

Constrained Navigation in Immersive Virtual Reality

Andrew J. Hanson

Eric A. Wernert

Computer Science Department
Indiana University
Bloomington, IN 47405 USA

Abstract

Finding one's way through a complex virtual environment is a standard task in virtual reality applications. Starting from previous work on mouse-driven constrained navigation for 3D desktop displays, we address the problem of enhancing the effectiveness of virtual environments using the constrained navigation paradigm. The fundamental prerequisite for the appropriateness of constrained navigation — that the designer can significantly improve the quality of the user's experience by choosing a predetermined parametric set of viewing parameters or algorithms — is identical, but the natural tools and characteristics are quite distinct. One basic difference is the need to support head-tracked motion within a viewing environment that limits the user's physical motion; another is the increased need to stabilize the direction of the forward view while allowing significant information to be displayed in peripheral directions. Attempting to control the "glass elevator" of a spatially immersive virtual environment such as the CAVE leads to several possible ways of replacing the desktop environment's mouse. We have concentrated on exploring two of these modes: using the position of the viewer relative to the floor (which replaces the mouse pad), and tracking gesture vectors of the wand controller or thumb joystick. The presence of three dimensions of controller space also allows the use of one-parameter "onion layer" families of 2D constraint spaces, substantially enriching the control possibilities. We illustrate the approach with a variety of examples, emphasizing the possibility of topologically nontrivial navigation spaces.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques. I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism. I.3.8 [Computer Graphics]: Virtual Reality—Applications.

Keywords: Navigation; Constrained Navigation; Viewing Control; Camera Control; Virtual Reality

1 Introduction

A typical virtual reality (VR) environment is defined by a "visualization pipeline" consisting of the abstract geometry of the scene or data representation, filters applied to that data to emphasize the task-critical components, a mapping into a renderable representation, and finally the scene rendering itself. In this paper, we address the question of extending the task-critical filtering step to include customized selection of parameterized viewpoints to be used in the rendering step. We define "constrained navigation" for the purposes of our treatment as the restriction of viewpoint generation and rendering parameters to goal-driven subclasses whose access is specified by the application designer. We will focus for definiteness on navigation through large-scale spaces, so the user's immediate environment appears to be executing a movement in the overall scene; we will not concern ourselves with applications in which the objects of interest are manipulated within the user's grasp.

The need for constrained navigation is particularly critical in immersive VR applications because the user can typically use both head-tracker motion within the *Physical Viewing Environment* (PVE) and motion of the entire PVE through the virtual environment to select scene views. Many times the freedom to move arbitrarily is counterproductive, leading to the "lost in space" syndrome. For example, consider moving through a complex curved tube representing a 3D hallway, a mathematical structure, or an injection mold: if no constraints are placed on viewer motion, one can easily crash through the walls; if only wall constraints are imposed, one often winds up facing against a wall instead of gliding smoothly through the tube looking at the significant features that are intended to be the main content of the application. The authors have had many frustrating experiences both as navigators and observers of VR applications where the dominant memory is of flying upside down in the fog or careening through the environment for long periods of time wondering what was supposed to be happening.

While ordinary computer viewpoint animation is in fact a species of constrained navigation consisting of a linear time sequence of camera models, we will be concerned with generalizing this standard method to 2D or even 3D parameter spaces to generate viewpoints in immersive VR applications. We will extend and contrast previous work on geometry-driven space navigation [6] and desktop constrained navigation [7] to handle VR environment issues. Early important work on intelligent navigation upon which we build includes that of Mackinlay et al. [8] and Phillips et al. [9], as well as motion control systems like those of Ware and Osborne [13], Drucker et al. [4], and Robinett and Holloway [10]. The addition of extra intelligence in the interface is illustrated by the work of Billingham and Savage [1] in VRAIS '96. The acquisition of cognitive maps, a special and limited application of constrained navigation, has attracted a great deal of attention as well, as exemplified in the work of Thorndyke, Goldin, and Hayes-Roth [5, 12, 11]; their general classification of spatial knowledge for geographic navigation into landmark knowledge, procedural knowledge, and survey knowledge is suggestive, though possibly not sufficiently broad to encompass navigation issues in some current VR applications.

