as far-field augmented reality [16]. Cutting [7] mentions occlusion, relative size, aerial perspective, and haze effects as the dominant depth cues for the far-field, as compared to motion parallax, binocular disparity, accommodation, and convergence, which are most effective at short range (and therefore not considered by us crucial factors for the interfaces described in this paper).

3D interaction techniques have been thoroughly explored for application in Virtual Reality. Interaction at a distance has been a particular focus [20][5][22]. We adapt the World in Miniature technique [22] to provide an overview of a particular subset of the occluded infrastructure (a room), which is selected by a simple 3D cursor, whose motion we control by a combination of head motion and a distance selector.

The concept of different layers of annotations that we employ is reminiscent of the idea of magic lenses and toolglasses that were originally developed for 2D and zoomable interfaces [3][4].

The concept of slicing is heavily used in the Visualization and Virtual Reality communities to obtain a good understanding of complex data distributions that can be mapped onto 2D planar segments of volumes. Example application domains are medicine [21][12] and the petroleum industry [17]. Direct volume rendering is another popular visualization technique for complex 3D data distributions [21][6].

Our tunnel tool, finally, is loosely based on the notion of cutaway views [10], as previously employed in AR by the KARMA system, an early indoor AR system that was applied to computer equipment maintenance tasks [9].

Our interactive tools have to work over much greater distances and many more levels of occlusion than anticipated by any of these systems.

3. Interface Core Concepts

The software is broken up into core concepts to support various aspects of the x-ray vision task. User input is supported by Tools, selection of important data is done using Layers, and rendering different information into different parts of the screen is done using Environments.

3.1. Tools

Users use tools to interact with their environment in various ways. Tools are started up and shut down as the user requests, allowing the user to choose what sort of interaction is appropriate for a given task. Tools can alter the display in first-person perspective, meaning the user's view of the real world is directly augmented, or in third-person mode, which allows the user to view selected parts of the virtual environment independently of the current view of the real world.

For the purposes of this paper, tools are started by keyboard input using a Twiddler2 input device [11]. In related work, we explored alternative modes of input, including vision based hand gesture recognition and voice input [14].

We decided to employ a modal interface for various reasons: first, because of the limited size of the mobile input device, the number of easily accessible input events is small, giving limited space to map key presses to tools and simple commands. Second, since several tools expect similar sorts of input, such as moving a 3d cursor, a modal interface of this sort allows for reuse of controls in a predictable manner. Finally, this approach keeps the system extensible, allowing future tools to reuse the interface and add new functionality to the system.

To handle cases in which multiple tools may be open, we use a stack metaphor to manage the current running set of tools. When an input event enters the system, it is given to the tool on the top of the stack. This tool can either consume the input event, in which case tools lower down in the stack do not process it, or it can pass it on to the next tool. If no tool uses an input event, it is dropped and ignored. This method of passing input through the system establishes a clear, predictable order of precedence between concurrently running tools. For the user's convenience, the tool stack is displayed in the lower left corner of the user's display.

3.2. Layers

The model of the environment is broken up into groups of objects, called layers – a familiar concept used in CAD systems and some paint programs. Our layers are semantically coherent sets of data, either physical or virtual. A layer may contain, for example, a particular set of volumetric data, all the pipes within a building, or any other conceptually related group of objects. These layers allow the user to specify the sort of objects they are interested in, and to turn off the rendering of objects deemed as unimportant.

3.3. Environments

Computer-generated imagery is divided into two types of overlays: data specific to the currently active tool and data shown independently of any particular tool. These two displays may show different data, for example if the user wants to see a wire frame superimposed on the real-world view of the building, but wants to see the contents of a room within the tool they are using. Each overlay is governed by an environment, which holds a set of active layers that determine what virtual objects should be displayed.

The Lens Environment holds the active layers for the tool dependant overlay. The Lens Environment gets its name from the virtual lens placed over the video image in first-person perspective tools, which creates a well-defined area for tool data display. Third-person perspective tools also display objects specified by the Lens Environment, but the graphics are not constrained to a lens, instead being displayed on a transparent glass-pane that covers the whole screen. These graphics are screen-stabilized, meaning that they "float" in front of the user and move with the user's head motion [1].

The Surroundings Environment stores active layers for data overlays that do not relate to any specific tool. When no tool is running, the layers specified by the Surroundings Environment are displayed over the video frame. When tools are active, these graphics are only shown on areas of the screen not covered by the tool's lens.

