Space Walking

Andrew J. Hanson and Hui Ma
Department of Computer Science
Indiana University
Bloomington, IN 47405

Abstract

We propose an interactive method for exploring topological spaces based on the natural local geometry of the space. Examples of spaces appropriate for this visualization approach occur in abundance in mathematical visualization, surface and volume visualization problems, and scientific applications such as general relativity. Our approach is based on using a controller to choose a direction in which to "walk" a manifold along a local geodesic path. The method automatically generates orientation changes that produce a maximal viewable region with each step of the walk. The proposed interaction framework has many natural properties to help the user develop a useful cognitive map of a space and is well-suited to haptic interfaces that may be incorporated into desktop virtual reality systems.

1 Introduction

In this article, we describe an interactive approach to visualizing topological manifolds. The interface can be realized in terms of standard mouse or joystick controllers as well as haptic interfaces. Properties of surfaces with high complexity due to strong curvature or self-intersections can be explored dynamically and pieced together mentally by the user in a manner difficult to achieve with standard graphical representations alone.

This work originates philosophically in our attempts to develop increasingly deeper conceptualizations of the properties of complex geometric objects (for a general overview, see Hanson, Munzner, and Francis [14]). Examples of other work in this area are the Geomview system and its auxiliary modules [16, 7], and Bryson's system for visualizing geodesics in gravitational spacetime metrics [1]. However, highly tuned interaction with objects represented by multiple coordinate patches in three, four, or more dimensions requires additional special tools: our own MeshView system [15] was developed precisely to enable enhanced 4D interaction, flexible 4D depth cues, and especially to display the relationship between a point on an abstract parameter mesh and its mapping onto a self-intersecting surface in a computer graphics image. 4D lighting methods can also supply additional cues [13]. Long experience viewing and manipulating manifolds such as projective planes, everting spheres, and Riemann surfaces (see, e.g., Carter [2], Francis [5], and Hanson [8]) suggests the need for additional intuitive tools for exploring such spaces. The techniques described here comprise another powerful, interaction-based approach for developing mental models of such structures (see, e.g., Tversky [19]).

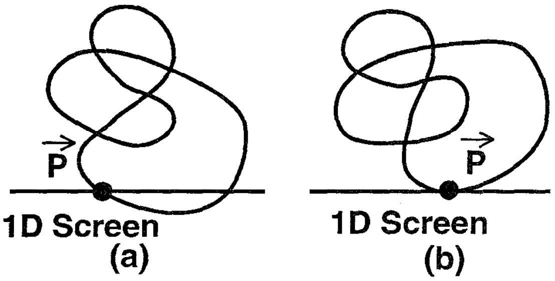

Figure 1: (a) A 2D curve with interest point $\vec{\mathbf{P}}$ as it might intersect a 1D display. (b) The curve rotated so the tangent vector at the interest point is aligned with the screen, giving a maximal projection.

2 Walking a Curve

In this section, we introduce the reader to space walking using the traversal of a space curve, a simple example that exposes the richness of the approach.

Assume that we have a closed 2D space curve along with a particular point of interest $\vec{\mathbf{P}}$ on that curve as shown in Figure 1. Now imagine projecting that curve to a single scanline of a CRT screen. To traverse or "walk" the curve, we must determine both a display strategy that represents the current interest point $\vec{\mathbf{P}}$ in a natural manner, and we must specify how to rigidly transform the full curve to make a transition from $\vec{\mathbf{P}}$ to a neighboring interest point $\vec{\mathbf{P}}'$. In this simple case, the controller can be thought of as a slider, 1D mouse,

(See color plates, page CP-15)

1070-2385/95 $4.00 © 1995 IEEE

Proceedings of the 6th IEEE Visualization Conference (VISUALIZATION '95)
1070-2385/95 $10.00 © 1995 IEEE

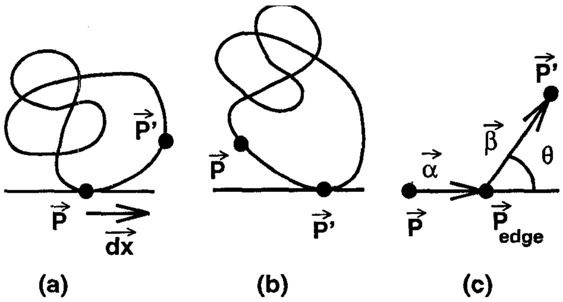

Figure 2: (a) Controller motion $\vec{\mathbf{dx}}$ initiating transition from interest point $\vec{\mathbf{P}}$ to the next interest point $\vec{\mathbf{P}}'$. (b) Rolling the curve to present a maximal projection at the new interest point. (c) Detail of the geometry involved in the transition using a discrete curve tessellation.

or 1D joystick.