In the following sections, we introduce a specific model for the constrained navigation task in immersive VR environments, outline the mathematical procedures for implementing the paradigm, and analyze a family of alternative navigation modes for commonly occurring application scenarios. We conclude with examples from our trial implementations and some preliminary user-evaluation observations.

2 User Model for Constrained VR Navigation

The fundamental user model that we introduce in this paper may be thought of symbolically as a "glass elevator," a room with windows corresponding to the display walls of a typical immersive VR environment such as the CAVETM [3]. In order to extend this user

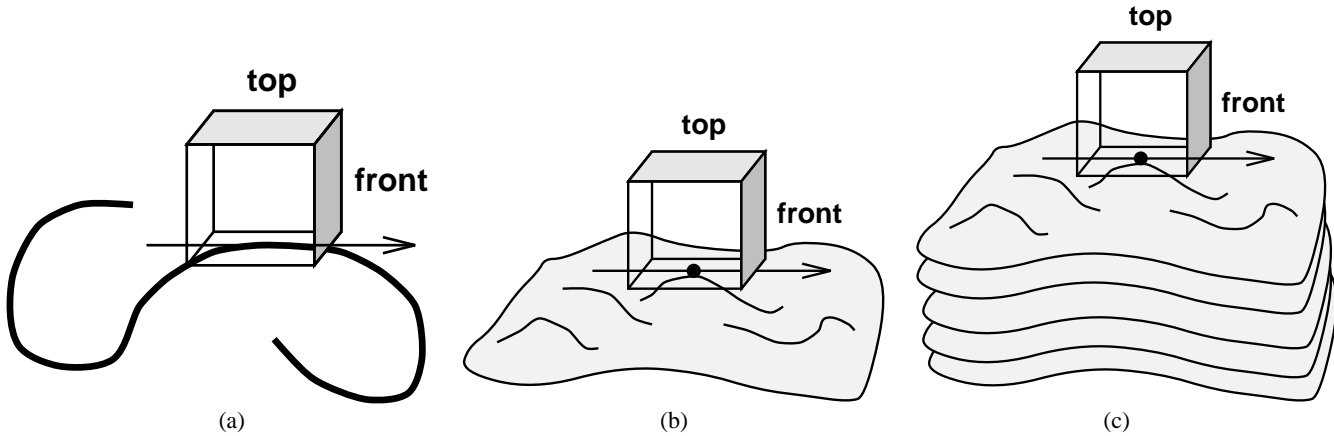


Figure 1: Scenarios with orientation of a room-like Physical Viewing Environment constrained by intrinsic properties of the guide manifold. (a) The PVE gliding on a 1D roller coaster, generating a standard animation sequence. (b) The moving PVE restricted to follow the constraints of a 2D glass roller coaster. (c) A 3D “onion” of nested 2D roller coasters, allowing restricted view changes keyed to a 3D controller.

model to other VR environments such as head-tracked displays, we will refer to the virtually movable space in which the user interacts as the PVE, or physical viewing environment. Following the CAVE paradigm, we will separate motion of the PVE’s own coordinate system in the scene environment from the automatic tracking of the user’s position and gaze within the restricted physical domain defined by the PVE (e.g., the walls of the CAVE). Thus the user may move within the PVE only in physically realistic ways. The question of goal-directed navigation then is transferred to the notion of moving the PVE through the environment to be visualized and generating both viewpoints and viewing parameters that enhance the ability to extract knowledge from the presented visual information.