4. The Toolset

The system examines two approaches to the x-ray vision problem: room-based and volume-based. The room-based approach is based on the observation that buildings can be modeled as collections of rooms, and room-based tools allow the user to interact with the environment at the room level. Volume-based techniques make no assumptions about the structure of the environment, and allow users to interact with arbitrary geometry.

To handle 'classic x-ray vision,' in which the user simply needs to see through walls or occluding obstacles, tools are implemented from the first-person perspective. In addition, since users may want to see rooms or data clouds from viewpoints they would not normally be able to achieve, third-person perspective tools are provided.

In the case of first-person view tools, a virtual lens is superimposed on the video, clearly defining an area for the display in which graphics will be displayed. This acts as a sort of cutaway view [10] into the video, and avoids giving the appearance that the graphics are floating somewhere between the user and their correct three-dimensional position.

To counteract the effects of distance, all first-person tools can be zoomed, which enables users to see a magnified view in the lens. Text indicators report the current zoom factor. While it is true that zooming makes objects in the Lens Environment scale differently than those in the Surroundings Environment, it is assumed that since the user must manually zoom they will never be surprised by this effect — the usability of the system remains intact.

4.1. The Tunnel Tool

The Tunnel Tool is perhaps the most obvious solution to the x-ray vision problem. The tool gets its name from the bounds of its view region, which forms a frustum extending from the user's position out along the direction of view. The tool renders data inside this frustum, giving the effect of looking down a tunnel into

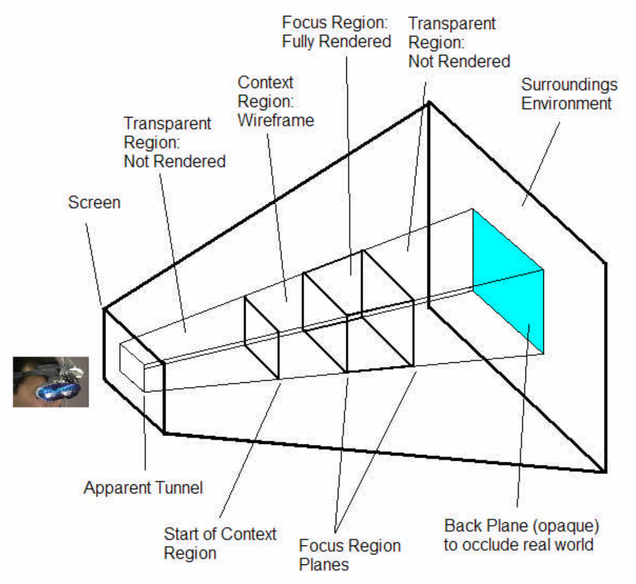

Focus Region: Fully Rendered
Transparent Region: Not Rendered
Context Region: Wireframe
Surroundings Environment
Transparent Region: Not Rendered
Screen
Apparent Tunnel
Start of Context Region
Focus Region Planes
Back Plane (opaque) to occlude real world

Figure 2, A conceptual diagram of the Tunnel Tool.

the geometry of the scene. Figure 2 shows a diagram of the Tunnel Tool's structure.

Inside the tunnel there are three planes, which split the space in the tunnel into the following regions:

The region between the first and second plane is rendered in wireframe, and is called the Context Region. The purpose of the Context Region is to provide the user with some context on what they are seeing without showing so much information that they would be confused. This means that if a user uses the Tunnel Tool while looking at a wall of a building, and this wall is in the Context Region, it will be rendered in wireframe, allowing the user to see through to whatever is behind it. The advantage here is that the user still sees the wall near what they are looking at without it getting in the way of the view.

The region between the second and third planes is called the Focus Region, and is rendered in solids. This is taken to be the region the user is interested in, and should be the focus of the view.

The regions between the user and the first plane, and behind the third plane are not rendered. These regions potentially contribute a great deal of information, which the user would have to sift through to find what it was they were looking for.

When the tool is in use, the user can slide the whole set of planes forward and backward in the tunnel by dragging a mouse (here: the Twiddler-2 trackpoint). This allows the user to view the scene as a progression of smaller, more easily interpreted slices of data. The depth of the focus and context regions can be adjusted by dragging the mouse with different button combinations, an operation that is used much less frequently than slicing through the volume.

