Constrained 3D Navigation with 2D Controllers

Andrew J. Hanson Eric A. Wernert

Computer Science Department
Indiana University
Bloomington, IN 47405 USA

Abstract

Navigation through 3D spaces is required in many interactive graphics and virtual reality applications. We consider the subclass of situations in which a 2D device such as a mouse controls smooth movements among viewpoints for a "through the screen" display of a 3D world. Frequently, there is a poor match between the *goal* of such a navigation activity, the control device, and the skills of the average user. We propose a unified mathematical framework for incorporating context-dependent constraints into the generalized viewpoint generation problem. These designer-supplied constraint modes provide a middle ground between the triviality of a single camera animation path and the confusing excess freedom of common unconstrained control paradigms. We illustrate the approach with a variety of examples, including terrain models, interior architectural spaces, and complex molecules.

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]: Applications.

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

1 Introduction

Navigation in 3D scenes, which we define as the process of selecting a continuously-changing set of viewing parameters, is a long-standing challenge for computer graphics and visualization applications. Computer animation, for example, requires the choice of a time sequence of camera models that can be considered as a one-parameter constraint; applicable techniques range from direct orientation interpolation (e.g., [18, 11]) to rule-based systems [9, 10]. The more complex task of interactive navigation has been considered in a wide variety of contexts, ranging from the viewing of simple 3D scenes on a desktop monitor to the control of fully immersive virtual reality environments. Examples of such viewing control methods run the gamut from orientation control paradigms (Brooks [4], Nielson and Olson [14], Chen et al. [5], Hanson [7], and Shoemake [20, 21]) to methods that intelligently focus on particular scene points such as Mackinlay et al. [12], constraint-based camera placement systems such as Phillips et al. [15], and general control systems such as those discussed by Ware and Osborne [25] and Drucker et al. [6]. The use of constraints in view selection specifically for virtual reality has been used, for example, by Robinett and Holloway [16] to go beyond the usual "flying" modality, and by Billinghurst and Savage [2] in an expert system context.

In this paper, we focus on the problem of using a 2 degree-of-freedom controller such as a mouse to move effectively through a displayed 3D environment with a particular task in mind; we assume that the system designer has at least some idea of how in fact to direct a naive user's attention to those aspects of the scene needed to meet a chosen goal. We present some very specific families of techniques that may be used by the designer to constrain the user's motion in ways that avoid the "lost-in-space" pitfalls of most airplane-style or helicopter-style controls with up to 6 (or more) degrees of freedom. Our fundamental notion is that, rather than controlling an unconstrained vehicle in 3D space, the 2D control device is actually moving the user on a constrained subspace, the "guide manifold," a kind of virtual 2D sidewalk. At every sample point of this virtual sidewalk, we may specify a "guide field" containing all the information the designer wishes to supply to a customizable algorithm computing the viewing parameters for the user. Typically, both the guide manifold and the guide fields are specified only at sample points, and interpolation methods are used to determine intermediate values. The manifold itself may be continuous, may consist of disjoint pieces, or may even cross over itself to give it "Riemann-manifold" properties that let the traveler traverse a circuit over and over to the same spot, and each time be presented with a new set of guide parameters. The parameters of the guide field may supply arbitrarily complex information to the designer's algorithm; we illustrate the power of the idea using applications to terrain navigation, architectural structures, and complex molecules. An evaluation of several basic features of the paradigm is currently in progress.

Combining Displacement Constraints and Viewing Constraints. There are several effective ways to construct a framework for constraint-based navigation in 3D viewing situations. In the simplest version, we just extend the one-parameter camera path of a traditional animation to a two-parameter surface in 3D space navigated by mouse strokes; each point of the surface incorporates a fixed camera-model field. In many cases, the data themselves provide a context of interest, and can thus be used to modulate a fixed viewing-parameter field relative to the source of interest.

The field variables may be fixed *a priori* at key vertices using designer-specified camera models (orientation plus focal length) and interpolated among key vertices; or the field variables may be computed from procedures combining fixed fields, dynamic or static scene data, and current viewer position and state (e.g., velocity). It then becomes the designer's problem, not the viewer's, to minimize the "lost in space" effect, and thus to optimize the viewer's ability to focus on the task that is the goal of the navigation.

A related example of such a system was introduced for the exploration of complex mathematical manifolds in Hanson and Ma [8]. The key constraint in this original concept was the idea that every direction on a 2D manifold implies a geodesic path determined by the intrinsic geometry; the manifold itself provides a constraint on the navigation by providing a "platform" on which the user walks and which continually rolls up to meet the viewer's feet, keeping a constant relative orientation between the viewer's vertical and the surface normal. This path automatically determines an orientation in response to directional changes of the 2D mouse control. The more general concepts proposed in the current paper follow from the realization that the manifold on which the viewer is "walking" could in fact be an *invisible sidewalk* created for the

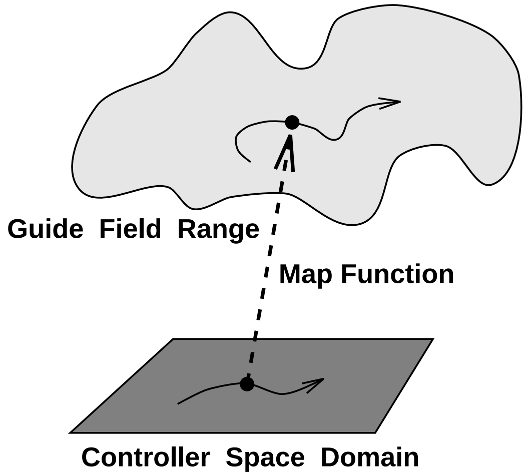


Figure 1: Diagram of the general mathematical concept of a guide field and its ramifications.

purpose of seeing *other* things in the surrounding world, and that the geodesic-constrained orientations can easily be replaced by a completely arbitrary field of quaternion orientations combined with a tandem field of focal lengths and additional viewing and control parameters if appropriate.

