Visualizing Quaternions

Course Notes for SIGGRAPH 2001

Course Organizer

Andrew J. Hanson
Computer Science Department
Indiana University
Bloomington, IN 47405 USA
Email: hanson@cs.indiana.edu

Abstract

This tutorial focuses on establishing an intuitive visual understanding of the relationship between ordinary 3D rotations and their quaternion representations. We begin building this intuition by showing how quaternion-like properties appear and can be exploited even in 2D space. Quaternions are then introduced in several alternative representations that do not necessarily require abstract mathematical constructs for their visualization. We then proceed to develop visualizations of quaternion applications such as moving frames and orientation splines. Finally, we briefly discuss the problem of generalizing quaternion concepts to higher dimensions using Clifford algebras.

Presenter's Biography

Andrew J. Hanson is a professor of computer science at Indiana University, and has regularly taught courses in computer graphics, computer vision, and scientific visualization. He received a BA in chemistry and physics from Harvard College in 1966 and a PhD in theoretical physics from MIT in 1971. Before coming to Indiana University, he did research in theoretical physics at the Institute for Advanced Study, Stanford, and Berkeley, and then in computer vision at the SRI Artificial Intelligence Center in Menlo Park, CA. He has published in IEEE Computer, CG&A, TVCG, ACM Computing Surveys, and has over a dozen papers in the IEEE Visualization Proceedings. He has also contributed three articles to the Graphics Gems series dealing with user interfaces for rotations and with techniques of N-dimensional geometry. Previous experience with conference tutorials includes a Siggraph '98 tutorial on N-dimensional graphics, a Visualization '98 course on Clifford Algebras and Quaternions, and a tutorial on Visualizing Quaternions presented at both Siggraph '99 and Siggraph 2000. Major research interests include scientific visualization, machine vision, computer graphics, perception, and the design of interactive user interfaces for virtual reality and visualization applications. Particular visualization applications currently being studied include an astrophysical treatment of the local galactic neighborhood of the sun, the exploitation of constrained navigation for visualization environments, and applications of graphics in dimensions greater than three to mathematics and theoretical physics.

Contents

Abstract **1**

Presenter's Biography **1**

Contents **3**

General Information **4**

1 Overview **5**

2 Fundamentals of Quaternions **5**

3 Visualizing Quaternion Geometry **5**

4 Quaternion Frames **6**

5 Clifford Algebras **6**

6 *MeshView* and Mathematica Demonstration Software **7**

References **7**

Slides: I: Fundamentals of Quaternions

Slides: II:Visualizing Quaternion Geometry

Slides: III: Quaternion Frames

Slides: IV: Clifford Algebras

Paper: "Geometry for N-dimensional Graphics," Andrew J. Hanson

Paper: "Rotations for N-dimensional Graphics," Andrew J. Hanson

Paper: "Visualizing Quaternion Rotation," J. C. Hart, G. K. Francis, and L. H. Kauffman

Paper: "Constrained Optimal Framings of Curves and Surfaces using Quaternion Gauss Maps," Andrew J. Hanson

Paper: IUCS Technical Report 518: "Quaternion Gauss Maps and Optimal Framings of Curves and Surfaces," Andrew J. Hanson

Paper: "Quaternion Frame Approach to Streamline Visualization," A.J. Hanson and H. Ma

Paper: "Meshview: Visualizing the Fourth Dimension," A.J. Hanson, K. Ishkov, and J. Ma

General Information on the Tutorial

Course Syllabus

Summary: This tutorial will deal with visualizable representations of quaternions, their features, technology, folklore, and applications. The introduction will focus on visually understanding quaternions themselves by exploiting parallels to complex variables and 2D rotations. Starting from this basis, the tutorial will proceed to give visualizations of advanced quaternion applications.

Prerequisites: Participants should be comfortable with and have an appreciation for conventional mathematical methods of 3D computer graphics and geometry used in graphics transformations and rendering. The material will be of most interest to those wishing to deepen their intuitive understanding of quaternion-based animation, moving coordinate frames, and 3D curves and surfaces appearing in graphics and scientific visualization applications.

Objectives: Participants will learn the basic facts relating quaternions to ordinary 3D rotations, as well as methods for examining the properties of quaternion constructions using interactive visualization methods. A variety of applications, including quaternion splines and moving coordinate frames for curves and surfaces, will be examined in this context. Finally, a few facts about the deeper relationship between quaternions and Clifford algebras in higher dimensions will be presented.

Outline: This is a two-hour tutorial and the material will be arranged approximately as follows:

I. *(45 min)* **Introduction to Rotation Representations.** Develop formulas and techniques for seeing how 2D rotations, orientation frames, and their time evolution equations can be visualized and studied using ordinary complex variables. Develop the parallel relationship between 3D rotations and quaternions.

II. *(45 min)* **Visualization Techniques for Quaternions.** Visualizing static and moving quaternion frames, along with quaternion splines, as 4D geometric objects.

III. *(15 min)* **Applications of Quaternion Visualization.** Extend this intuition into the quaternion representation of moving orientation frames.

IV. *(15 min)* **Clifford Algebras: the Bigger Picture.** Start to see how it all fits into Clifford algebras.

1 Overview

Practitioners of computer graphics and animation frequently represent 3D rotations using the quaternion formalism, a mathematical tool that originated with William Rowan Hamilton in the 19th century, and is now an essential part of modern analysis, group theory, differential geometry, and even quantum physics. Quaternions are in many ways very simple, and yet there are enormous subtleties to address in the process of fully understanding and exploiting their properties. The purpose of this Tutorial is to construct an intuitive bridge between our intuitions about 2D and 3D rotations and the quaternion representation.