Positioning the interest point using maximal projection. As shown in Figure 1a, it does not make sense to let the 2D projection of the curve at $\vec{\mathbf{P}}$ have an arbitrary oblique orientation with respect to the screen face; our first assumption, which we will maintain throughout, is that we should always attempt to orient the tangent to the curve *parallel to the screen face* at the interest point $\vec{\mathbf{P}}$, as indicated in Figure 1b. We will call this choice the *maximal projection.*

Positioning the curve. Now let $\vec{\mathbf{dx}}$ be the interactive controller input, which is interpreted to mean "move along the curve in the direction $\vec{\mathbf{dx}}$ by arc length $\|\vec{\mathbf{dx}}\|$." The resulting action changes the interest point from $\vec{\mathbf{P}}$ to $\vec{\mathbf{P}}'$ as we slide along the curve as shown in Figure 2a,b. Suppose for simplicity that the curve is represented as discrete line segments, as shown in Figure 2c. If $\vec{\mathbf{P}} + \vec{\mathbf{dx}}$ does not reach the edge vertex $\vec{\mathbf{P}}_{\mathrm{edge}}$, the simplest motion is to translate by $\vec{\mathbf{dx}}$ in the screen tangent direction. Otherwise, we take $\vec{\alpha} = \vec{\mathbf{P}}_{\mathrm{edge}} - \vec{\mathbf{P}}$, $\vec{\beta} = \vec{\mathbf{P}}' - \vec{\mathbf{P}}_{\mathrm{edge}}$, set $\|\vec{\mathbf{dx}}\| = \|\vec{\alpha}\| + \|\vec{\beta}\|$, translate $\vec{\mathbf{P}}_{\mathrm{edge}}$ to the origin, rotate by $\cos\theta = \vec{\alpha} \cdot \vec{\beta} / \|\vec{\alpha}\| \, \|\vec{\beta}\|$, and make a final translation by $\|\vec{\beta}\|$ to place $\vec{\mathbf{P}}'$ at the screen center, where it becomes the new interest point $\vec{\mathbf{P}}$. One obvious strategy for smoothing this transition is to do the rotation in finite time using $\theta(t) = t\theta$ with t increasing from zero to one.

Remark: Self-intersecting curves are treated naturally in this framework by requiring local continuity of the tangent vector. A natural source of interesting self-intersecting curves is the class of non-self-intersecting 3D curves that possess a self-intersecting 2D projection. Accordingly, we next examine 3D space curves, and find that they introduce new interesting issues and intrinsic properties that generalize easily to ND space curves. 3D subspaces turn out to play a special role in walking curves. Later, we will see that 4D subspaces play a similar special role in the analysis of surface walking.

Extension to 3D space curves. First suppose we have a 3D space curve, but are still restricted to a 1D display screen, and have chosen a 2D projection. When we restrict the tangent to the interest point of the curve's 2D projection to lie in the 1D screen face, we see that the 3D curve will be tangent to the plane perpendicular to the line of sight through the interest point, but may make an oblique angle to the 1D screen line drawn in this plane. We then have the following choices for positioning the 3D curve:

- **2D tangent.** Ignore 3D altogether, make the 2D projection tangent to the 1D screen's center at the interest point $\vec{\mathbf{P}}$ and walk the 2D projection only, as in Figure 2.

- **3D tangent, 2D adjustment.** If the 3D curve's tangent at $\vec{\mathbf{P}}'$ does not lie in the projection plane, first align it with the projection plane as shown in Figure 3a,b. Then rotate the result in the projection plane to produce the maximal projection at $\vec{\mathbf{P}}'$ as shown in Figure 3c.

- **3D tangent, 3D adjustment.** Alternatively, ignore the 2D projection altogether, translate directly from $\vec{\mathbf{P}}$ to $\vec{\mathbf{P}}'$, and rotate the entire 3D curve so that the tangent at $\vec{\mathbf{P}}'$ aligns with the 1D screen line. The only change is that now $\vec{\mathbf{P}}'$ is a 3D vector and the rotation is in the 3D plane defined by the three (assumed non-collinear) points $(\vec{\mathbf{P}}, \vec{\mathbf{P}}_{\mathrm{edge}}, \vec{\mathbf{P}}')$. The angle of rotation corresponds to the 3D dot product, with the geometry as in Figure 2c.