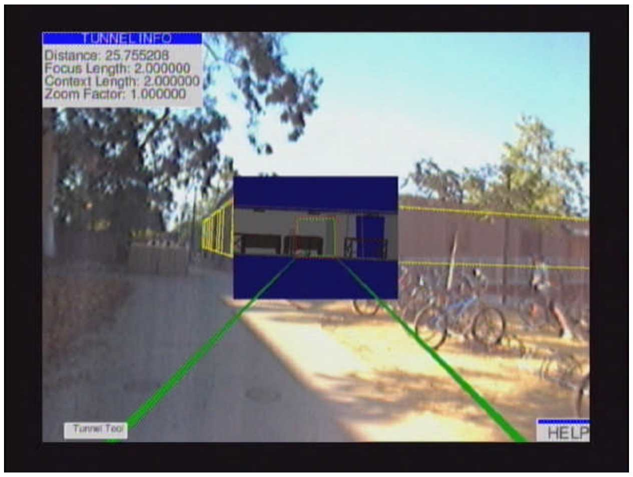

TUNNEL INFO
Distance: 25.755208
Focus Length: 2.000000
Context Length: 2.000000
Zoom Factor: 1.000000

Figure 3, Using the Tunnel Tool to view the area behind a wall.

Figures 3, 5, and 6 show images of the Tunnel Tool in use. Figure 3 shows a user slicing away an occluding wall and looking inside a room in the near building. Figure 4 displays a simulated heat distribution in the buildings, and figure 5 shows the use of the Tunnel Tool to view a small slice of it. The view in figure 5 displays a dense patch in the distribution that would otherwise be difficult to notice.

It is important to realize that the use of the lens causes problems of its own. The real world is partially occluded, so the depth context of the geometry displayed is unclear: the objects could be small and nearby or large and far away. To deal with this problem, two types of indicators are provided to help users judge the distance to

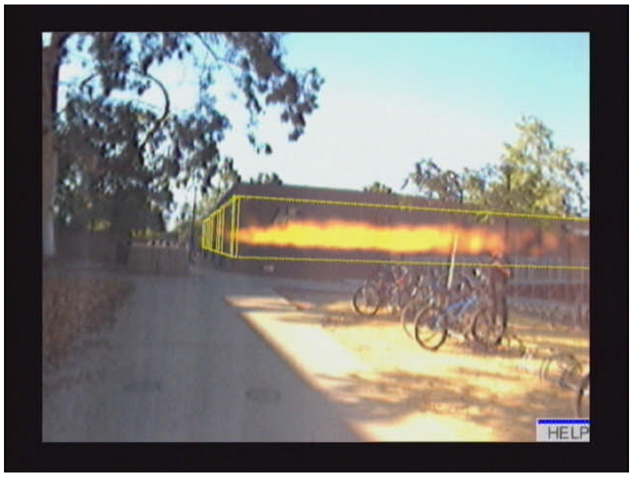


Figure 4, A simulated heat distribution.

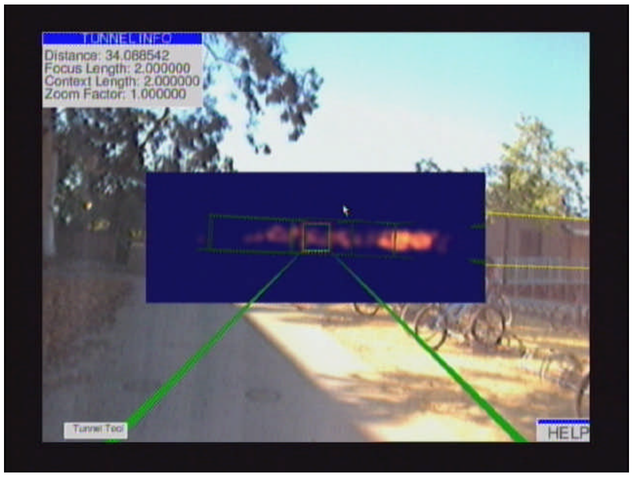

TUNNEL INFO
Distance: 34.088542
Focus Length: 2.000000
Context Length: 2.000000
Zoom Factor: 1.000000

Figure 5, Using the Tunnel Tool to view a slice of the heat distribution.