Below, we propose several additional families of dynamic procedures for determining the current camera parameters in addition to fixed key vertex values and the geodesic interpolation methods of Hanson and Ma [8]; these range from methods based on metric relations between the navigation surface and the nearby scene or terrain, to methods that could be based on arbitrary rules in the manner of Karp and Feiner, or Billinghurst and Savage [9, 10, 2]. While we focus here on 2D mouse-based interfaces, the framework clearly extends to immersive virtual reality environments, where the virtual space of the control device can select points and orientations in a 3D volume, instead of simple 2D mouse coordinates. We defer exploration of such issues for the time being in order to focus here on fundamental concepts of direct application to the most common visualization systems.

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 parameters modifying the influence of the controller. This is represented schematically in Figure 1. We begin with a bare controller position $(u,\, v)$, assuming the implicit availability of heading and velocity information $(\dot{u},\, \dot{v})$, and define a map $\mathbf{G}(u, v)$ from the domain of the control device to the full space Φ of parameters. In principle the range of the parameter space can include anything, even computed quantities. Thus we write

$$\mathbf{G} : (u, v) \mapsto \Phi \ , \tag{1}$$

where objects in the range Φ 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: mouse response, importance weighting, 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 mouse position to vary a two-parameter camera orientation (θ, ϕ), 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 (θ, ϕ) in the guide manifold. Therefore, we retain all the scene-viewing parameters in a single data structure, and specify local 2D patches with coordinate vertices in that parameter space that correspond to 2D 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. Achieving smoothness in all variables is problematic, but can be addressed in various ways discussed below.

Winged Patches. The simplest relation mapping the controller space to the scene viewing parameters is generated by a single rectangular patch in one-to-one correspondence with the 2D mouse position, as shown in Figure 2a. To create navigable manifolds in more complex situations, we must sew together many of these fundamental pieces to form a connected whole. The simplest practical way to achieve this is to require that the edge shared by two adjacent patches be "winged:" that is, the curve representing the edge must contain pointers to the rectangular patches that share it, allowing a navigation algorithm to detect the end of one patch and implement a transition to the next patch. Figure 2b illustrates a typical structure that can be represented in this way; many interesting topological objects one might wish to represent, such as a sphere, require two or more such patches (for further details, consult any elementary text on differentiable topological manifolds). There are many ways one might handle winged patches in practice, and such

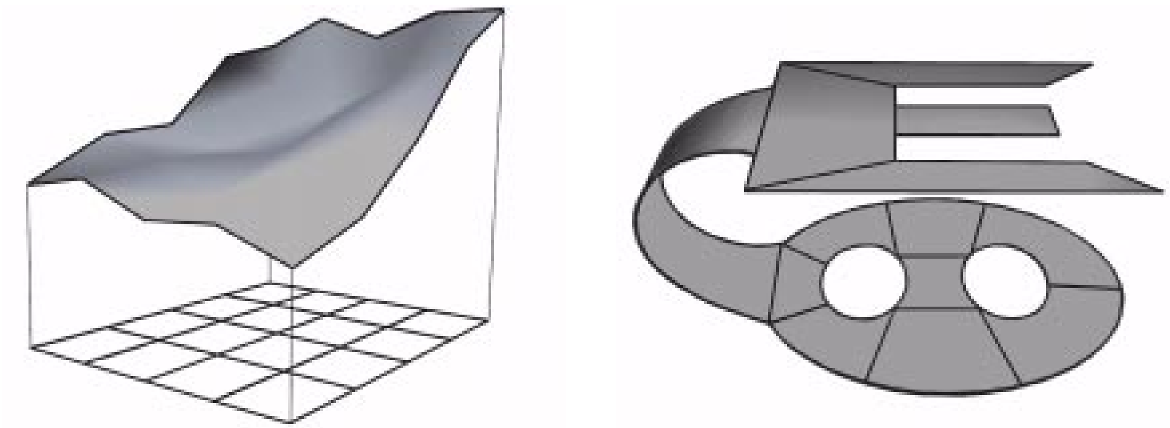


Figure 2: (a) A rectangular patch in mouse space (below), lifted to a guide surface in 3D (above). (b) A network of rectangular guide patches pieced together into a generalized guide surface using winged edges to relate one patch to another.

issues as continuity and differentiability across the transition edges are open to the designer; in some cases a smooth transition, achievable using spline techniques, may be essential, and in other cases a transition with a discontinuous derivative may create the desired effect.

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 7, 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 $\hat{\mathbf{u}}$, the "look at" direction of the camera frame $\hat{\mathbf{k}}$, and projection $\hat{\mathbf{p}}$ of the terrain gradient onto the plane perpendicular to $\hat{\mathbf{u}}$; then, if $\cos\phi = \hat{\mathbf{p}} \cdot \hat{\mathbf{k}}$ describes the angle between the projected terrain gradient and the camera gaze direction, one rotates the camera about the $\hat{\mathbf{u}}$ vector by $c\phi$, where $c = \|\hat{\mathbf{p}}\|/\|\hat{\mathbf{p}}\mathrm{max}\|$ is the relative magnitude of the projected gradient strength.