  At this point, there is an arbitrary rotational degree of freedom about the curve's new tangent vector, the screen line; we have the choice of leaving the orientation where it falls after aligning the tangent, which is quite arbitrary, or choosing a geometrically motivated orientation. Provided the curve's second derivative does not vanish, we can choose an intrinsic orientation based on the Frenet frame [4, 6],

$$\vec{\mathrm{T}}(s) = \frac{\vec{\mathrm{x}}'(s)}{\|\vec{\mathrm{x}}'(s)\|}$$

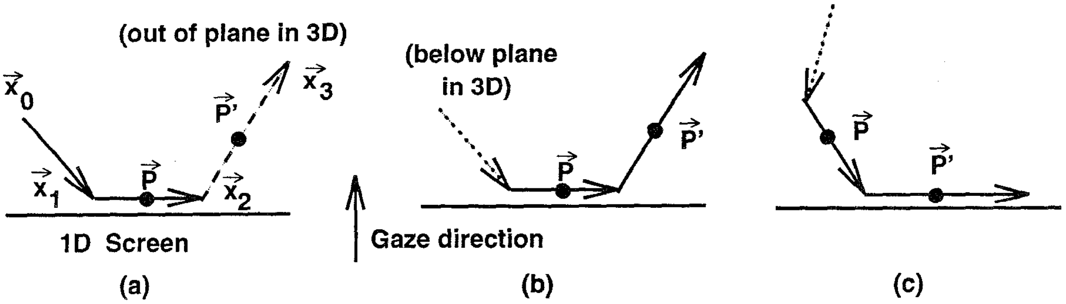

Figure 3: One possible approach to maintaining maximal projection of a 3D space curve during the transition between facets. (a) Maximal projection of a tessellated section of a 3D space curve is aligned so that the triangle $(\vec{x}_0, \vec{x}_1, \vec{x}_2)$ lies in the plane defined by the screen and the gaze direction. (b) Rocking $\vec{P}'$ into the screen plane by rotating around $\vec{x}_2 - \vec{x}_1$. (c) Rotate in screen plane to achieve maximal projection of the segment containing $\vec{P}'$.

$$\begin{aligned}\vec{B}(s) &= \frac{\vec{x}'(s) \times \vec{x}''(s)}{\|\vec{x}'(s) \times \vec{x}''(s)\|} \\ \vec{N}(s) &= \vec{B}(s) \times \vec{T}(s) \, .\end{aligned}$$

A natural choice is to force the tangent circle spanned by $\vec{T}$ and $\vec{N}$ at $\vec{P}$ to lie entirely in the 2D projection plane.

N dimensions. In higher dimensions, we note the remarkable fact that the local characteristics of the curve that concern us lie within a 3D subspace: the points $(\vec{x}_0, \vec{x}_1, \vec{x}_2, \vec{x}_3)$ defining the tangents to the sequence of interest points as indicated in Figure 3 define a tetrahedron, a 3-simplex, regardless of the dimension of the four N-dimensional vectors describing the endpoints of the three edges. Thus we may rephrase the alignment of the Frenet curve normal $\vec{N}$ with the 2D projection plane for arbitrary dimension as follows: (1) define the plane containing the screen vector and the gaze vector to be aligned with the edges containing $(\vec{x}_0, \vec{x}_1, \vec{x}_2)$. The desired new orientation has the tangent at $\vec{P}'$ parallel to the screen vector; furthermore, we want the plane containing the screen vector and the gaze vector to be aligned with the plane of the edges $(\vec{x}_1, \vec{x}_2, \vec{x}_3)$. Since this transformation is strictly in a 3D subspace, one has several choices: (1) align $\vec{P}'$ with the screen direction first and then rotate about the line containing $\vec{P}'$ to place the line containing $\vec{P}$ back in the projection plane; (2) rotate about the line containing $\vec{P}$ until the line containing $\vec{P}'$ is in the projection plane, then rotate in that plane until $\vec{P}'$ is in the screen tangent direction; (3) better yet, one can perform a geodesic quaternion interpolation between the two frames within the 3D subspace [18].

The strategies that align the 3D tangent instead of the simpler 2D projected tangent have the effect of "pulling the curve through a tube" surrounding the 1D screen line. Also note that in these algorithms the curve may rotate rapidly about the 1D screen line if the maximal projection aligning the curve's Frenet normal with the 2D projection plane is always enforced.

3 Walking in Space

We next outline the extension of the basic concepts of curve walking to surfaces and higher dimensional manifolds. From this point on, we will assume that we have a 2D screen space with a well-defined normal gaze direction, and that surface patches are projected from a 3D graphics space, the "projection volume," onto this 2D screen. Higher dimensional objects will typically first be transformed to the 3D projection volume in a manner analogous to the projection of a 3D curve to a 2D projection plane noted earlier. We begin by examining surfaces (2-manifolds) with vertex coordinates in 3D Euclidean space. We assume that a surface is defined by a collection of vertices $\{\vec{x}_i\}$ with a data structure such as a *winged edge* data structure that allows one to determine the faces adjoining each edge and each vertex.

Paths on a surface in 3D. First, we consider a surface represented by triangular facets in ordinary 3D space. Our basic hypothesis of *maximal projection* then requires that the facet of interest lies completely in the screen plane.