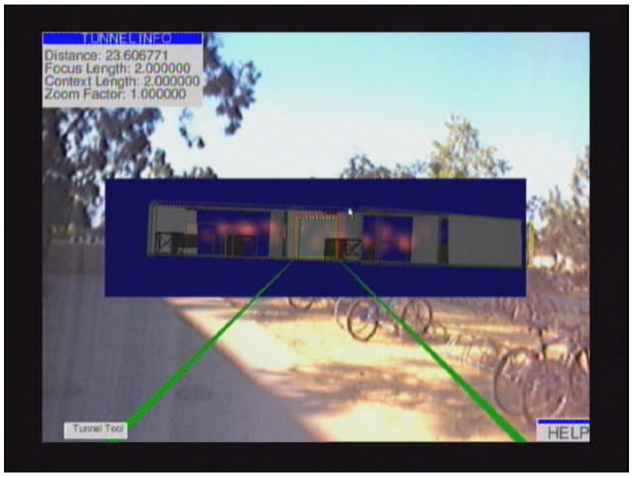

TUNNEL INFO
Distance: 23.606771
Focus Length: 2.000000
Context Length: 2.000000
Zoom Factor: 1.000000

Figure 6, Visual artifacts and partially displayed rooms may be hard to interpret

the planes. At medium and long range, perspective is a very useful distance cue [7], so the first set of indicators is shown as virtual three-dimensional objects. These indicators take the form of a pair of lines, or rails, extending outward from the user. A set of three upright boxes sits on these rails, indicating the position of each of the planes. As the user slides the planes backward and forward, these boxes shrink and grow with perspective matching their virtual distance from the user. The second indicator of distance is a text readout that reports the distance in meters to the first plane of the tool.

The Tunnel Tool is very useful for exploring volumetric data, allowing users to view the volume one narrow slice at a time. For general x-ray vision use, however, the Tunnel Tool tends to clip walls and other solid geometry at angles, leaving confusing artifacts and blue spaces between walls. Also, because the planes cut any geometry within the tunnel, as the distance between the user and the planes grows, an increasing number of rooms will be cut. This leads to a large number of partially-rendered rooms and objects, which can further confuse users. Figure 6 shows this effect, where the tunnel tool has been used to view a very short slice of data diagonally cutting a building. To avoid these problems, the Room-based approach and specifically the Room Selector Tool have been developed.

4.2. Room Selector Tool

The Room Selector Tool is the first-person perspective implementation of the Room-based approach. This tool allows the user to slide a three-dimensional cursor from the user's position out along a vector in the direction of view. When the cursor lies inside a room in the environment model, the virtual representation of the room is displayed in the tool lens. The system processes the geometry of the room and shows any walls that occlude space in the room from the user's view in wireframe, and walls that do not occlude room space as solid. Objects in the room will be shown or not shown depending on their Layer membership and the current active layers in the Lens Environment.

As with the Tunnel Tool, two indicators are provided to help users judge depth of the cursor. The text readout reports the distance to the point in meters. The virtual object indicator has the same basic structure of rails and an upright box, but also includes a solid sphere that indicates the precise position of the cursor. As with the Tunnel Tool, these virtual objects shrink and grow as the user slides the cursor forward and backward.

In addition, because an interactive sliding process generates the final image of a room, the user remembers the rooms between the start point and final room, and therefore should have some idea about the depth that is not shown in the rendering.

A sequence of rooms shown by the Room Selector Tool is shown in figure 7. This sequence was taken by extending the cursor to a distance at which it entered the building, and then looking left and right to view adjacent rooms.

Figure 8 shows the use of zoom features. In this case, the user has extended the cursor to a room in the most distant of the three buildings, and then zoomed in to get a closer look at the contents of the room. Zooming acts as a simple magnification of the view in the tool's lens. In this case, the view is magnified to be four times normal. Note that zooming causes AR registration to break down, since we currently only magnify the virtual graphics in the tunnel region. While users seem to cope with this concept of a magnifying lens quite well, we are planning