Interest Vectors. Interest vectors are a generalization of the data modulation method of the previous paragraph. When the viewer is positioned at any point in a particular scene, the designer may record both viewer information, such as gaze direction $\mathbf{g}$, and a direction of interest $\mathbf{d}$ in the scene appropriate to the current viewer state. These typically provide sufficient information to specify a context-based, weightable state change for the camera model. A typical example would compute the plane containing $\mathbf{g}$ and $\mathbf{d}$ and rotate about the direction normal to that plane, $\mathbf{g} \times \mathbf{d}$, by an angle that is either small, for passing interest, or sufficient to place $\mathbf{g}$ exactly in line with $\mathbf{d}$, for very high interest. In other cases, the "up" direction of the camera frame may be fixed or constrained, making a rotation about the $\mathbf{g} \times \mathbf{d}$ forbidden; in such circumstances, we project $\mathbf{d}$ onto the plane perpendicular to the "up" direction and use the projected vector as the interest direction instead, as in the data modulation example.

Interest vectors can easily be designed using "interest fields" related to the level-sets for implicit surfaces employed, e.g., by Blinn [3]. By defining a 3D scalar function that is large near a selected family of scene points, the designer can use the gradient to specify where the user's attention should be directed whenever the user draws near; the corresponding level-set implicit surfaces define manifolds of equal "attention importance" in the navigation space, and could be displayed optionally as navigation cues. Note that a separate interest field can in principle be supplied for each parameter, allowing, e.g., the camera focal length, to be varied independently in complex ways throughout the navigation.

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 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 [12], 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. Figure 3 illustrates a field that causes very small responses in the foreground depression where the scale is 0.1, and very large responses at the background peak, where the scale approaches 3.

3 Designing Constrained Navigation Applications

3.1 Basic Components

Our constrained navigation paradigm in its basic form requires an interactively renderable 3D scene plus the following:

- **Constraint Surface.** A surface data structure every point of which can be reached in a predictable manner by incremental motions of a 2D mouse. In practice, one would therefore almost always use as building blocks rectangular arrays of 3D points corresponding to projections onto the 2D rectangular mouse coordinates. These can be joined as in Figure 2b to form a patchwork of polygons that can be traversed incrementally. More complex surfaces (e.g, multiple coverings, multi-branched soap-bubbles) may be used in a similar fashion for particular applications. The most intuitive constraint surface is a sidewalk-like mesh of 3D points, but nothing prevents us from choosing, e.g., latitude and longitude of camera orientation.

Creating a constraint surface for a given problem can be facilitated in some cases by studying the features of the problem. For example, the toroidal navigation surface chosen in Figure 10 is essentially a level set of the electron density. Complex topological objects and terrain models can provide their own initial navigation surfaces by creating parallel surfaces a fixed distance away, or projected outward from the surface normals. Many problems thus contain strong hints to guide the design of an appropriate family of constraint surfaces.

- **Camera Model Field.** At each point of the constraint surface, the designer must attach those values of the camera model field complementary to the constraint surface (orientation if the constraint surface is spatial, position if the constraint surface is orientation, etc.). Thus at each point of the constraint surface array we typically construct a data structure consisting of the variables $\mathbf{G}(u, v) = (x,\, y,\, z,\, q_0,\, q_1,\, q_2,\, q_3,\, f)$, which describe the 3D position, the orientation in terms of a quaternion frame, and the focal length (or perhaps the camera frustum). In practice, these fields would normally be specified at key vertices and interpolated to the intermediate points of the constraint surface.

3.2 Interpolation

Given the normal situation where only a finite number of sample points appear in the array of camera model fields, we require $\mathbf{G}(u, v)$ to be interpolated at intermediate points. This is typically accomplished for rectangular sample spaces by taking local 4×4 rectangular grids of anchor points and performing a bicubic Catmull-Rom spline interpolation, thus ensuring that all grid field values are actually on the interpolated surface. Quaternions must be used to achieve smooth orientation interpolations as noted by Shoemake [18, 19], and refined in subsequent work such as that of Schlag [17], Nielson [13], and Kim, et al. [11]; 2D rectangular extensions of these methods are straightforward. Other variables such as the focal length and controller response field can be interpolated similarly in tandem.

However, experiments with our applications made it clear that one cannot in general produce interpolations based on arbitrary anchor values that produce equivalent perceptions of smoothness in *both* camera position and orientation (or focal length, or whatever). If the knot points are equally spaced in spatial position, the orientation changes may not be uniformly spaced, and vice versa. Among the solutions to this problem currently being investigated are: the adoption of a combined metric in the full parameter space to define a hybrid variety of uniformly spaced knot points, the use of a dynamical model resembling a moving gyroscope that is solved to determine the camera motion, and a similar generalization of the method of Barr, et al. [1] to include spatial parameters as well.

3.3 Methods for Determining the Camera Model Field

We next present a selection of approaches that can be used to determine the camera model structure at any particular point of a navigation path.

Constant key vertices. The simplest configuration utilizes a designer-supplied grid of constant camera parameters, along with a procedure for interpolation among the grid points. The predefined key vertex method is well-adapted to many classic applications, and can easily be understood (and even defined) as a family of deformations of a single fixed camera-animation path.

