

# Constrained Navigation Environments

Andrew J. Hanson      Eric A. Wernert  
Stephen B. Hughes  
Computer Science Department  
Indiana University  
Bloomington, IN 47405 USA

hanson@cs.indiana.edu, ewernert@cs.indiana.edu, shughes@mail.sis.pitt.edu

## Abstract

*Finding one's way through a complex virtual environment is a standard task in 3D graphics and virtual reality applications. Constrained navigation is a method that appropriately restricts the user's degrees of freedom when there is a poor match between the goal of an exploration activity, the control device, and the user's familiarity with the exploration domain. The fundamental prerequisite for the adoption of constrained navigation is that the designer can significantly improve the quality of the user's experience by choosing a predetermined parametric set of viewing parameters or algorithms. We discuss families of constrained navigation methods appropriate to desktop and immersive virtual reality applications. We illustrate the approach with a variety of examples, emphasizing the possibility of topologically nontrivial navigation spaces, and present the results of a preliminary user study.*

## 1. Introduction

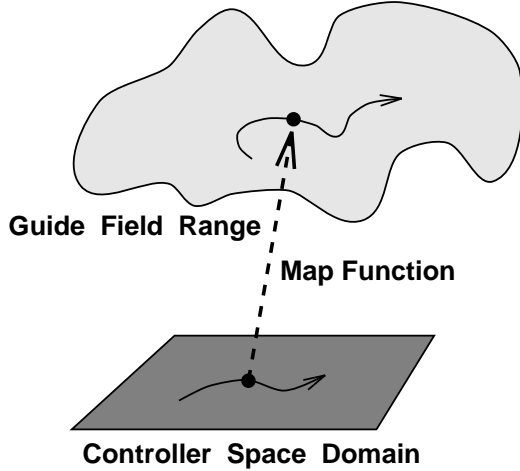
We define “constrained navigation” for the purposes of this treatment as the restriction of viewpoint generation and rendering parameters to goal-driven subclasses whose access is specified by the application designer. We will focus for definiteness on navigation through large-scale spaces, so the user's immediate environment appears to be executing a movement in the overall scene; we will not concern ourselves with applications in which small objects of interest are manipulated within the user's grasp.

In this article, we summarize the desktop-oriented two-degree-of-freedom controller methods for constrained navigation introduced in Hanson and Wernert [8], extend the concepts to immersive virtual reality (VR) environments, and present some initial user interface results.

The need for constrained navigation is particularly critical in immersive 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 impression concerned the difficulty of navigation rather than the subject of the display.

Ordinary computer viewpoint animation is in fact a species of constrained navigation consisting of a linear time sequence of camera models. Here we generalize this standard method to 2D or even 3D parameter spaces to generate viewpoints in 3D desktop and VR applications. We extend and contrast previous work on geometry-driven space navigation [7] and desktop constrained navigation [8] to handle VR environment issues. Early important work on intelligent navigation upon which we build includes that of Mackinlay et al. [9] and Phillips et al. [10], as well as motion control systems like those of Ware and Osborne [16], Drucker et al. [5], and Robinett and Holloway [11]. The addition of intelligent interfaces is illustrated by the work of Billinghurst and Savage [2]. The acquisition of cognitive maps, a special and limited application of constrained navigation, has attracted a great deal of attention as well, as described in the work of Thorndyke, Goldin, and Hayes-Roth [6, 14, 13]; 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.



**Figure 1. Diagram of the general mathematical concept of a guide field and its ramifications.**

## 2. Fundamental Methods.

The basic idea behind our approach is the concept of mapping a controller domain into a guide field range consisting of the parameters needed to construct the scene image, possibly combined with scene-specific data and other parameters modifying the influence of the controller. This is represented schematically in Figure 1. We begin with a bare controller position  $\mathbf{u}$ , assuming the implicit availability of heading and velocity information  $\dot{\mathbf{u}}$ , and define a map  $\mathbf{G}(\mathbf{u}, \dot{\mathbf{u}})$  from the domain of the control device to the full space  $\Phi$  of parameters. In principle the range of the parameter space can include anything, even computed quantities. Thus we write