Constrained-Frame Model. Viewpoint changes typically correspond to sliding motion on a parametric surface or more general space that we refer to as the “constraint space” or “guide manifold,” although such characterizations may not always be strictly valid. In one natural model of constrained motion, the coordinate frame of the PVE is closely tied to the guide manifold, e.g., by a sliding constraint that keeps the heads-up direction aligned with the normal to the guide surface. Figure 1 presents schematics of three simple models of how our glass elevator rides on a constraint space, ranging from a 1D motion equivalent to a conventional roller coaster (basically an animation path), to a 2D motion visualizable as a 2D roller coaster, to a 3D environment characterized by an additional 1-parameter “onion skin” family of 2D roller coasters; the various issues introduced by these scenarios will be detailed below.

Gimbal Model. In the above scenario, a natural local surface normal or similar construct determines the “heads-up” direction for the PVE. Another alternative is to release the orientation of the generated view from the constraints of the roller-coaster-like constraint manifold. In this model, the glass elevator’s space may be imagined as suspended on gimbals, as illustrated schematically in Figure 2; the gimbal assembly itself is still rigidly attached to one particular point of the constraint manifold, but the orientation of the PVE may be specified arbitrarily by the designer. Typical applications here would involve cases where a specific orientation, such as the direction of gravity, is natural or comfortable for the viewer; this would simulate a kind of amusement park ride where some combination of gravity and realistic or ad hoc forces specify a natural

“up” direction for the viewer’s enclosing environment.

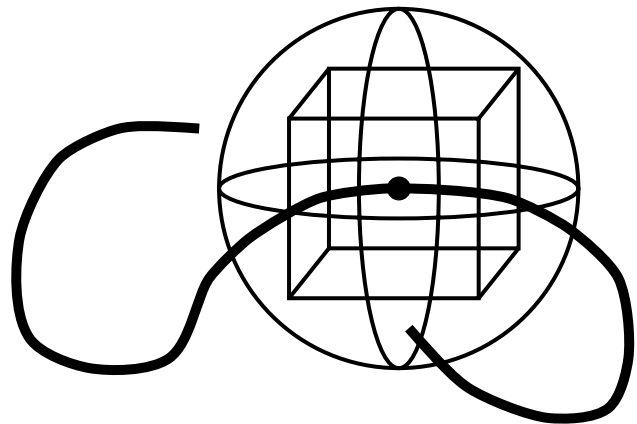


Figure 2: Scenario with glass elevator suspended in a gimbal assembly, allowing arbitrary orientation unrelated to the constraint manifold geometry. Here we show the PVE on gimbals on a 1D roller coaster; the other configurations are easily imagined.

3 Control Methods

As introduced in Hanson and Wernert [7], the fundamental concept of constrained navigation is a map between the parameters of a controller domain and a guide field range that provides a dependent set of parameters to the scene rendering apparatus. Given a bare controller parameter vector \mathbf{u} with optional velocity information $\dot{\mathbf{u}}$, we define a map $\mathbf{G}(\mathbf{u})$ from the domain of the control device to the full space Φ of parameters. For immersive virtual reality devices, the controller domain can include any combination of head-tracker position and hand-held orientation controller position, orientation, and buttons. Although we can imagine very complex mechanisms for using the twelve or more scalar parameters of the controllers of a CAVE-like environment, we will restrict ourselves to a particular style involving at most three simultaneous controller parameters (i.e., taking \mathbf{u} to be at most 3D) because it seems to us to support

most of the elegant applications of the constrained navigation user paradigm.

The controller parameters are used to create the scene presented to the PVE, the space in which the viewer is actually immersed; while the viewing parameters are in principle as flexible for CAVE-like environments as for the desktop, in practice one tends to use parameters creating scenes at the scale of the viewer's body, and the camera focal length and stereopsis parameters are normally adjusted to human scale as well. Thus, except for perhaps an overall scaling that might be used to achieve an "Alice in Wonderland" variation in perceived viewer size, one typically would change only the position and orientation of the PVE's "glass elevator" within the modeled virtual world.