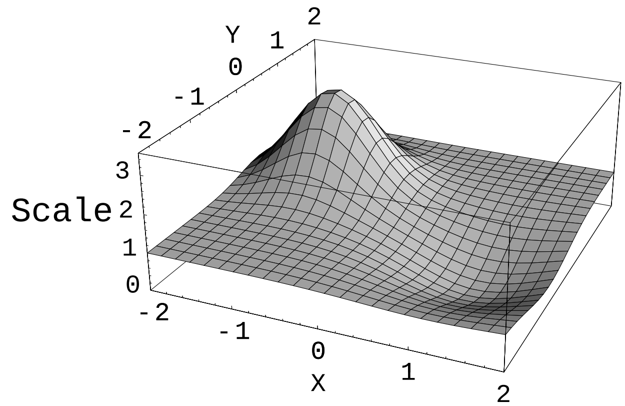


Figure 3: A scaling field that could be used, in regions of value greater than unity, to magnify the screen distance traversed by a unit mouse motion; similarly, in regions of value less than unity, this field would slow the mouse response to provide fine-grained control in those limited areas where it is required.

Space-walk frames and constrained "up" fields. The basic manifold traversal method of Hanson and Ma [8] can be used with 2D constraint manifolds of arbitrary complexity, and is extensible to 3D as well. Effective use of the method requires data stored in a winged-edge format rather than the simpler 2D parametric rectangular grid format that we have implicitly assumed for most of the discussion. The intrinsically defined transitions from polygon to polygon allow one to navigate a complex surface keeping the world "up" direction aligned with the surface normal throughout the transversal. While it is natural to have the gaze direction pointed in the direction of motion, this is not required; fixed camera parameters can be prestored at each vertex and modulated either by scene features or the default space-walk camera frame.

Another interesting variant is to specify only the "up" direction of the camera frame at each point (manually or from the normal to the constraint manifold); then the camera has a single rotational degree of freedom at each point that can be determined from the context, e.g., viewer velocity, or other data.

3.4 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. See Figure 8 for an example.

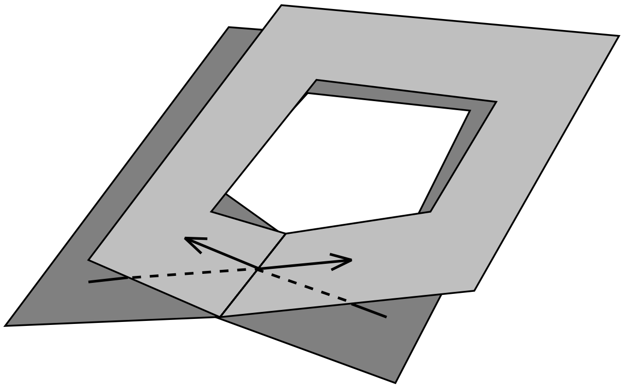


Figure 4: An example of a navigation manifold that contains more than one possible layer, hence more than one possible camera model, depending on one's route to the scene.

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 [12] 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 [16]. An example is given in Figure 9.

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 4, 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 11. The reader can imagine arbitrarily complex variants, including instantaneous state transitions between entirely different guide fields.

3.5 Dynamic Mapping Techniques

Several prospects for more complex control strategies appear promising for future work.

Lead time. Sometimes we want to have the system react to where we *will* be, not where we are. This leads one to implement virtual navigation avatars (we might call them "navatars") sailing in front of the viewer, and requires some predictive computation. Once the hypothesized avatar position is determined by an appropriate algorithm, the designer can present varying options tying the motion more or less closely to the avatar, or perhaps allowing diversions in the avatar's path.

Viewer state procedures and rules. The user state in a navigation problem contains a number of variables that can be tracked and computed, particularly those involving velocity and heading history (see, e.g., some of the techniques reviewed in Chen et al. [5]). Arcade games often exploit such information, particularly to add challenge to a control strategy by preventing direct manipulation of the object to be controlled. In physical simulations, momentum, friction, and air resistance play a crucial role in making driving and flight simulators realistic. Such factors can be incorporated into the procedures or rules determining the evolution of the camera field on the constraint surface to accomplish a number of intuitive physical effects.

Context-based rules. A variety of approaches have been proposed in the literature to use context-based knowledge, expert system domain rules, and artificial intelligence planning methods to determine transitions among camera positions in animation or even complete animation paths (see, e.g., [9, 10, 2]). It is clearly appropriate to apply such techniques to the more general philosophy of constrained navigation proposed here; this is a fertile area for future research.