Next, as illustrated in Figure 4, we choose a point of interest $\vec{P}$ in the facet and give an algorithm for transforming the display as we move to a new interest point $\vec{P}'$. Assuming a 2D controller motion $\vec{dx}$, we proceed as follows:

Clip. First, we clip the vector $\vec{P} + \vec{dx}$ to the triangle containing $\vec{P}$. That is, working within the

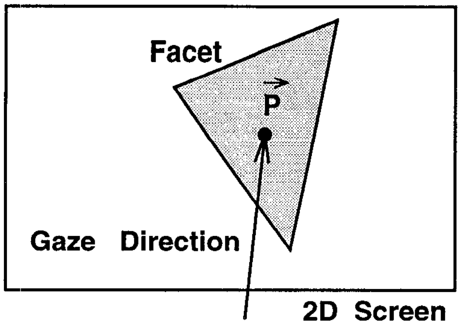

Figure 4: Surface facet with interest point $\vec{P}$ maximally projected onto a 2D screen from an implicit 3D graphics space behind the screen.

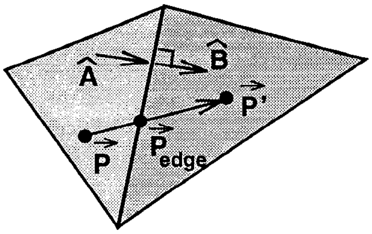

Figure 5: Flattened facet-to-facet transition.

2D screen coordinate system with triangle coordinates $(\vec{x}_0, \vec{x}_1, \vec{x}_2)$, compute each normal $\vec{n}_{01} = (-(y_1 - y_0), (x_1 - x_0))$, etc., and each $t = \vec{n}_{01} \cdot (\vec{x}_0 - \vec{P}) / \vec{n}_{01} \cdot \vec{dx}$, etc.; if all positive t's are greater than one, $\vec{P}' = \vec{P} + \vec{dx}$ lies in the face; otherwise the minimum positive t gives the intersection point $\vec{P}_{edge} = \vec{P} + t\vec{dx}$.

If $\vec{P}'$ lies in the current face, simply translate that point to the screen center. If $\vec{P}'$ lies in the adjacent face, rotate as prescribed below and then translate $\vec{P}'$ to the screen center. If $\vec{P}'$ crosses the clipping boundary of the adjacent face, repeat the clipping and rotation procedure until it lies within a face. This procedure amounts to flattening the neighboring facets as shown in Figure 5 to find the destination interest point $\vec{P}'$ in the local 2D coordinate system of the final face.

Rotating between facets. In the facet model, vertices are points of vanishing measure that in fact carry the Gaussian curvature in their angular deficits. For now, we assume we never actually pass through a vertex; all paths will be across edges from one face to another; smoothing models can in principle distribute the curvature across the faces and permit direct traversal of vertices.

We now compute the transition across a given edge, assuming we know the unit vectors $\hat{A}$ and $\hat{B}$ perpendicular to the transition edge, as indicated schematically in Figure 6. We want to translate the origin to the point $\vec{P}_{edge}$ and rotate by the angle $\cos\theta = \hat{A}\cdot\hat{B}$ in the plane containing $\hat{A}$ and $\hat{B}$ so that $\hat{B}'$ now lies in the screen plane and is identical to the original $\hat{A}$.

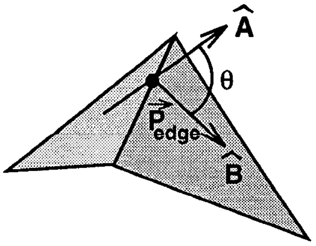

Figure 6: 3D facet rotation.

We use a Gram-Schmidt procedure to derive the norm-preserving rotation using the orthonormal basis vectors $\hat{A}$ and $\hat{C} = (\hat{B} - \hat{A}\cos\theta)/\sin\theta$. (If $\hat{A}$ and $\hat{B}$ are parallel, no rotation takes place and we translate directly to $\vec{P}'$ in the plane of the original facet.) Every vertex $\vec{V}$ in the *entire* data structure undergoes the following rotation:

$$\begin{aligned}\vec{V}' &= \vec{V} - \hat{A}(\hat{A}\cdot\vec{V}) - \hat{C}(\hat{C}\cdot\vec{V}) \\ &\quad + \begin{bmatrix}\hat{A} & \hat{C}\end{bmatrix}\begin{bmatrix}\cos t\theta & \sin t\theta \\ -\sin t\theta & \cos t\theta\end{bmatrix}\begin{bmatrix}\hat{A}\cdot\vec{V} \\ \hat{C}\cdot\vec{V}\end{bmatrix} \qquad (1)\end{aligned}$$

where we may vary t slowly between zero and one to avoid sudden jumps in the orientation.