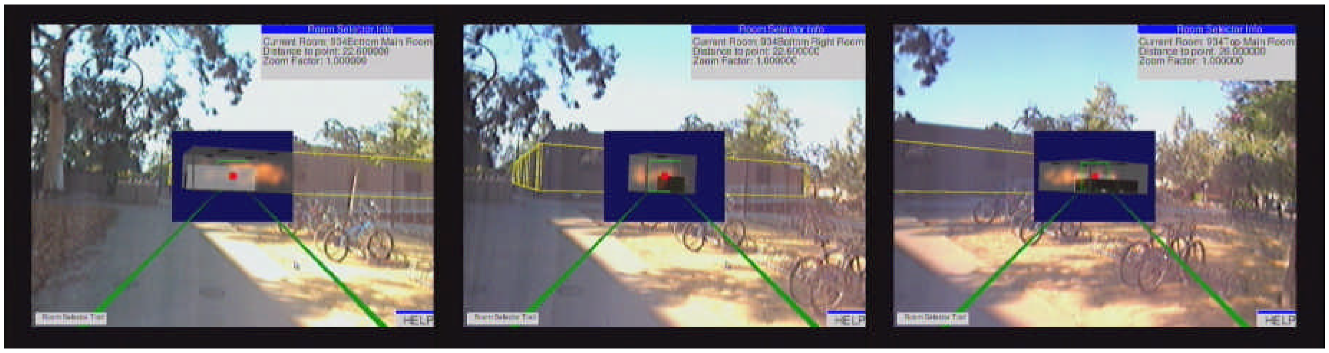

Room Selector Info
Current Room: 934Bottom Main Room
Distance to point: 22.600000
Zoom Factor: 1.000000
Room Selector Info
Current Room: 934Bottom Right Room
Distance to point: 22.600000
Zoom Factor: 1.000000
Room Selector Info
Current Room: 934Top Main Room
Distance to point: 26.000000
Zoom Factor: 1.000000
HELP

Figure 7, A sequence of rooms shown by the Room Selector Tool.

to compare this approach with one in which we zoom into the entire scene, including the camera image.

The Room Selector Tool solves the problems posed by the Tunnel Tool, namely confusing artifacts and the display of too many partial rooms. Unfortunately, it creates new problems. In particular, there may exist objects in the room that occlude other objects belonging to the same layer. In this case, using just the Room Selector Tool, the user would have to walk around and find a better viewpoint to look at the room to get around the occluding objects. In many environments, this may not be feasible. To solve this problem, we have developed the Room in Miniature Tool.

4.3. Room In Miniature Tool

The Room in Miniature Tool is the third-person perspective implementation of the Room based approach. It is in part based on the World in Miniature [22], to which it bears marked similarity. Third-person views for AR systems are not at all a new concept [1][20][19]. In our case, they are a convenient means to better resolve complex occlusion situations.

From the Room Selector Tool the user can enter the Room in Miniature Tool, which switches the view from first-person perspective to third-person perspective on the selected room. The user is then presented with a view of the room, with contents as determined by the Lens Environment, as in the Room Selector Tool. The important difference here is that this view is screen stabilized, meaning that it will stay in the user's view regardless of the direction they are looking. The room is fixed some distance away from the user, at which they can fit the whole room in their view. The room can be rotated about its center point, and the walls are rendered according to the new viewpoint. Instead of relying on orbital viewing [15] or other head-gesture control [1], we simply let the user control room rotation using the mouse (trackpoint). When selecting a Room in Miniature, the user effectively performs a context switch and now navigates in VR rather than AR. The choice of control reflects that and allows the user to keep oriented towards the physical room. Transition animations between the first-person and the third-person views help bridging the gap between AR and VR [13][20].

The net result of all this is that the user can select a room with the Room Selector Tool and by using the Room in Miniature Tool, view the room from angles that may not be possible from the user's current position. For example, if the user has selected a room that contains cubicles or other room dividers that block view of some target object, the user can simply open the Room in Miniature Tool, rotate the room around and look from the other side of the divider, or from through the ceiling, to get a top down overview of the room. Figure 9 shows exactly this situation, where the room includes a room divider, which blocks the view of room contents from some angles. The user has used the Room in Miniature Tool to rotate the room to a more useful position.

In addition to the room, an avatar of the user is also shown. The avatar is shown wherever the user is located relative to the actual center of the room. The idea is to

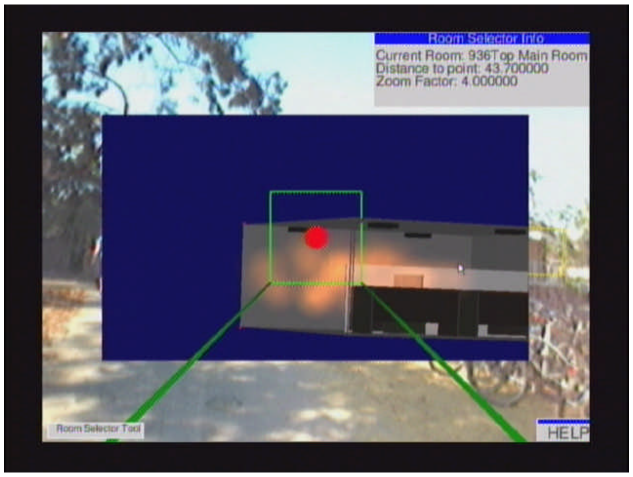
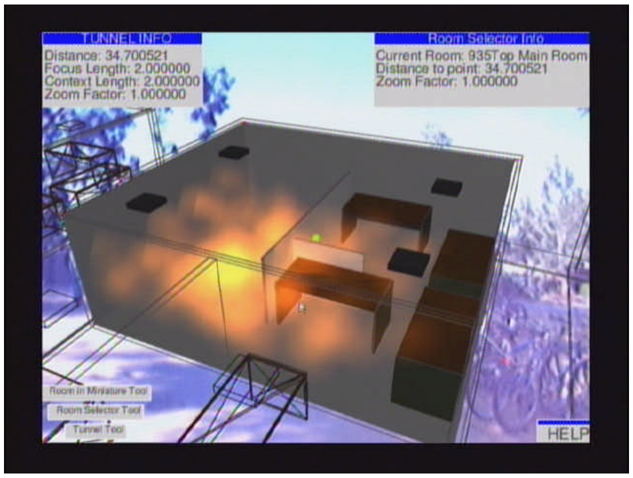