3.1 Basic Navigation Scenarios

2D Modes. For our first set of immersive navigation approaches, we simply work with the user model of a "2D roller coaster," with the following as examples:

- **CAVE floor as mouse pad.** In this algorithm, one navigates as though one were in a very simple 3D desktop environment controlled by a mouse limited to a rectangular range; the only difference is that the head-tracker coordinates are projected to the floor of the user space, and those coordinates are used as virtual "mouse coordinates" to determine the control parameters within a rectangular range. Navigation consists of walking around the user space, looking in any desired direction, but with the head tracker coordinate controlling, e.g., the position and front-wall orientation in a navigation space that is logically rectangular. Unlike the desktop situation, however, one need not change the view parameters to view objects below; one simply looks down at the floor of the CAVE. This method is thus well-suited for certain domains such as simulated aerial terrain observation, and the user has the feeling of being a "walking airplane." Familiar 2D mouse methods such as dragging and rowing can be implemented by "turning on" the motion control only when a controller button is depressed.
- **CAVE wall as mouse pad.** If the desired motions are parallel to the front wall of the PVE, horizontal body motions are potentially (though not necessarily) counterintuitive. An alternate approach uses some physical parameter, such as the wand position or pointing the wand to a point on the front wall, to determine navigation control points. In some applications this mode is much more natural than looking at the CAVE floor past your feet to see objects of interest.
- **Surfing by leaning.** Using a center fixed by default or by a controller button press, this mode measures the horizontal displacement of the head tracker from the center, controlling the direction of sliding on the "2D roller coaster" by leaning in that direction. The same result, with somewhat less kinematic effect, can be achieved by using button-held displacements or rotations of the hand-held controller. Some attention to smoothing small motions seems necessary to avoid jitter. We note that assigning the user a dynamic response, with reduced response at the beginning of a motion, can be used to suppress annoying noise.
- **Golf cart.** This control mode is employed in numerous variations in many existing applications: in our preferred version, the user moves on the 2D constraint surface either forward or backward by pressing a button to start accelerating up to maximum velocity. The path of motion is a circle with radius determined by the direction that the hand-held controller is pointing: forward motion if the controller points at the center of the front screen, large radius if the controller points

slightly to the side, minimum radius if the controller points ninety degrees or more to either side. The head tracker gaze angle projected to the floor can be used alternatively, although this confounds somewhat the freedom of gaze direction that seems to be a major advantage of the immersive compared to the desktop environment.

A third dimension. We started with the restriction to 2D constraints for the spatial navigation routes because these are closest to our normal walking modes for human navigation on the earth's surface. One can add more dimensions to the controller space to make smoothly varying changes in three or more dimensions as well. But in many applications, even those that might seem natural for flying or underwater swimming, a fully continuous third degree of navigational freedom is not always needed to attain the objective of designer-controlled, goal directed navigation. We focus here on a discretely constrained version of the addition of dimensions, thus following through on the idea of the "glass elevator" by restricting the third dimension of navigation to travel between "virtual floors."

The transitions in the third dimension can range from conservatively incremental, as indicated in Figure 1c, to a literal interpretation of jumps between "floors" of a virtual structure or jumps to different worlds. In complex applications such as molecular visualizations, one might even change the scale and topology during the transition between one set of constraints and another; Figure 3 schematically illustrates a discrete 3D constraint space of this type. The idea is that switching to different virtual floors does not necessarily mean a change in actual physical elevation of the viewer's environment; the designer may define discrete but related onion-skin layers that interweave and intersect in complex ways to achieve varying goals. The transition between two layers is algorithmically achieved by keeping the same 2D parameter value, and activating a relatively smooth transition between the viewing parameters stored at those control values in the two neighboring layers.