Surfaces in N dimensions. The equations presented so far in fact extend trivially to arbitrary dimensions for the vertex coordinates. The only essential difference is that the vector $\hat{B}$ may have a component in some direction $\hat{w}$ outside the projection volume spanned by $(\hat{x}, \hat{y}, \hat{z})$, and the procedure given so far rotates only the *projection* of the face of the new interest point $\vec{P}'$ into the screen plane. That projection may be slightly tilted and therefore not maximal unless we make an additional rotation in the z-w plane to line it up and eliminate the $\hat{w}$ component; we may use Gram-Schmidt to define $\hat{w}$ in any dimension as follows:

$$\begin{aligned}\hat{B}' &= \frac{\hat{B} - \hat{x}(\hat{x}\cdot\hat{B})}{(1 - (\hat{x}\cdot\hat{B})^2)^{1/2}} \\ \hat{B}'' &= \frac{\hat{B}' - \hat{y}(\hat{y}\cdot\hat{B}')}{(1 - (\hat{y}\cdot\hat{B}')^2)^{1/2}} \\ \hat{w} &= \frac{\hat{B}'' - \hat{z}(\hat{z}\cdot\hat{B}'')}{(1 - (\hat{z}\cdot\hat{B}'')^2)^{1/2}} \ .\end{aligned}$$

Tessellated surfaces in arbitrary dimensions $N > 3$ can therefore be handled in a manner similar to our

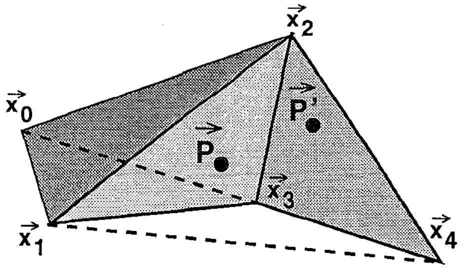

Figure 7: Aligning surface facets with a 3D subspace in dimensions 4 or greater exploits 5 vertices that define 3 neighboring triangles, 2 overlapping tetrahedra (3-simplexes) whose sixth edges are the dotted lines, and one nondegenerate 4-simplex.

discussion for curves in dimensions $N > 2$. The essence of the argument, represented in Figure 7, is that the two triangles and four vertices of a winged edge in any dimension define a 3D subspace that we can require to be aligned with the 3D volume that eventually projects to the 2D screen. When making a transition to a new interest point $\vec{\mathbf{P}}'$, we introduce a *new* 3D subspace that shares the x-y plane with the facet containing $\vec{\mathbf{P}}$ and has a new component in the $\hat{\mathbf{B}}$ direction. If $\hat{\mathbf{B}}$ has components only in the $(\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}})$ subspace, we are done.

If not, we rotate the entire object and generate a maximal projection that not only has $\vec{\mathbf{P}}'$'s face lying in the screen x-y plane, but also has the tetrahedron containing $\vec{\mathbf{P}}$ and $\vec{\mathbf{P}}'$ contained completely in the 3D projection volume. *This is the analog for surfaces of computing the Frenet frame curve normal and readjusting the curve normal to lie in the plane normal to the screen.*

As in the space curve treatment, we have three alternatives: (1) Apply the analog of Eq. (1) to rotate $\hat{\mathbf{B}}$ in the z-w plane so that it has no $\hat{\mathbf{w}}$ component, then perform the standard 3D facet transition to align the new $\hat{\mathbf{B}}$ with the original $\hat{\mathbf{A}}$. (2) First align the new $\hat{\mathbf{B}}$ with the original $\hat{\mathbf{A}}$, then apply Eq. (1) in the z-w plane to eliminate the $\hat{\mathbf{w}}$ component. (3) Identify the 4D subspace (the 4-simplex) generated by the 5 vertices shown in Figure 7, compute the 4D frame with normal $\hat{\mathbf{m}}$ to the 3-simplex defined by the "old" winged edge faces, compute the distinct 4D frame with normal $\hat{\mathbf{n}}$ relative to the 3-simplex defined by the "new" winged edge faces containing $\vec{\mathbf{P}}$ and $\vec{\mathbf{P}}'$, then perform a geodesic rotation between the two 4D frames. Details of the required methods may be found in Hanson [10, 11].

Remark: Earlier, we showed how the Frenet-frame treatment of space curves is essentially exhausted in 3D. The method just described shows how a winged-edge treatment of arbitrary surfaces naturally leads to 4D subspaces, and that no further complexities arise in developing an analogous treatment of surface geometry in higher dimensions. Extending this treatment to winged faces for walking three-manifolds intrinsically requires 5D subspaces; K-manifolds whose tessellated geometry is described by winged $(K-1)$-planes will require $(K+2)$-dimensional subspaces to treat the most general maximal projection requirements in arbitrary dimensions.