Figure 8, Using Zoom to magnify the view of a room in the Room Selector Tool.


TUNNEL INFO
Distance: 34.700521
Focus Length: 2.000000
Context Length: 2.000000
Zoom Factor: 1.000000

Room Selector Info
Current Room: 935Top Main Room
Distance to point: 34.700521
Zoom Factor: 1.000000

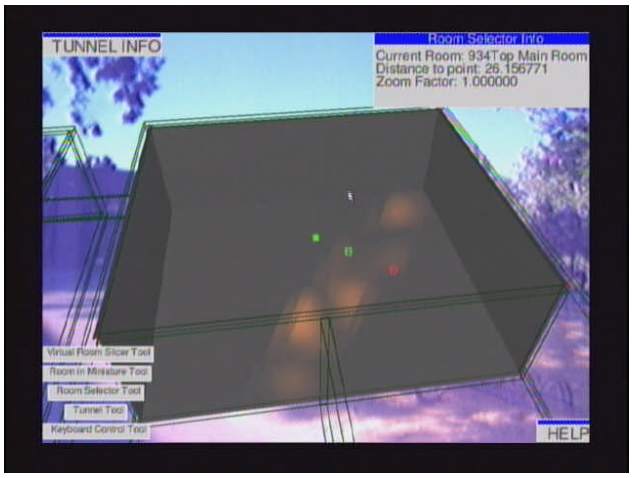
Figure 9, The Room in Miniature Tool allows users to see the room from viewpoints they could not normally achieve.

prevent the user from becoming disoriented while using the tool and being confused as to the layout of the room relative to their position.

4.4. Room Slicer Tool

The Room Slicer Tool is the third-person view implementation of the volume-based approach. The tool is used in conjunction with the Room in Miniature tool, meaning that the user first selects a room with the Room Selector Tool, then opens the Room in Miniature tool, and finally opens the Room Slicer.

The Room Slicer Tool allows the user to sift through geometry or volumetric data inside a given room, while retaining all the advantages of the Room in Miniature Tool. The tool is conceptually similar to the Tunnel Tool, in that it allows the user to set a pair of planes that select the data to display. Because the user should have plenty of context from the view of the Room in Miniature, the context region is not included in this tool. The Room Slicer Tool is most useful for viewing parts of volumetric clouds inside rooms, since the slice can be moved back and forth through the room (away from and towards the user's position, which is denoted by an avatar) and the room can be rotated around using the Room in Miniature functionality. The volumetric data can be constrained to interesting regions and viewed from angles not normally available from the user's position. The Room Slicer is particularly useful to understand the geometry of a particular slice depth and distance that may look confusing from a first person perspective. The top-down view of the slicing process, in conjunction with denoting the user's location with an avatar, significantly clarifies the viewing geometry.


TUNNEL INFO

Room Selector Info
Current Room: 934Top Main Room
Distance to point: 26.156771
Zoom Factor: 1.000000

Figure 10, The Room Slicer allows users to explore a data cloud while using the Room in Miniature Tool.

Figure 10 shows the Room Slicer tool in action. In this case, the user has enabled only a single layer in the lens, corresponding to the simulated heat distribution. The user is viewing a small slice of the data, possibly looking for important features in the volume. The room, and the slice, can be rotated about and viewed from any angle using Room in Miniature Tool functionality. The user's avatar is not currently in view. It is located off the right bottom corner in this case, following an imaginary line through the red and green spheres denoting the front and back planes of the focus region. The third sphere is marking the center of the selected room.