4 Techniques

Interpolation. Normally, the designer enters a lattice of anchor values for the desired viewpoints; we have implemented this as an array of generalizations of the VRML camera model data field. Controller values are assumed to move smoothly between the anchor points, so intermediate view parameters must be generated by interpolation. We have typically taken the controller parameters to act as spline parameters for Catmull-Rom splines modeling the guide manifold, and have interpolated the remaining viewing field values (orientation, scale, etc.) in a dependent way. This leads to a classic interpolation anomaly, namely that spatial derivatives are smooth while angular derivatives may not have the same level of continuity. This is unavoidable for simple methods, and involves complex additional computations and assumptions to get smoothness in all parameters; the conventional approach is to adjust the angular anchor values if serious anomalies are observed.

Interest Vectors. Various methods may be used to automatically generate certain changes in, say, viewing orientation (the orientation of the glass elevator or PVE in our terminology). One very powerful method that we have investigated uses a scalar field with large values at interesting objects to define a gradient field pointing towards the locations of interest. By taking the cross product of this with an appropriate vector such as the default gaze direction, a rotation can easily be defined that changes the default gaze to a gaze arbitrarily weighted towards the interest point. Appropriate "interest fields" can be easily defined using variants on Blinn's method for implicit surface modeling [2].

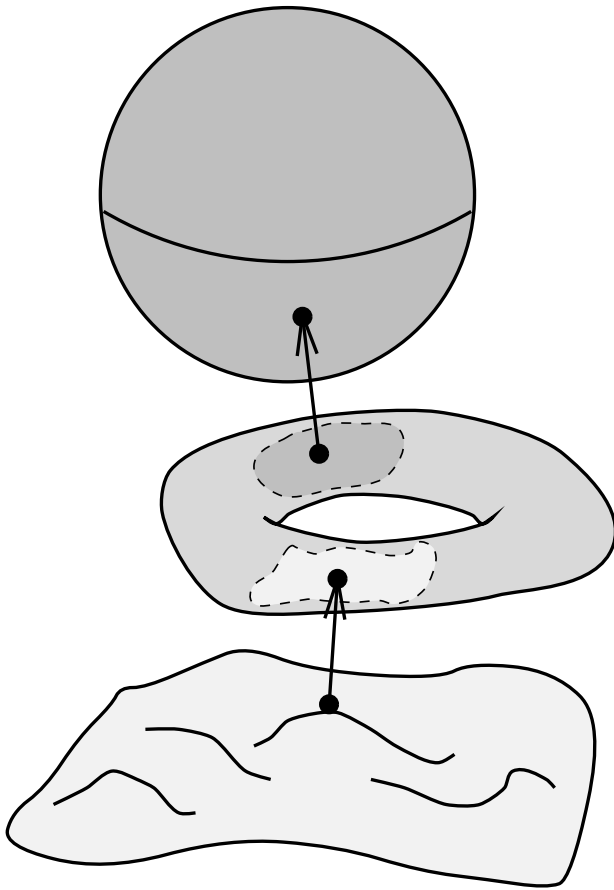


Figure 3: Adding a third controller dimension with a user model of “changing floors” to get to a new 2D navigation space.

5 Examples

We next present several examples of constrained navigation applications for immersive virtual reality environments based on the CAVE.

- Cylindrical Elevator.** To begin with, we consider the problem of inspecting the exterior of a building, in this case a VRML model of the Indiana University Student Building Clock Tower. This particularly simple case might be treated just as easily by moving the object instead of the viewer, but this will not be true in general. Therefore let us suppose first that we need to fly around the building using an airplane-like controller to inspect every facet; clearly, we would have immediate difficulties keeping the building directly in view as we flew by it. By supplying a cylindrical constraint surface with inward-pointing camera parameters at every point, as shown in Figure 4, we can guarantee a reasonable interface; by pointing the wand up, down, left, or right, the user can ride the “glass elevator” up and down the building, or can circle to the left or right, but will always have the front wall oriented directly at the nearest point of the building. Furthermore, by choosing gravity to define the constant heads-up direction, we can preserve relative stability of the user’s perceived environment.
- Platter Stacks.** To illustrate the general idea of a constraint manifold that is three-dimensional, we present in Figure 5 a