$$\mathbf{G} : (\mathbf{u}, \dot{\mathbf{u}}) \mapsto \Phi, \quad (1)$$

where objects in the range  $\Phi$  include such things as

1. Camera position on guide manifold: the point in the universe where the virtual owner of the device appears to be standing.
2. Camera orientation: where the virtual user is looking.
3. Camera properties: parameters such as focal length (wide angle, telephoto lens), depth of field, and binocular convergence.
4. Viewing properties: fog, light attenuation, etc.
5. Control modifiers: controller response, importance weighting, gradients of the data fields, etc.
6. Visualization application parameters: streamline characteristics, particle source location, pseudo-color assignments, etc.

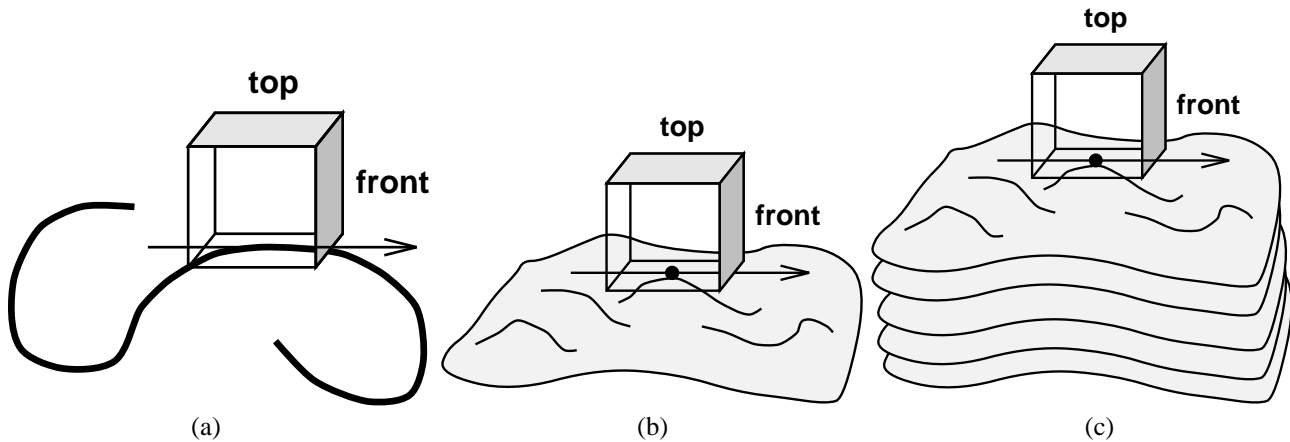
By retaining successive values of these fields in the control program, the designer can also create rate-of-change-dependent responses.

For most practical purposes, the controller domain corresponds locally to a path in the guide manifold that is equivalent to a surface in the 3D world. However, one can imagine applications in which more general mappings might be useful. For example, one might instead use the controller position to vary a two-parameter camera orientation  $(\theta, \phi)$ , treat this orientation as the independent variable of the guide manifold, and treat spatial position as a dependent guide field variable attached to each point of  $(\theta, \phi)$  in the guide manifold. Therefore, we retain all the scene-viewing parameters in a single data structure, and specify local patches with coordinate vertices in that parameter space that correspond to controller position. Each value of the independent controller variables then selects a particular set of parameters (e.g., one camera position and an orientation out of the space of possible viewing angles at that position). These dependent variables are typically determined by selecting samples on a lattice in the control space, and thus we must interpolate all these variables in tandem. Note that achieving smoothness in all variables may be nontrivial.

## 3. User Model for Constrained VR Navigation

The fundamental VR user model that we favor 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 CAVE™ [4]. In order to extend this user 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.