5. System Architecture

5.1. Hardware Architecture

The system runs on a Dell Precision M-50 laptop with a Quadro-4Go video card, running Microsoft Windows XP. Orientation tracking is done using an Intersense InertiaCube2. Video capture is done using a Point Grey Firefly camera. The InertiaCube2 and the Firefly are mounted on a Sony Glasstron PLM-S700 HMD.

User input is handled by a Handykey Twiddler2 keyboard, which includes a thumb-controlled track point, which we use to control our 3D cursor. In previous work [14] we considered other input modalities, such as hand gestures and speech, but these methods are not within the scope of this paper.

5.2. Software Architecture

The software used in this system is a custom OpenGL application using the OpenGL Utility Toolkit (GLUT).

The application communicates with a program that retrieves video frames from the Firefly camera and then paints those pixels into the background. The depth buffer is then cleared, and rendering is done over the camera's image. The final onscreen image is a result of multiple rendering passes: first, the Surroundings Environment is rendered, and once again the depth buffer is cleared. Tool dependant data is rendered over the combined Surroundings and video image.

Tools that use lenses make use of OpenGL's support for scissoring and clipping planes. Rendering is constrained to the lens with a call to glScissor, which defines a rectangular region of the screen in which graphics can be displayed. The Tunnel Tool and Room Slicer tool set up clipping planes to constrain rendering to the correct three-dimensional region. The Room Selector tool also makes use of clipping planes to constrain rendering to the space inside the selected room. Zooming is implemented by adjusting the field of view of the camera in OpenGL before rendering the contents of a lens. By halving or quartering the field of view, we effectively render at twice or four times magnification of the normal view.

Volumetric rendering is done using very simple textured splats [6], where each splat has a color with intensity and opacity determined by the value of its data point. The texture used is a simple circle, most intense in the center, dropping off linearly with distance. Each splat is billboarded to be continuously facing the camera.

The environment model is read in from an XML file, which represents the scenegraph and contains information about the layout of rooms within the building. This is necessary for the Room Selector and Room in Miniature Tools, which rely on knowledge of the room volumes within the building to function properly. LibExpat is used to do the XML parsing.

VRML models are supported through the CyberVrml library. VRML can be used to add objects to the environment, and provides a modeling language that is easier to use than the custom XML file format used for the general environment layout.

6. Discussion and Future Work

Static image display methods are limited in that they must either display all the information for a given scene or they must somehow select information to hide. This leads to the Superman's X-Ray Vision problem, because either the user will be confused or there will be circumstances in which the user cannot view the information they want. Interactive techniques are a way to get around these limitations: by letting the user interactively select the information to be displayed, the system is able to impart depth cues and display the information in a manner that does not confuse the user.

Future work should include the investigation of other interactive techniques, such as sliding tools that directly incorporate the best static visualization methods from [16], or interactively placed, world stabilized cross-section tools. So far we have only evaluated the usability of our tools by having them tested by several researchers in our laboratory. On top of these informal expert evaluations, we plan to conduct more formal user studies to evaluate the tools on controlled tasks, in order to better understand how inexperienced users of such systems want the tools to act, and to determine what indicators of depth are most useful for tools of this nature.

Virtual x-ray vision is almost undeniably useful. Interactive visualizations, rather than static ones, make the task of viewing spaces occluded by many walls or other obstacles much more manageable. To make the effect correct and useful, though, accurate modeling of the environment is necessary. In the case of buildings, an accurate model of the walls is obviously important for registration, but it is also important to realize that without accurate modeling of the contents of the rooms in a building, any information the user sees through the system will be erroneous.

Real world applications of virtual x-ray vision systems would probably need to interface with sensors and object tracking mechanisms, and offer specialized coverage of the environment rather than attempt to solve the general x-ray vision problem. Another option might be to incorporate multiple video feeds per room, and then display the video stream appropriate for a user's position and viewing angle.

7. References

[1] Bell, B., T. Höllerer, and S. Feiner, 2002: An annotated situation-awareness aid for augmented reality. In *Proc. ACM UIST 2002 (Symp. on User Interface Software and Technology)*, Paris, France, 213–216.

[2] Billinghurst, M., H. Kato, and I. Poupyrev, October 2001: The MagicBook: a Transitional AR Interface. In *Computers and Graphics,* **25**(5), 745–753.

[3] Bier, E. A., M. C. Stone, K. Fishkin, W. Buxton, and T. Baudel, 1994: A taxonomy of see-through tools. In *Proc. CHI 94*, ACM press, 358–364.