4 Examples

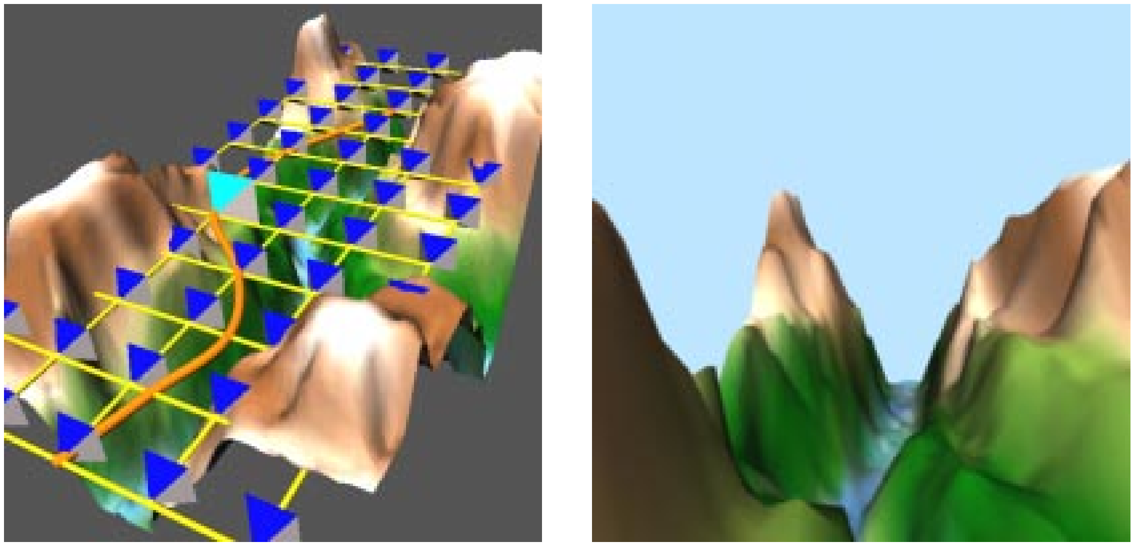
In this section, we present a series of examples realized by implementations using the Open Inventor class libraries in the IRIS Explorer and Open Inventor environments; we note in particular that many of the needed quaternion-based classes and methods are already supplied. We implemented our own Catmull-Rom interpolator based on the Schlag algorithm [17].

Wandering Camera Path with Wandering View. In a traditional computer animation, the camera itself may follow many different constraints such as looking at a single point on the ground throughout the motion, tracking a moving object in the scene, or staring in a fixed direction. Figure 5(a,b) shows a generalization of the latter with the viewer's trajectory confined to a plane. In Figure 6a, the path is still constrained to the plane, but designer-placed camera orientations are used as key vertices for a quaternion spline interpolation; Figure 6b shows the scene viewed from the same point as Figure 5b, but with the modified camera field.

Terrain Navigation: Conservative Flight Path. In Figure 7, we show a more realistic guide manifold for navigating a terrain model; we employ a contoured 3D constraint surface and constrain the camera "up" vector to be the surface normal. The camera orientation at each point is determined by rotating relative to the constant gaze direction to look slightly in the direction of the terrain gradient below. We note that we need not require a global "up" direction; if desired, we can transition smoothly from "right-side-up" in the world to "upside-down" (see below).

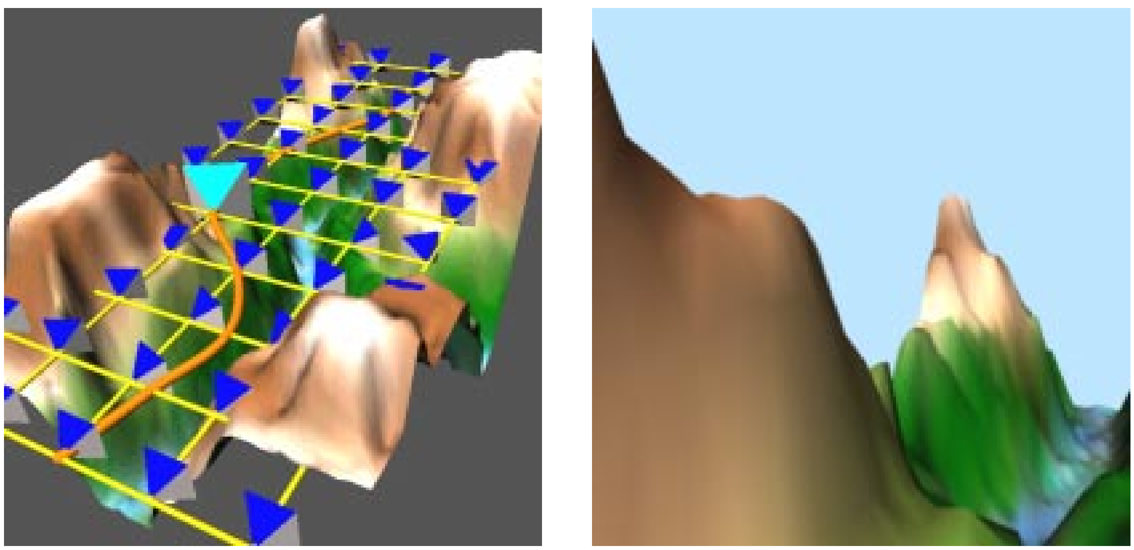
Spotlight Attention Focus. An example of the spotlight technique, which can be used to focus the user's attention on a point that is not necessarily aligned with the direction of the camera gaze or the direction of motion, is shown in Figure 8.

Terrain Navigation: Vista Point Ahead! A tour designer in the paradigm presented here has not only the ability to keep wandering users in a limited set of viewpoints and to keep their attention focused only on what they are supposed to see, but also to prepare special treats. In particular, the constraint surface itself may vary dramatically, and the focal length can be controlled and interpolated throughout the grid just like the other variables. In the scenario presented in Figure 9, the designer has placed two "vista points" in the scene which the user may approach at will while roaming the constraint space. Figure 9a focuses on one particular point that causes the user to rise rapidly above the world to a very high vantage point,



(a) (b)

Figure 5: Camera path constrained to plane with fixed camera orientation. (a) View of path and camera model control points on constraint surface. (b) View using camera model field at selected point.