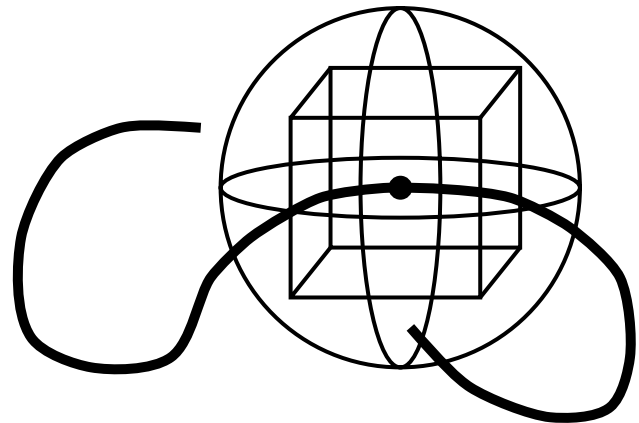
**Constrained-Frame Model** Viewpoint changes typically correspond to sliding motion on a parametric surface or more general space that we refer to as the “constraint space” or “guide manifold,” although such characterizations may not always be strictly valid. In one natural model of constrained motion, the coordinate frame of the PVE is closely tied to the guide manifold, e.g., by a sliding constraint that



**Figure 2. 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.**

keeps the heads-up direction aligned with the normal to the guide surface. Figure 2 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 3; 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 3. 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.**

#### 4. Control Methods Appropriate for VR

Given a bare controller parameter vector  $\mathbf{u}$  with optional velocity information  $\dot{\mathbf{u}}$ , we define a map  $\mathbf{G}(\mathbf{u}, \dot{\mathbf{u}})$  from the domain of the control device to the full space  $\Phi$  of parameters. For immersive virtual reality devices, the controller domain can include any combination of head-tracker po-

sition 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.

**2D Navigation 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

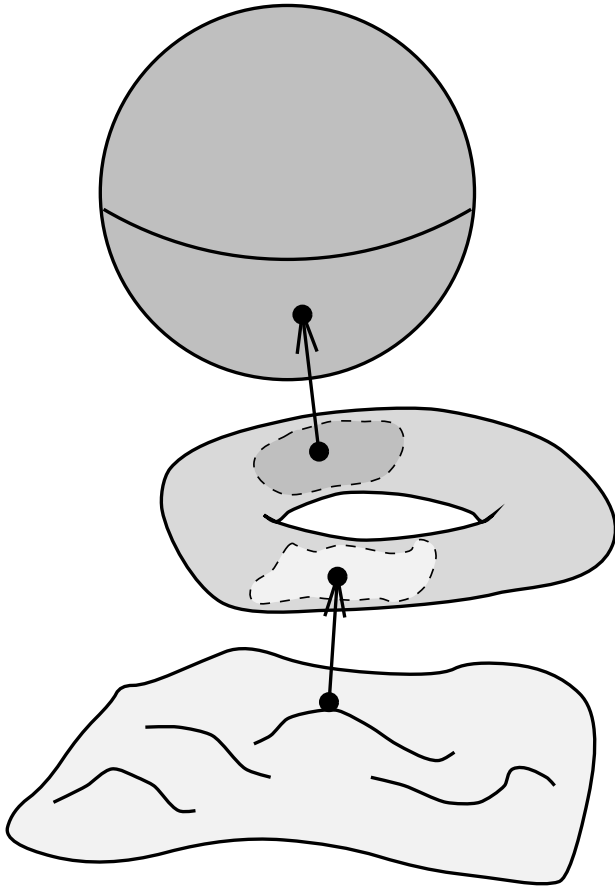
wand to a point on the front wall, to determine navigation control points. In some applications this mode is much more natural than looking at the CAVE floor past your feet to see objects of interest.

- **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.

**3D Navigation Modes.** 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 2c, 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 4 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.



**Figure 4. Adding a third controller dimension with a user model of “changing floors” to get to a new 2D navigation space.**