family of related 2D constraint surfaces appropriate for different approaches to viewing a terrain model. The PVE frame orientation is typically constrained to align with the surface normal in this application. At the simplest level, we just pass over the terrain model on a flat guide manifold; if we want a more and more intense experience of the bumpiness of the terrain, we can select the more convoluted constraint surfaces. An important point to note is that, while each of these surfaces is on a “different floor” in terms of the logical space, requiring a 3D jump of the controller to get between them, the guide manifolds themselves can be almost coincident; moving a foot in the third controller direction can take you to the same exact coordinate in the environment’s space, but with connectivity to a different set of neighboring camera models. This provides a very powerful potential context-switching ability.

- Toroidal Elevator.** In Figure 6, we present a topologically nontrivial guide manifold, a torus specifically tuned to investigate a large, complex molecule. The viewing parameters are fixed throughout to allow the best local view of the molecular structure while utilizing the externally-fixed direction of gravity to stabilize the heads-up direction. From the top of the torus, the user looks down through the CAVE floor to see the total structure.

6 Preliminary Human Factors Observations

Quicktime movies of animations of the clock tower scene as viewed using constrained navigation are available in <file://ftp.cs.indiana.edu/pub/hanson/tower-grid.mpg> and <file://ftp.cs.indiana.edu/pub/hanson/tower-view.mpg>. The first is an avatar view to set the context, and the second shows the scene from the point of view of the CAVE user.

At the time of this writing, we are just getting our own CAVE installed and are preparing to obtain user feedback on the features of the methods described here. It was therefore not possible to have human factors results for inclusion at this moment, but we will include some preliminary results in the final paper. More extensive user interface studies are being undertaken as student research projects over the coming year.

7 Conclusion

In this paper, we have introduced extensions specific to immersive virtual reality of the one-parameter camera path of a traditional animation. To enhance goal-specific experiences for the user, the designer in this scenario must provide a “virtual roller coaster” constraint manifold along with view-orientation parameters that intelligently evolve the viewing parameters of the “glass elevator” or PVE in which the user is immersed. One must limit the viewer’s freedom of navigation enough to focus attention and prevent loss of context, but not so much as to disturb the feeling of exploration and discovery appropriate to the viewer’s task.

Acknowledgments

AJH gratefully acknowledges the cordial hospitality of Claude Puech and the members of the iMAGIS laboratory, a joint project of CNRS, INRIA, Institut National Polytechnique de Grenoble, and Université Joseph Fourier, where this research was initiated. Thanks are also due to the staff of the Indiana University Computing Services group and to CICA, the Indiana University Center for

Innovative Computer Applications. This research was made possible in part by NSF infrastructure grant CDA 93-03189.