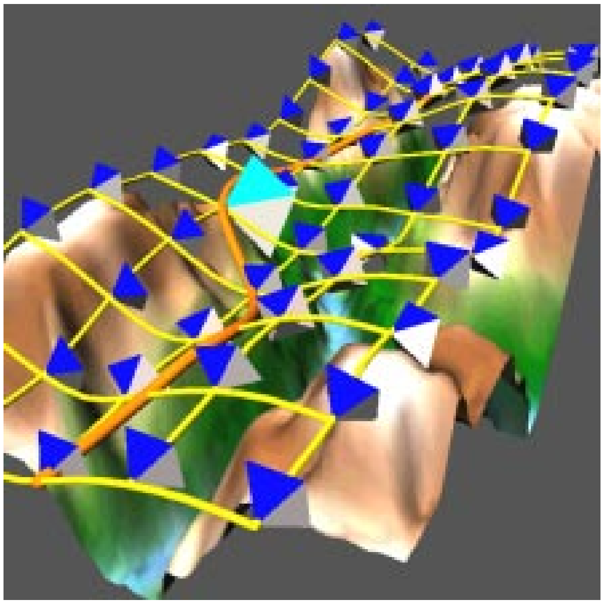
(a) (b)

Figure 6: Camera path constrained to plane with camera orientation 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.

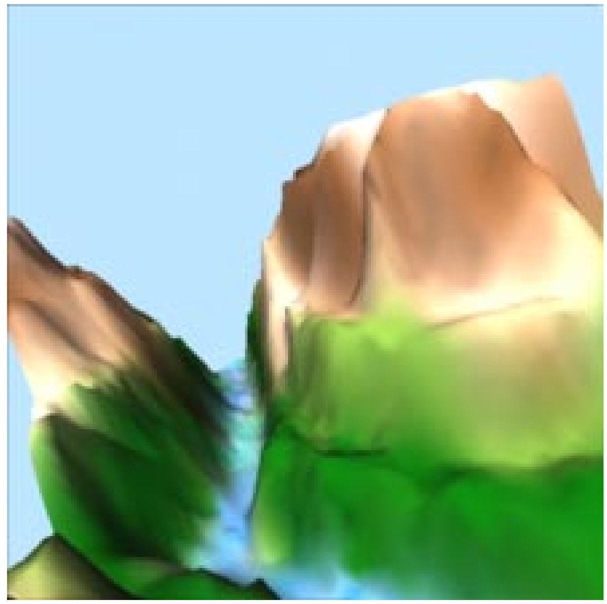
while the camera is forced to look down below at the retreating scene data, creating the view of Figure 9b. Figure 9c is rather like a highway rest stop, where approaching a particular point on the constraint surface swings your gaze direction around, points at a landmark you might never have noticed otherwise, and puts a "telephoto lens" on the camera so that the view automatically zooms in on the point in question.

Molecule Navigation. The most challenging applications for constrained navigation involve the perusal of objects with no natural orientation. Here we have both the advantage of being permitted great flexibility, and the drawback of having to decide on a particular guiding strategy. Figure 10a shows how we have chosen a toroidal navigation manifold that entirely envelops a helical molecule. This constraint surface allows us to move quickly to every conceivable viewpoint on the molecule with a series of very simple mouse strokes. To keep the user in context, we make the "up" direction inside the molecule the same direction as outside, while tilting a bit at the top and the bottom to keep focused on the structure and give a clear end-on view, as shown in Figure 10c. Here the goal of the navigation was to give the viewer a fluid way to see every conceivable surface, inside and outside, of the virtual cylinder around which the helical molecule is wrapped.

Architectural Interior Navigation. More complex topologies arise naturally when we examine detailed 3D structures such as buildings and room interiors. Here it is natural to include new levels of constraints and choices. In the example of Figure 11, we restrict



(a)



(b)

Figure 7: 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.

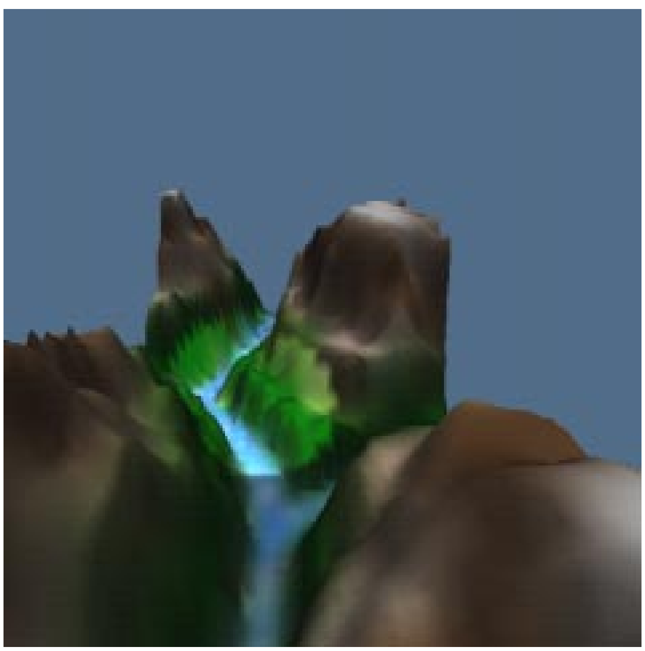


Figure 8: Spotlight focused on an area of interest that is slightly displaced from camera gaze and motion directions. This allows greater flexibility in keeping the context while redirecting attention.

user motion in a single room to encircle an object of interest, which happens to be a model of a virtual reality environment. This simple example of a multiple-patch data structure is used to define a double circuit of "carpeting" around the object of interest like that noted also in Figure 4; this guide manifold serves both to prohibit areas with physical obstructions, and to permit different things to be emphasized on even and odd tours around the room. Thus, the goal of the first circuit of the walkway is to focus on the display screens, while the second time around we use an effective interest field to focus instead on the placement of the projectors.