4 Facet-based vs Smoothly Interpolated Control

Our approach so far has treated manifolds as locally planar segments in the manner of flat-shaded polyhedral computer graphics models and the Regge calculus [17] treatment of Riemannian geometry. Thus all transitions between facets across winged edges are indistinguishable from walking on flat space; all the curvature is contained in the "angular deficits," the difference between 2π and the actual sum of facet angles at each vertex. For relatively dense manifold triangulations, the angular deficit at each vertex is very small, and navigation of the manifold appears fairly smooth (see the examples in Figures 8, 9, and 10). However, just as we may wish to approximate the appearance of smooth shaded surfaces starting from an underlying discrete tessellation, we may wish to extend the walker application with an interpolation scheme that makes the manifold appear to be smooth. For example, we see in Figure 3 that we really want a curve's segment-based Frenet normal and tangent aligned in the screen projection plane at $\vec{\mathbf{x}}_1$, not at $\vec{\mathbf{P}}$, which should already be interpolating between the orientations at $\vec{\mathbf{x}}_1$ and $\vec{\mathbf{x}}_2$; a similar observation holds for surface walking as well.

The first step in generating a smoothly-varying manifold walk is to produce a local tangent line at each vertex of a curve, a local tangent plane at each vertex of a surface, and so on. The second, and more problematic, task is to produce a compelling interpolation procedure that connects these tangent frames continuously to one another over the entire manifold.

In 3D, for example, the standard interpolated shading algorithm that computes a vertex normal by averaging the normals to the surrounding faces is a good start; this normal defines the 3D tangent plane at the vertex. Just as bilinear interpolation of the vertex normals generates face-interior normals for local Phong shading calculations, we can produce interpolated tangent planes perpendicular to these local normals. However, we have one additional complexity not present in the smooth shading algorithm: the *relative*

Proceedings of the 6th IEEE Visualization Conference (VISUALIZATION '95)
1070-2385/95 $10.00 © 1995 IEEE

orientations of the planes perpendicular to the normals determine the facet orientation in the maximal projection. Thus we must choose an entire *coordinate frame*, not just a normal, at each vertex; if we can choose frames that fit with their neighbors in an appealing way, we can proceed using standard quaternion spherical interpolations [18].

In dimensions higher than three, this procedure is additionally complicated by the fact that there is no longer a normal vector to a surface, but a normal subspace of dimension $N-2$.

5 Interactive Interfaces

User alignment with maximal projection. The plane in which the maximal projection is enforced can be oriented arbitrarily in the 3D graphics coordinates of the projection volume. This plane is exactly analogous to the tube $y^2+z^2=r^2$ through which a space curve is forced to pass to align it with the x-axis of a 1D screen; the flat facets of a surface are effectively drawn through a hypertube $z^2+w^2=r^2$ in 4D and thus forced into the x-y plane. But this x-y plane can be oriented with the *screen* coordinate system, in which case we are "walking" on the surface with gravity pulling us in the gaze direction, or it could be oriented with the *floor*, in which case the maximal projection is aligned with the gaze direction and we get the impression of walking on a treadmill oriented to the gravitational pull of the outside world as we sit at the computer console.

Extension to haptic interfaces. Given any 3D haptic interface, we can make the surface, in either screen or treadmill orientation, an impenetrable barrier or a cage constraining our motion to 2D. In static mode, one could keep the object fixed with the center maximally projected and allow the user to feel a limited continuous neighborhood of the center point, thus getting a tactile sense of the surface shape. In dynamic mode, one can assign a velocity and direction to the displacement of the tactile probe from the center, thus empowering the user effectively to drive around the manifold while feeling the changes in shape and slope a short distance away from the central maximal projection point.

Direct manipulation characteristics. The interface we have described is a generalization of a context-free, direct-manipulation 3D rolling-ball orientation controller (see, e.g., [9]) to arbitrary manifolds and dimensions. Properties such as non-commutativity of paths between points that are observed in this style of orientation control are transformed into the rich geometric properties and symmetry transformations of arbitrary topological spaces. In addition, just as a scene controlled by a rolling ball can move either in the same direction as the controller motion or in the opposite direction, we can traverse a manifold either way: walking forward to a new position on the manifold, or pushing the manifold in the controller direction.

Extension to three-manifolds and wands. The approach can be naturally extended to three-manifolds with winged faces, and so on. In contrast to the Maniview [7] approach, which uses an "insider's" view emphasizing discrete symmetry groups and face identifications by producing repeated copies of the manifold sewn to itself in the 3D viewing space, we would navigate specific embeddings of 3-manifolds in four or more dimensions. This approach would fit well with applications treating 3D scalar fields as 4D terrain elevation maps [12], and would be ideally adapted to 4D orientation control using three-degree-of-freedom control devices such as a wand [3, 11]. Natural applications of this technique occur not only in topology and volume visualization, but also in the Regge calculus approach to general relativity [17].