## 5. Implementation 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

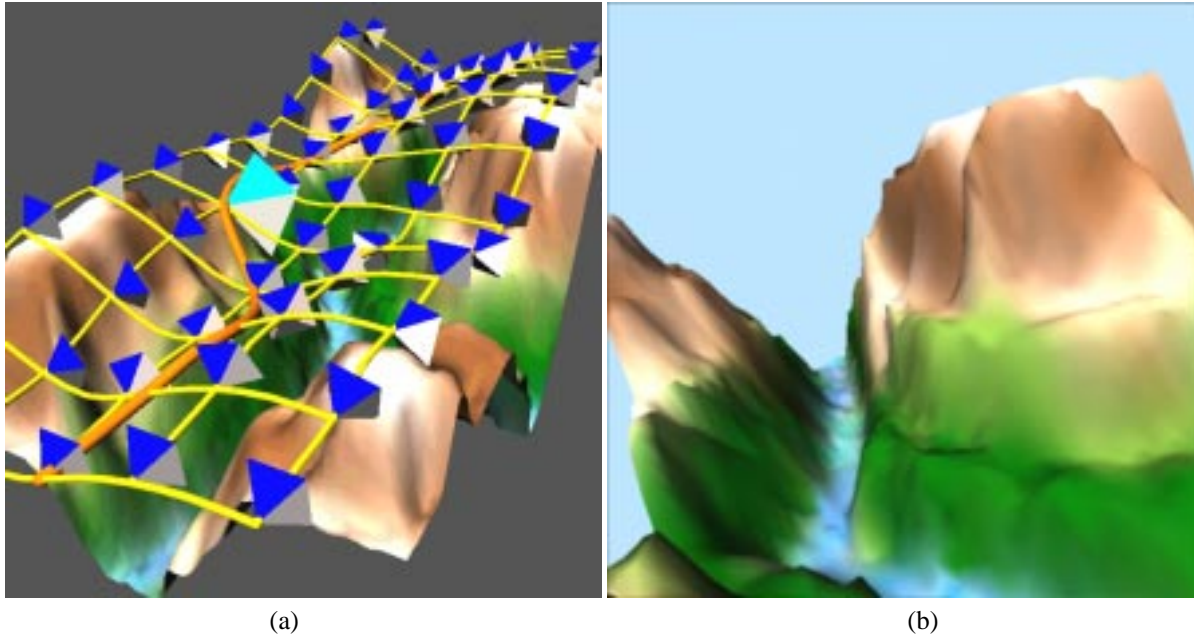
to move smoothly between the anchor points, so intermediate view parameters must be generated by interpolation. We have typically taken the controller parameters to act as spline parameters for Catmull-Rom splines modeling the guide manifold, and have interpolated the remaining viewing field values (orientation, 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 [3]. A separate interest field can in principle be supplied for each parameter, allowing, e.g., the camera focal length, to be varied independently throughout the navigation.

**Modulation by Data.** We can immediately go beyond the already useful idea of having predetermined camera parameters at each point of the navigable space by defining *modifiers* of the default parameters. In Figure 5, we show the result of using the gradient  $\nabla\phi$  of the terrain elevation model as a cue: starting with an “up” direction aligned with the surface normal, we rotate the camera by a weighted amount to turn gently towards the gradient into the valley.

An explicit example is the following: at each point of the coordinate-space guide manifold, determine the “heads up” direction of the camera frame  $\hat{u}$ , the “look at” direction of the camera frame  $\hat{k}$ , and projection  $\hat{p}$  of the terrain gradient onto the plane perpendicular to  $\hat{u}$ ; then, if  $\cos\phi = \hat{p} \cdot \hat{k}$  describes the angle between the projected terrain gradient and the camera gaze direction, one rotates the camera about the  $\hat{u}$  vector by  $c\phi$ , where  $c = \|\hat{p}\|/\|\hat{p}_{\max}\|$  is the relative magnitude of the projected gradient strength.

**Sensitivity Fields.** A number of applications have identifiable areas where one wants to have very fine control, and others where one wants coarse control for quickly traversing large, uninteresting areas. We note two examples that



**Figure 5. Camera path constrained to complex surface with camera orientation keyed to constraint surface normal and modulated by terrain gradient. (a) View of path and camera model control points on constraint surface. (b) View using camera model field at selected point.**

fit cleanly into our framework: (1) Velocity-based displacement. Several common mouse interfaces have long supported this feature: the velocity of the mouse is measured, and as the speed increases, the overall displacement is amplified accordingly, allowing quick navigation to all corners of the screen. (2) Response field. Here, we just define a scalar field over the guide manifold and use it to magnify or reduce the bare controller displacement at each local point. Effects such as those of Mackinlay [9], could be achieved without the use of scale factors simply by refining the mesh near the critical points of the guide manifold. However, it is awkward to make the changes occur smoothly in such a mesh, and the continuous scale change field overcomes this.