[4] Bier, E. A., M. C. Stone, K. Pier, W. Buxton, and T. DeRose, 1993: Toolglass and Magic Lenses: The see-through interface. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, Kajiya, J. T., editor, volume 27, 73–80.

[5] Bowman, D. A. and L. F. Hodges, 1997: An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments (color plate S. 182). In *Proceedings of the Symposium on Interactive 3D Graphics*, ACM Press, New York, 35–38.

[6] Crawfis, R. and N. Max, 1993: Texture splats for 3d scalar and vector field visualization. In *Proc. IEEE Visualization '93*, IEEE Computer Society Press, 261–266.

[7] Cutting, J.E., 1997: How the eye measures reality and virtual reality. *Behavior Research Methods, Instruments, and Computers*, **29**(1), 29–36

[8] Feiner, S., B. MacIntyre, T. Höllerer, and A. Webster, 1997: A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment. In *Proc. ISWC '97 (First Int. Symp. on Wearable Computers)*, Cambridge, MA, 74–81.

[9] Feiner, S., B. MacIntyre, and D. Seligmann, July 1993: Knowledge-based augmented reality. In: *Communications of the ACM*, **36**(7), 52–62.

[10] Feiner, S. and D. Seligmann, 1992: Cutaways and ghosting: Satisfying visibility constraints in dynamic 3D illustrations. In: *The Visual Computer*, **8**(5-6), 292–302.

[11] Handykey Corp., 2001: Twiddler2 chord keyboard. http://www.handykey.com.

[12] Hinckley, K., R. Pausch, J. Goble, and N. Kassell, 1994: Passive Real-World Interface Props for Neurosurgical Visualization, In *Proc. ACM CHI '94 (Conference on Human Factors in Computing Systems)*, 452-458.

[13] Höllerer, T., S. Feiner, and J. Pavlik, 1999: Situated Documentaries: Embedding Multimedia Presentations in the Real World. In *Proc. ISWC '99 (Third Int Symp. on Wearable Computers)*, San Francisco, CA, 79–86.

[14] Kölsch, M., R. Bane, T. Höllerer, and M. Turk Touching the Visualized Invisible: Wearable AR with a Multimodal Interface. Under review. Available at http://www.cs.ucsb.edu/~holl/pubs/kolsch-2004-tvi.pdf

[15] Koller, D.R., M.R. Mine, and S.E. Hudson, 1996: Head-Tracked Orbital Viewing: An Inter-action Technique for Immersive Virtual Environ-ments. In *Proc. UIST '96 (ACM Symposium on User Interface Software and Technology)*, 81–82.

[16] Livingston, M. A., J. E. Swan II, J. L. Gabbard, T. Höllerer, D. Hix, S. J. Julier, Y. Baillot, and D. Brown, 2003: Resolving multiple occluded layers in augmented reality. In *Proc. ISMAR '03 (Int. Symposium on Mixed and Augmented Reality)*, Tokyo, Japan, 56–65.

[17] Midttun, M. and C. Giertsen, 1998: Petroleum applications of virtual reality technology: Introducing a new paradigm. http://www.seg.org/meetings/past/seg1998/techprog/int5/papr327.pdf.

[18] Piekarski, W., B. Gunther, and B. Thomas, 1999: Integrating virtual and augmented realities in an outdoor application. In *Proc. IWAR '99 (Int. Workshop on Augmented Reality)*, San Francisco, CA, 45–54.

[19] Piekarski, W. and B. Thomas, 2001: Tinmith-Metro: New outdoor techniques for creating city models with an augmented reality wearable computer. In *Proc. ISWC '01 (Fifth Int. Symp. on Wearable Computers)*, Zürich, Switzerland, 31–38.

[20] Poupyrev, I., M. Billinghurst, S. Weghorst, and T. Ichikawa, 1996: The go-go interaction technique: Non-linear mapping for direct manipulation in VR. In *Proc. UIST '96 (ACM Symposium on User Interface Software and Technology)*, 79–80.

[21] Serra, L., T. Poston, N. Hern, C. B. Choon, and J. A. Waterworth, 1995: Interaction techniques for a virtual workspace. 79–80.

[22] Stoakley, R., M. Conway, and R. Pausch, 1995: Virtual reality on a WIM: Interactive worlds in miniature. In *Proceedings of Human Factors in Computing Systems (CHI '95)*, 265–272.

[23] White, W.W., 2004: X-Ray Window: Portable Visualization on the International Space Station. In *SIGGRAPH '04 DVD ROM,* Sketches, Session: Monkeying with Reality, p. 3