References

- [1] M. Billinghurst and J. Savage. Adding intelligence to the interface. In *Proceedings of VRAIS '96*, pages 168–175, 1996.
- [2] J. F. Blinn. A generalization of algebraic surfaces. *ACM Trans. on Graphics*, 1:235–256, 1982.
- [3] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 135–142, August 1993.
- [4] S. M. Drucker, T. A. Galyean, and D. Zeltzer. Cinema: A system for procedural camera movements. In *Computer Graphics*, pages 67–70, 1992. Proceedings of 1992 Symposium on Interactive 3D Graphics.
- [5] S. E. Goldin and P. W. Thorndyke. Simulating navigation for spatial knowledge acquisition. *Human Factors*, 24(4):457–471, 1982.
- [6] A. J. Hanson and H. Ma. Space walking. In *Proceedings of Visualization '95*, pages 126–133. IEEE Computer Society Press, 1995.
- [7] A. J. Hanson and E. Wernert. Constrained 3d navigation with 2d controllers. In *Proceedings of Visualization '97*. IEEE Computer Society Press, 1997. In press. Also available as Indiana University Computer Science Department Technical Report 479, and by anonymous ftp from ftp.cs.indiana.edu, pub/hanson/vis97.pdf; corresponding animations may be found in the files chemgrid.mpg and chemview.mpg.
- [8] J. D. Mackinlay, S. Card, and G. Robertson. Rapid controlled movement through a virtual 3d workspace. In *Computer Graphics*, volume 24, pages 171–176, 1990. Proceedings of SIGGRAPH 1990.
- [9] C. B. Phillips, N. I. Badler, and J. Granieri. Automatic viewing control for 3d direct manipulation. In *Computer Graphics*, pages 71–74, 1992. Proceedings of 1992 Symposium on Interactive 3D Graphics.
- [10] W. Robinett and R. Holloway. Implementation of flying, scaling, and grabbing in virtual worlds. In *Computer Graphics*, pages 189–192, 1992. Proceedings of 1992 Symposium on Interactive 3D Graphics.
- [11] P. W. Thorndyke and S. E. Goldin. Spatial learning and reasoning skill. In H.L. Pick and L.P. Acredolo, editors, *Spatial Orientation: Theory, Research, and Application*, pages 195–217. Plenum Press, New York, 1983.
- [12] P. W. Thorndyke and B. Hayes-Roth. Differences in spatial knowledge acquired from maps and navigation. *Cognitive Psychology*, 14:560–589, 1982.
- [13] C. Ware and S. Osborne. Exploration and virtual camera control in virtual three-dimensional environments. In *Computer Graphics*, volume 24, pages 175–184, 1990. Proceedings of 1990 Symposium on Interactive 3D Graphics.

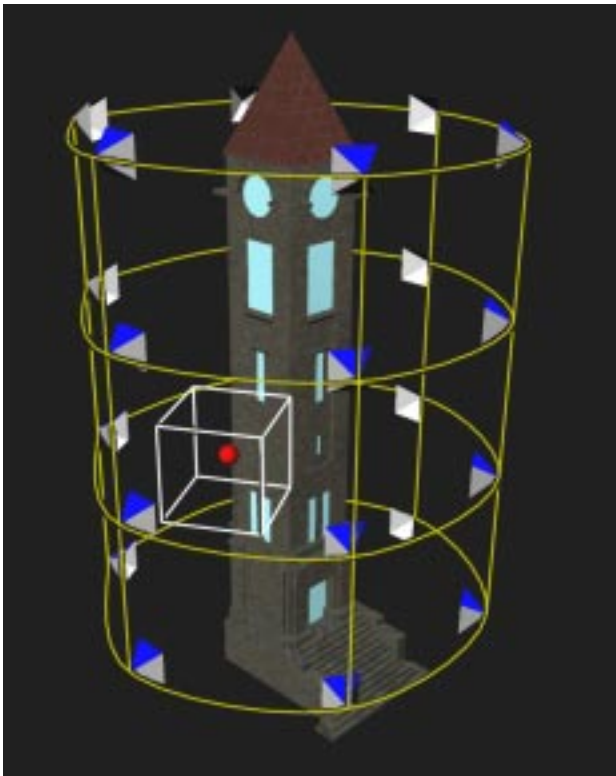


Figure 4: A clock tower and a cylindrical guide manifold designed to inspect it from all angles using a simple 2D plane of navigation parameters. The PVE is shown as a white box, with the user's head symbolized as a red globe.