6 Examples

We conclude with several visual examples. In practice, the concepts presented are strongly dependent on motion cues and the sense of direct manipulation, and so are difficult to represent with static images.

- In Figure 8, we show a family of geodesic paths that would be taken by a walker exploring an ordinary 3D torus.
- We display in Figure 9 selected geodesic paths resulting from walking on the spun trefoil knotted sphere embedded in 4D, projected to 3D. Note how the paths follow the continuous surface in 4D despite the presence of self-intersecting surfaces in the 3D projection. The color coding on the 3D surface denotes the 4D depth from which it was projected, with blue near and red far in 4D.
- Figure 10 shows a family of walker trajectories on the one-sided projective plane embedded in 4D and projected to 3D. The chosen projection results in a surface that it is part way between a crosscap and Steiner's Roman surface. Here 4D depth is also denoted by color. One-sided surfaces present interesting logistical problems for path-tracing strategies, but the paths themselves are straightforward.

Proceedings of the 6th IEEE Visualization Conference (VISUALIZATION '95)
1070-2385/95 $10.00 © 1995 IEEE

7 Conclusion

In this paper, we have introduced a method for visualizing topological data structures by "walking" their geometry in a way that we feel has much potential for enhancing users' cognitive maps of a space. It is uniquely adapted to exploring high-dimensional mathematical data structures because of its ability to continuously flatten out the local neighborhood into the user's viewing space while maintaining a global context. For surfaces that appear self-intersecting in a particular projection, the method's locally-continuous traversal of the surface provides information interactively that is hidden even in a stereographic display. One can imagine many other applications, such as relational databases, 2D scalar fields, 3D scalar fields, and lattice models of general relativity. Furthermore, the method is ideally adaptable for newly-available haptic technologies, since it relies on local continuity that embodies an intrinsic model for force-feedback and tactile navigation.

Acknowledgments

This work was supported in part by NSF grant IRI-91-06389; addition equipment support was provided by NSF grants CDA-92-23008 and CDA 93-03189. We thank Jiping Sha for several informative conversations.

References

[1] Bryson, S. Virtual spacetime. In *Proceedings of Visualization '92* (1992), IEEE Computer Society Press, pp. 291–298.

[2] Carter, J. S. *How Surfaces Intersect in Space.* World Scientific, 1993.

[3] Cross, R. A., and Hanson, A. J. Virtual reality performance for virtual geometry. In *Proceedings of Visualization '94* (1994), IEEE Computer Society Press, pp. 156–163.

[4] Eisenhart, L. P. *A Treatise on the Differential Geometry of Curves and Surfaces.* Dover, New York, 1909 (1960).

[5] Francis, G. K. *A Topological Picturebook.* Springer Verlag, 1987.

[6] Gray, A. *Modern Differential Geometry of Curves and Surfaces.* CRC Press, Inc., Boca Raton, FL, 1993.

[7] Gunn, C. Discrete groups and visualization of three-dimensional manifolds. In *Computer Graphics* (1993), ACM, pp. 255–262. Proceedings of SIGGRAPH 1993; Annual Conference Series 1993.

[8] Hanson, A. A construction for computer visualization of certain complex curves. *Notices of the Amer.Math.Soc. 41*, 9 (November/December 1994), 1156–1163.

[9] Hanson, A. J. The rolling ball. In *Graphics Gems III*, D. Kirk, Ed. Academic Press, Cambridge, MA, 1992, pp. 51–60.

[10] Hanson, A. J. Geometry for n-dimensional graphics. In *Graphics Gems IV*, P. Heckbert, Ed. Academic Press, Cambridge, MA, 1994, pp. 149–170.

[11] Hanson, A. J. Rotations for n-dimensional graphics. In *Graphics Gems V*, A. Paeth, Ed. Academic Press, Cambridge, MA, 1995, pp. 55–64.

[12] Hanson, A. J., and Heng, P. A. Four-dimensional views of 3d scalar fields. In *Proceedings of Visualization '92* (1992), IEEE Computer Society Press, pp. 84–91.

[13] Hanson, A. J., and Heng, P. A. Illuminating the fourth dimension. *Computer Graphics and Applications 12*, 4 (July 1992), 54–62.

[14] Hanson, A. J., Munzner, T., and Francis, G. K. Interactive methods for visualizable geometry. *IEEE Computer 27*, 7 (July 1994), 73–83.

[15] Ma, H., and Hanson, A. Meshview. A portable 4D geometry viewer written in OpenGL/Motif, available by anonymous ftp from geom.umn.edu.