## 6 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 gen-

eral. 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 8, 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 10 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 con-

straint 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 11, 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.

## 7. Designer Techniques

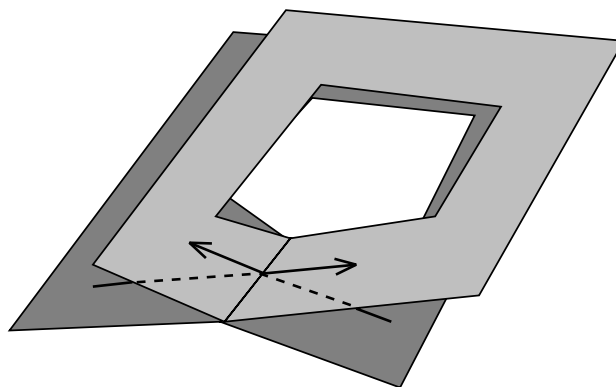
There are a variety of techniques that we have found useful in practice to enhance the utility, visual immediacy, and flexibility of the constrained navigation framework. Among these we note especially the following:

**Fog, Spotlights, etc.** The actual scene appearance can equally well be modulated to suit the designer’s needs. We suggest the following methods: (1) Fog. As one passes through a scene, one can limit the visibility to a handful of key regions by obscuring the most distant objects. Other application-dependent depth cues can be used if appropriate. (2) Spotlights. Whether or not the camera model allows you to change its gaze, you can shine a spotlight on any desired sector to emphasize it. This is very easy in OpenGL, requiring only the definition of a few key-frame values of a direction. The spotlight need not be large, nor coincide with the gaze or motion directions.

**Vista Points.** A fundamental context-defining technique available in such a navigation system is the “scenic overlook.” This is very much like an overlook on a vacation highway, except that the signposts and annotated vista points can be placed anywhere in 3D space continuously connected to the sidewalk. As the viewer approaches the critical vista point itself, changes in the focal length, camera orientation, and control response can be imposed by the designer to exactly emulate features such as Mackinlay et al.’s [9] controlled approach, or even “dynamic field glasses” that focus in on distant scene features as though one had

donned zoomable binoculars to pan across the scene of interest, similar to one scenario of Robinett and Holloway [11].

**Multiple Coverings.** Another fundamental technique is the “multiple covering” navigation surface. (Readers with mathematical backgrounds will recognize this as a relative of Riemann surfaces in complex variable theory.) Here, one creates a surface that may come back to the same point by many different routes; a simple example is a double ribbon, as shown in Figure 6, which allows the camera to point in one family of directions the first time around the ribbon, in other directions the second time, and to return to the original state the third time around. An explicit application is depicted in Figure 9. The reader can imagine arbitrarily complex variants, including instantaneous state transitions between entirely different guide fields.



**Figure 6. An example of a navigation manifold that contains more than one possible layer, hence more than one possible camera model, depending on one’s route to the scene.**

## 8. Preliminary Human Factors Observations

Users of virtual environments typically hope to obtain two types of information: the presence of particular objects and the organizational relationship between those objects. These requirements fit into the spatial knowledge framework established by Siegel and White [12]. The ability to extract and remember features from the scene defines landmark knowledge. Survey knowledge refers to understanding the global configuration of objects. The constrained system appears naturally to facilitate development of both types of spatial knowledge.

Extensive human factors studies of the comparative effectiveness of particular scenarios are beyond the intended