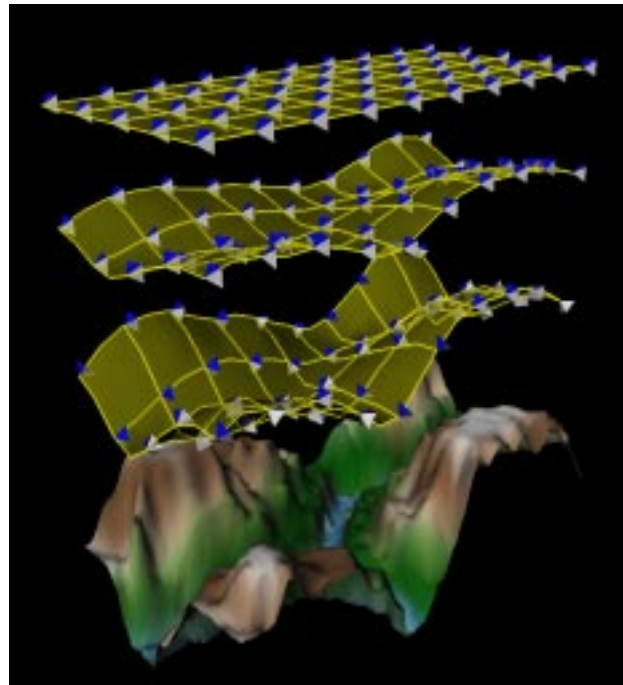
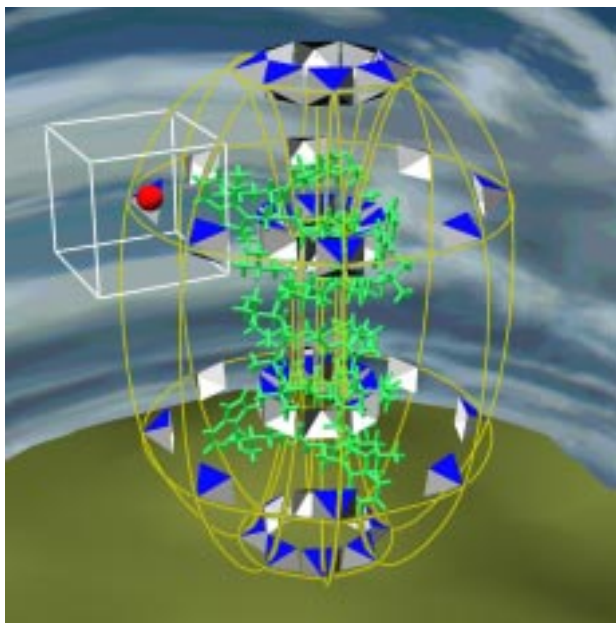
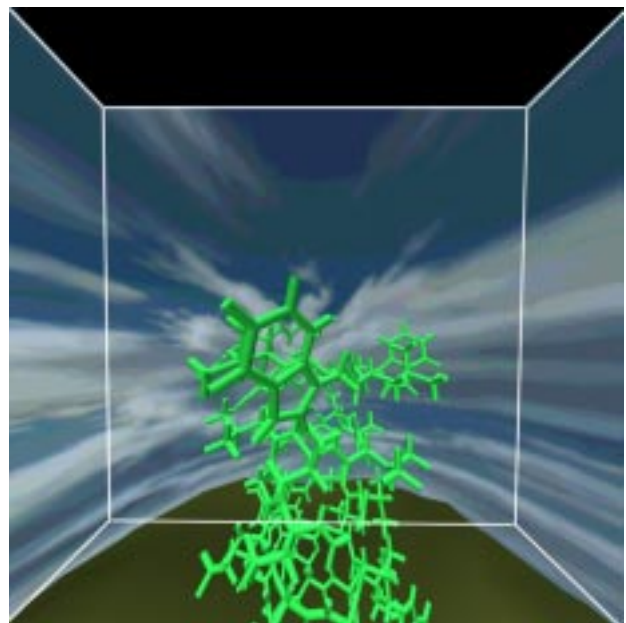


Figure 5: A terrain model with a family of related 2D constraint surfaces; these implicitly form a 3D manifold that can be accessed by a 3D motion of the controller. Also note that, in principle, this could be an exploded schematic of set of 2D surfaces that coincided at many points in 3D space instead of being separated as they are here for clarity.



(a)



(b)

Figure 6: Exploring a large molecule using the constrained navigation paradigm. (a) Overview of the guide manifold, sample camera models, and a representative position of the PVE shown as a white box. (b) The scene as viewed from inside the immersive environment.