[16] Phillips, M., Levy, S., and Munzner, T. Geomview: An interactive geometry viewer. *Notices of the Amer. Math. Society 40*, 8 (October 1993), 985–988.

[17] Regge, T. General relativity without coordinates. *Nuovo Cimento 19* (1961), 558–571.

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

[19] Tversky, B. Spatial mental models. *The Psychology of Learning and Motivation 27* (1991), 109–145.

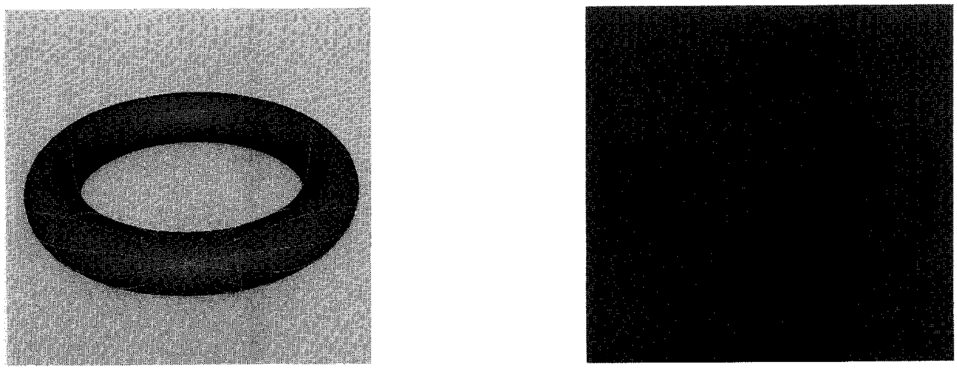

Figure 8: (a) Families of geodesic paths followed by the space walker algorithm on an ordinary torus in 3D. (b) The 3D paths on the torus with the surface removed.

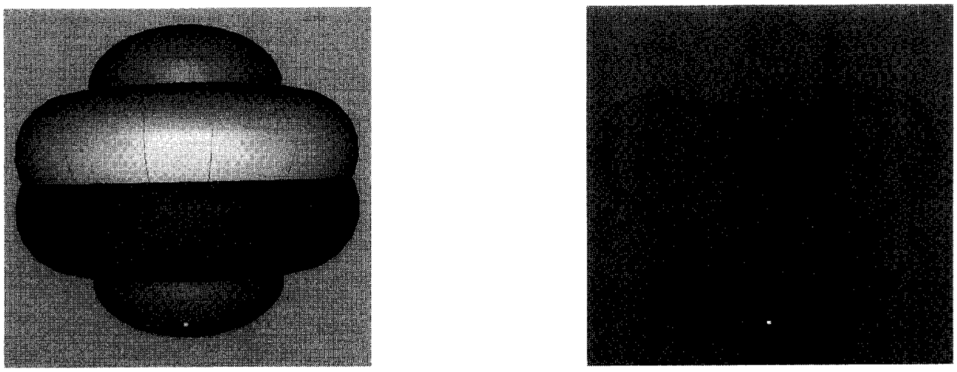

Figure 9: (a) Selected geodesic paths resulting from walking on the spun trefoil knotted sphere embedded in 4D. Note how the paths follow the continuous surface in 4D despite the presence of self-intersecting surfaces in this 3D projection. (b) The 3D paths with the surface removed.

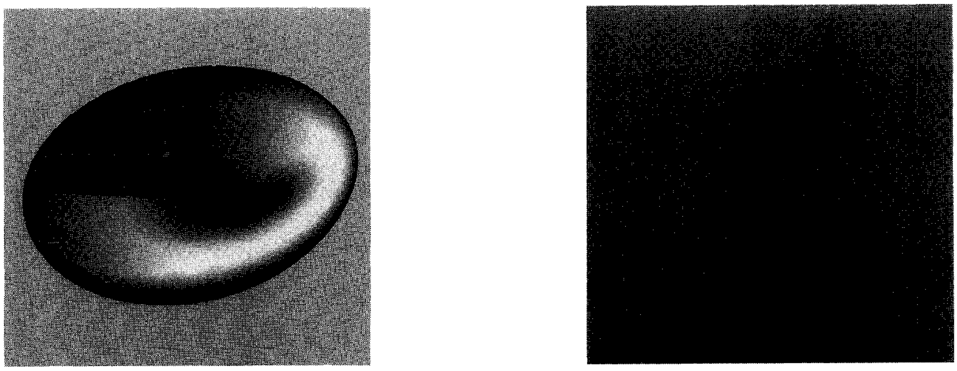

Figure 10: (a) Families of geodesic paths on a projective plane embedded in 4D and projected to 3D so that it is part way between a crosscap and Steiner's Roman surface. (b) The 3D paths on the projective plane with the surface removed.