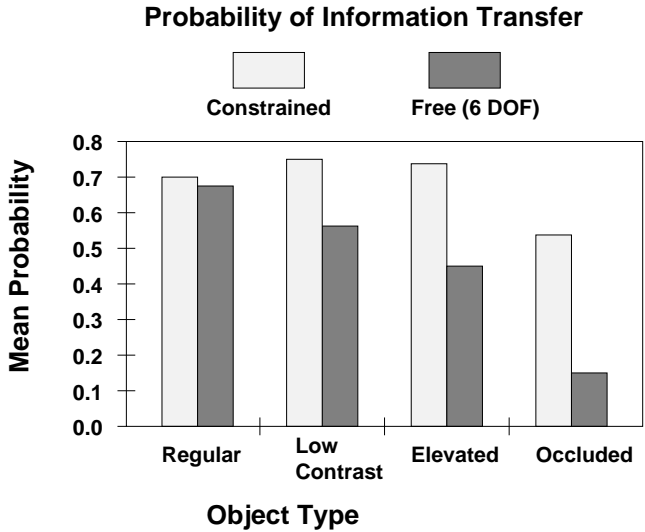
scope of this paper. Nevertheless, an initial evaluation of alternative exploration modes has been carried out for room-like worlds, using criteria inspired by those of Thorndyke et al. [15, 14, 13]. Specifically, this study attempted to validate the hypothesis that the use of interest vectors increases the landmark knowledge of a 3D environment.

Twenty undergraduate subjects, half male and half female, were asked to navigate several rooms using either a constrained system with a 2D controller or an unconstrained six-degree-of-freedom (6DOF) spaceball. Upon completion of the task, they were asked to identify the objects they had seen in the environment. To measure information transfer, we recorded the number of objects the person was near (for this study, a person was near an object if they were in the same room as the object). We then calculated the ratio of nearby objects to the number of objects correctly recalled. The ratio was then weighted by the subject’s bias/sensitivity (note: subject sensitivity was statistically not affected by the control paradigm).

The data were examined to determine the likelihood of information transfer over four categories of objects: elevated, occluded, low-contrast and plain. An object was said to be elevated if it was not viewable unless the subjects either elevated themselves (+Y) or looked up (+Pitch). An object was defined as occluded if it was obstructed from view in more than four directions; such objects could therefore only be viewed from a particular vantage point. A low-contrast object was characterized as having the same ambient color as the background; the viewer typically had to approach such an object quite closely in order to discriminate it from the background. Finally, a plain object was defined as any object that was not elevated, occluded, or low-contrast. Plain objects could be viewed readily from multiple perspectives.

As shown in Figure 7, plain objects were recognized equally well by the constrained and free techniques, with  $F(1, 18) = .19, p < .67$ . The constrained techniques were significantly better at discerning low contrast objects, with  $F(1, 18) = 4.42, p < .05$ . The constrained techniques were far superior at revealing elevated and occluded objects, with  $F(1, 18) = 16.87, p < .0008$  and  $F(1, 18) = 20.33, p < .0003$ , respectively.

These preliminary findings suggest that the use of a constrained system results in increased landmark knowledge over a free system. The reason for this improvement can be attributed to additional the guidance provided by the constrained system. By fixing the subjects’ gaze on a particular object, or by imposing large involuntary sweeping changes in the field of view, we were providing extra cognitive cues. Moreover, a constraint field can be designed so that viewers have no choice but to find objects they may not have thought to look for, while users of the free system are able to walk right past the object, oblivious to its existence.



**Figure 7. Results of preliminary user study, showing probability that different classes of objects in a multi-room environment are recalled by users provided with constrained navigation tools (left bar) vs full 6 degree-of-freedom spaceball navigation tools (right bar).**

Although empirical evidence is still pending, we can anticipate an increase in survey knowledge as well. Traditional 3D navigation techniques burden the user with the task of manipulating a controller with six degrees of freedom. This detracts from the user’s spatial memory and causes disorientation (see, e.g., Arthur et al., [1]). However, with a constrained system, the user’s burden is typically reduced. Navigators should thus be able to focus more on the environment instead of the controls, expediting survey knowledge development.

The conclusion is that, for the object retention task in an environment consisting of connected rooms, retention ratios of hard-to-notice objects in a room range from 55% to 75% for users of the constrained system, compared to 10% to 35% for users given 6 degree-of-freedom navigation controls. Thus the concept of using a constrained system to focus on a user goal seems well-founded.

## 9. Conclusion

In this paper, we have introduced an extension of the one-parameter camera path of a traditional animation to a multiparameter space appropriate for constrained navigation in both 3D desktop and immersive virtual reality environments. The basic strategy is to supply a set of view-determining data at each sample point of a “virtual side-walk,” along with possibly state-dependent procedures to

create the actual view to be presented. To enhance goal-specific experiences for the VR 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.