5 Preliminary Human Factors Observations

Substantive human factors studies of the comparative effectiveness of particular scenarios are beyond the intended scope of this paper. Nevertheless, an evaluation of alternative exploration modes is currently being pursued for room-like worlds, using criteria inspired by those of Thorndyke et al. [24, 23, 22]. Preliminary results, which will be extended and presented elsewhere, suggest that 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. Another, fairly obvious, experimental observation is the fact that keeping the camera's vertical axis relatively stable is important to prevent users from developing motion discomfort.

6 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. Detailed examples have been worked out and presented for the particular case of a 3D through-the-screen display controlled by a 2D mouse. The basic strategy is to supply a set of view-determining data at each sample point of a "virtual sidewalk," along with possibly state-dependent procedures to create the actual view to be presented. Ultimately, it is up to the designer to 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, human factors testing, and additional experimentation with "smart" controls that balance prestored constraints against user state. An ideal system would likely include a history-sensitive expert system to recompute the camera model at each step the viewer takes on the journey.

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. We are grateful to Stephen Hughes for his essential contributions to the preliminary user interface studies. Thanks are also due to the staff of CICA, the Indiana University Center for Innovative Computer Applications, for their support. This research was made possible in part by NSF infrastructure grant CDA 93-03189.

References

[1] A. Barr, B. Currin, S. Gabriel, and J. Hughes. Smooth interpolation of orientations with angular velocity constraints using quaternions. In *Computer Graphics Proceedings, Annual Conference Series*, pages 313–320, 1992. Proceedings of SIGGRAPH '92.

[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] F. P. Brooks. Walkthrough — a dynamic graphics system for simulating virtual buildings. In *Computer Graphics*, pages 9–21, 1987. Proceedings of 1986 Workshop on Interactive 3D Graphics.

[5] M. Chen, S. J. Mountford, and A. Sellen. A study in interactive 3-d rotation using 2-d control devices. In *Computer Graphics*, volume 22, pages 121–130, 1988. Proceedings of SIGGRAPH 1988.

[6] 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.

[7] A. J. Hanson. The rolling ball. In David Kirk, editor, *Graphics Gems III*, pages 51–60. Academic Press, Cambridge, MA, 1992.

[8] A. J. Hanson and H. Ma. Space walking. In *Proceedings of Visualization '95*, pages 126–133. IEEE Computer Society Press, 1995.

[9] P. Karp and S. Feiner. Issues in the automated generation of animated presentations. In *Graphics Interface 1990*, pages 39–48, 1990.

[10] P. Karp and S. Feiner. Automated presentation planning of animation using task decomposition with heuristic reasoning. In *Graphics Interface 1993*, pages 118–127, 1993.

[11] M.-J. Kim, M.-S. Kim, and S. Y. Shin. A general construction scheme for unit quaternion curves with simple high order derivatives. In *Computer Graphics Proceedings, Annual Conference Series*, pages 369–376, 1995. Proceedings of SIGGRAPH '95.

[12] 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.

[13] G. M. Nielson. Smooth interpolation of orientations. In N. M. Thalman and D. Thalman, editors, *Computer Animation '93*, pages 75–93, Tokyo, June 1993. Springer-Verlag.

[14] G. M. Nielson and Dan R. Olson. Direct manipulation techniques for 3d objects using 2d locator devices. In *Computer Graphics*, pages 175–182, 1987. Proceedings of 1986 Workshop on Interactive 3D Graphics.

[15] 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.

[16] 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.

[17] J. Schlag. Using geometric constructions to interpolate orientation with quaternions. In James Arvo, editor, *Graphics Gems II*, pages 377–380. Academic Press, 1991.

[18] K. Shoemake. Animating rotation with quaternion curves. In *Computer Graphics*, volume 19, pages 245–254, 1985. Proceedings of SIGGRAPH 1985.

[19] K. Shoemake. Animation with quaternions. Siggraph Course Lecture Notes, 1987.

[20] K. Shoemake. Arcball rotation control. In Paul Heckbert, editor, *Graphics Gems IV*, pages 175–192. Academic Press, 1994.

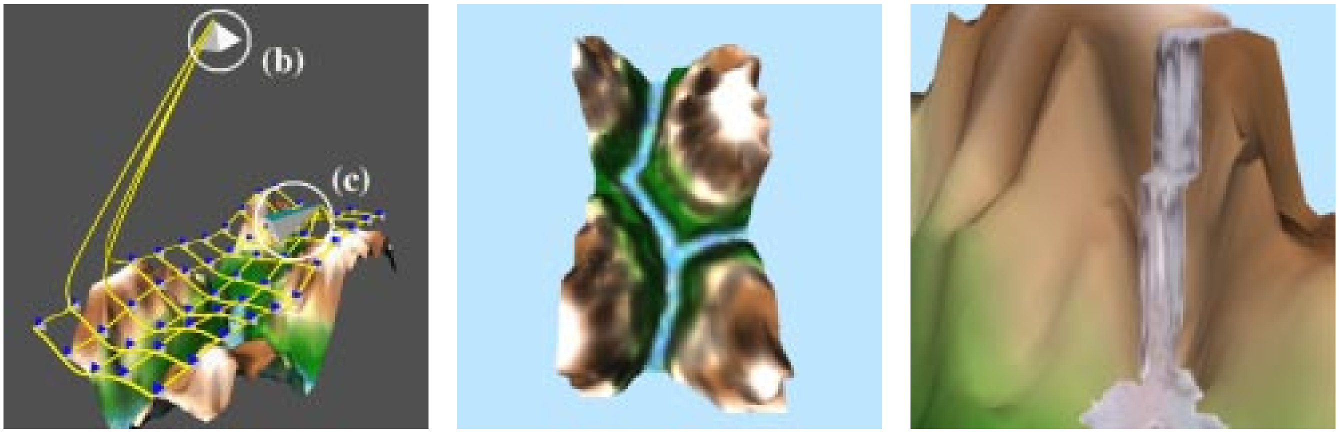
[21] K. Shoemake. Fiber bundle twist reduction. In Paul Heckbert, editor, *Graphics Gems IV*, pages 230–236. Academic Press, 1994.