Future plans include extensions to more complex virtual reality environments and controllers, additional human factors testing, and experimentation with “smart” controls that balance constraints against the user’s state and goals.

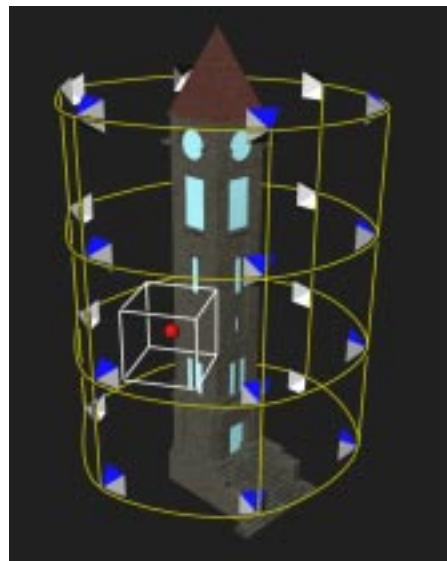
## Acknowledgments

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.

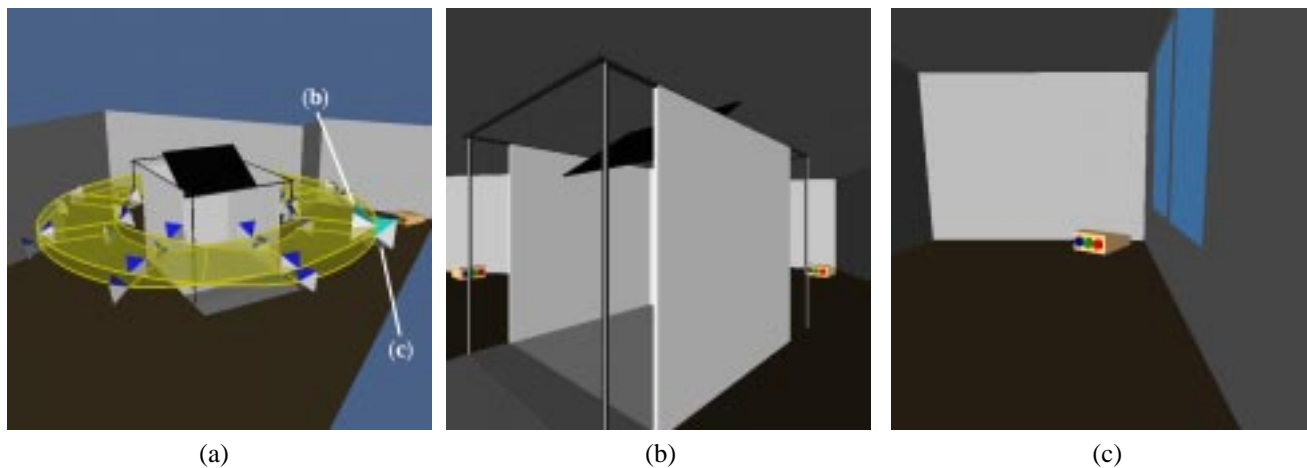
## References

- [1] E. Arthur, P. Hancock, and S. Chrysler. Spatial orientation in real and virtual worlds. In *Proceedings of the Human Factors and Ergonomics Society*, 1993.
- [2] M. Billinghurst and J. Savage. Adding intelligence to the interface. In *Proceedings of VRAIS '96*, pages 168–175, 1996.
- [3] J. F. Blinn. A generalization of algebraic surfaces. *ACM Trans. on Graphics*, 1:235–256, 1982.
- [4] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In J. T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 135–142, Aug. 1993.
- [5] 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.
- [6] S. E. Goldin and P. W. Thorndyke. Simulating navigation for spatial knowledge acquisition. *Human Factors*, 24(4):457–471, 1982.
- [7] A. J. Hanson and H. Ma. Space walking. In *Proceedings of Visualization '95*, pages 126–133. IEEE Computer Society Press, 1995.
- [8] A. J. Hanson and E. A. Wernert. Constrained 3d navigation with 2d controllers. In *Proceedings of Visualization '97*, pages 175–182. IEEE Computer Society Press, 1997.