[22] 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.

[23] P. W. Thorndyke and B. Hayes-Roth. Differences in spatial knowledge acquired from maps and navigation. *Cognitive Psychology*, 14:560–589, 1982.

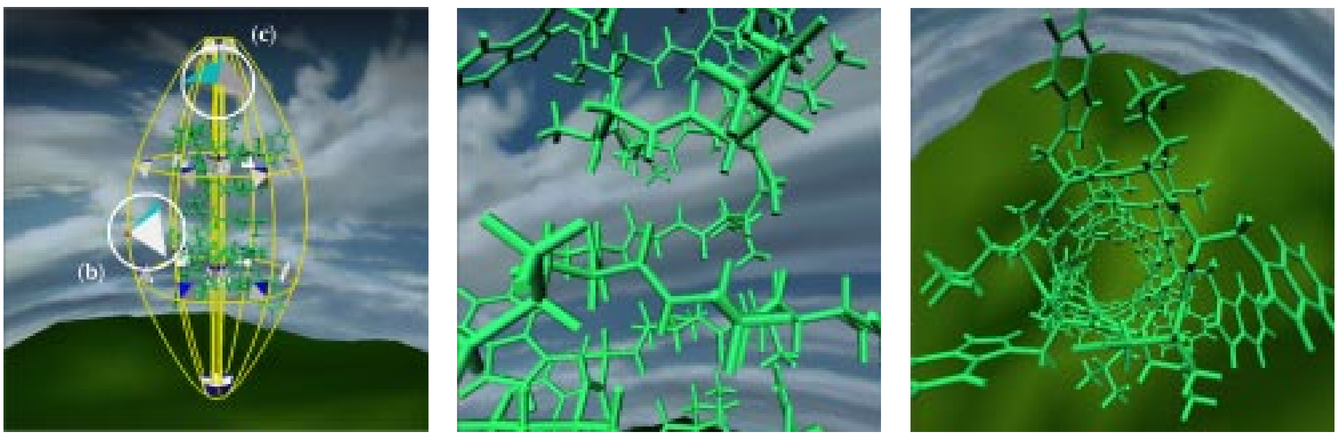
[24] P. W. Thorndyke and C. Stasz. Individual differences in procedures for knowledge acquisition from maps. *Cognitive Psychology*, 12:137–175, 1980.

[25] 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.



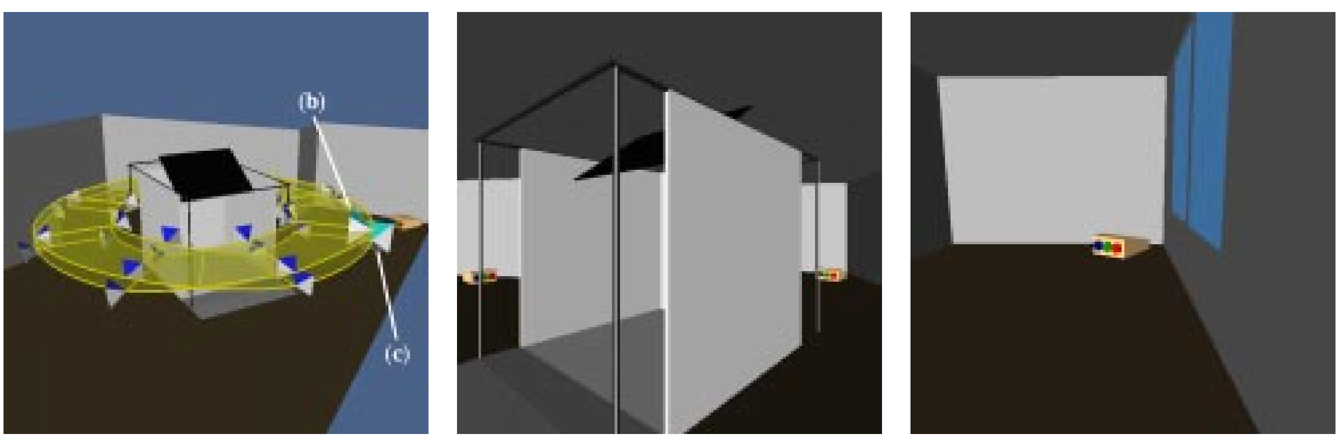
(a) (b) (c)

Figure 9: (a) A constraint surface with an "overlook" and a "vista point" having a telephoto lens. (b) View of scene from overlook. (c) Zoomed view of base of overlook from vista point.



(a) (b) (c)

Figure 10: (a) Toroidal constraint surface appropriate to the large cylindrical molecule shown. (b) Choice of camera parameters at the midsection of the molecule. (c) Choice of camera parameters at the ends of the molecule provides a clear holistic view down the central core.



(a) (b) (c)

Figure 11: (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.