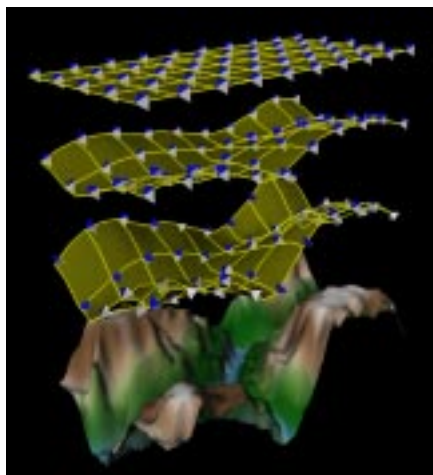
- [9] J. D. Mackinlay, S. Card, and G. Robertson. Rapid controlled movement through a virtual 3d workspace. In *Computer Graphics*, volume 24, pages 171–176, 1990. Proceedings of SIGGRAPH 1990.
- [10] C. B. Phillips, N. I. Badler, and J. Granieri. Automatic viewing control for 3d direct manipulation. In *Computer Graphics*, pages 71–74, 1992. Proceedings of 1992 Symposium on Interactive 3D Graphics.
- [11] 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.
- [12] A. W. Siegel and S. White. The development of spatial representations of large-scale environments. In H. Reese, editor, *Advances in child development and behavior, Vol. 10*. Academic Press, 1975.
- [13] P. W. Thorndyke and S. E. Goldin. Spatial learning and reasoning skill. In H. Pick and L. Acredolo, editors, *Spatial Orientation: Theory, Research, and Application*, pages 195–217. Plenum Press, New York, 1983.
- [14] P. W. Thorndyke and B. Hayes-Roth. Differences in spatial knowledge acquired from maps and navigation. *Cognitive Psychology*, 14:560–589, 1982.
- [15] P. W. Thorndyke and C. Stasz. Individual differences in procedures for knowledge acquisition from maps. *Cognitive Psychology*, 12:137–175, 1980.
- [16] 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.



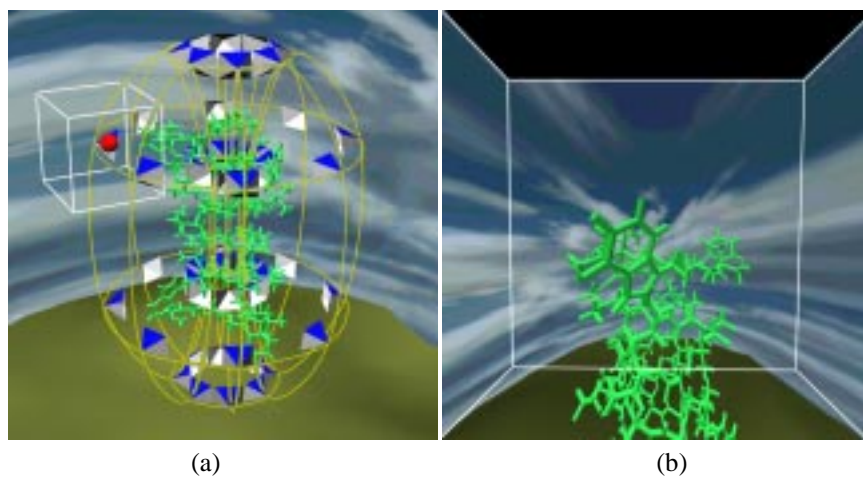
**Figure 8. A clock tower and a cylindrical guide manifold designed to inspect it from all angles using a simple 2D plane of navigation parameters. The PVE is the white box, and the user’s head is the small globe inside the box.**



**Figure 9. (a) Example of a multiple-valued constraint configuration. (b) View from marked point first time around the path. (c) View from marked point second time around the path, showing a different detail to the viewer.**



**Figure 10. A terrain model with a family of related 2D constraint surfaces; these implicitly form a 3D manifold accessed by a 3D controller motion.**



**Figure 11. 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.**