The Tutorial will begin with an introduction to rotations in 2D, which will be found to have surprising richness, and will proceed to the construction of the relation between 3D rotations and quaternions. Quaternion visualization methods of various sorts will be introduced, followed by some applications of the quaternion frame representation to problems of interest by graphicists and visualization scientists. Finally, we will briefly touch on the relationship between Clifford algebras and quaternion rotation representations. An extensive bibliography of related literature is included, as well as several relevant reprints and technical reports, a Mathematica implementation of the Quaternion Frenet Equations, and the Meshview software system for viewing 4D objects.

2 Fundamentals of Quaternions

We will begin with a basic introduction to rotations in general, showing how 2D rotations contain the seeds for what we need to understand about 3D rotations; see, for example, [39]. We will then proceed to look at a variety of methods for understanding quaternions and making meaningful pictures of constructs involving them. These methods will range from some of the ideas introduced by Hart, Francis, and Kauffman [53] for motivating the need for double-valued parameterizations of rotations, to theoretical background given in [46, 47, 40, 50].

Traditional treatments of quaternions range from the original works of Hamilton and Tait [35, 81] to a variety of recent studies such as those of Altmann, Pletincks, Juttler, and Kuipers [2, 69, 59, 63]. The 4D frames of the quaternions themselves, in contrast to the relationship between 3D frames and quaternions, are treated in the German literature, e.g., [12, 65].

In our treatment, we will focus on the use of 2D rotations as a rich but algebraically simple proving ground in which we can see many of the key features of quaternion geometry in a very manageable context. The relationship between 3D rotations and quaternions is then introduced as a natural extension of the 2D systems.

3 Visualizing Quaternion Geometry

In order to clearly understand our options for making graphical visualizations of quaternions, we next look at the ways in which points on spheres can be viewed in reduced dimensions, discovering luckily that 3D graphics is *just* sufficient to make a usable interactive workstation system for looking at quaternions, quaternion curves, and even quaternion surfaces. The basic "trick" involves

the observation that if we have a four-vector quaternion $q = (q_0, \mathbf{q})$ obeying $q \cdot q = 1$, then the four-vector lies on the three-sphere S^3 and has only three independent components: if we display just $\mathbf{q}$, we can in principle *infer* the value of $q_0 = \sqrt{1 - \mathbf{q} \cdot \mathbf{q}}$. We supply a viewer, the Meshview system [51, 52] developed by the presenter and his students, which allows the input and interactive examination of quaternion objects.

4 Quaternion Frames

In this section, we study the nature of quaternions as representations of frames in 3D. Our visualizations again exploit the fact that quaternions are points on the three-sphere embedded in 4D; the three-sphere (S^3) is analogous to an ordinary ball or two-sphere (S^2) embedded in 3D, except that the three-sphere is a solid object instead of a surface. To manipulate, display, and visualize rotations in 3D, we may convert 3D rotations to 4D quaternion points and treat the entire problem in the framework of 4D geometry. The methods in this section follow closely techniques introduced in Hanson and Ma [46, 47] for representing families of coordinate frames on curves in 3D as curves in the 4D quaternion space. The extensions to coordinate frames on surfaces and the corresponding induced surfaces in quaternion space are studied in [40, 50].

The same methods extend to the study of quaternion animation splines, introduced to the graphics community originally by Shoemake [73]. We give an overview of the issues of constructing splines with various desirable continuity properties following the method of Schlag [71] applied to quaternion Bezier, Catmull-Rom, and uniform B-splines. These methods do not easily permit long-lasting periodic motion; this problem, as well as some problems with derivative computation, are resolved by the exponential map approach of Kim, et al. (see, e.g., [61, 32]); The exponential map and its visualization properties are therefore briefly noted as well. Alternative approaches such as the rational quaternion spline method of Jüttler [59, 60] and the variational methods of Barr et al. [10, 70] are mentioned but not treated in detail.

5 Clifford Algebras

The quaternion-based formalism for handling and visualizing rotations works well in dimensions 2, 3, and 4 because in those dimensions the Spin group, the double covering of the orthogonal group, has simple topology and geometry. Going beyond four dimensions is of course much harder. Clifford algebras form the basis used in pure mathematics to treat the Spin groups in arbitrary dimensions (see, e.g., [3, 58]); furthermore, viewed in the context of arbitrary dimensions, studying the Clifford algebra approach provides additional depth to our understanding of dimensions 2, 3, and 4 — we can get a better feeling for what properties are accidents of the low dimension and which are in fact general and extensible concepts.

6 *MeshView* and Mathematica Demonstration Software

The course CDROM should contain a copy of the MeshView 4D visualization package [51, 52], which runs under SGI-IRIX, SUN-SOLARIS, and, with some effort, Linux. This contains a sample of quaternion data sets, as well as many other 4D objects, and has facilities specifically designed for quaternion visualization experiments. There is also a Mathematica notebook `qfrmint.nb` that explicitly implements a numerical integration of the Frenet frame equations in quaternion form, vastly improving the exactly equivalent calculation for the standard Frenet equations implemented by Gray [33].

Acknowledgments

The slightly edited versions of the author's papers from Graphics Gems IV and V [38, 39] are included in the CDROM and Course Notes with the kind permission of Academic Press. Republished in the Course Notes are two key papers from IEEE Transactions on Visualization and Computer Graphics [47], and from the Proceedings of IEEE Visualization [40] of the IEEE Computer Society Press. We thank Ji-Ping Sha for his patience with Clifford algebras, and John Hart for supplying the paper by Hart, Francis, and Kauffman included here.

References

[1] B. Alpern, L. Carter, M. Grayson, and C. Pelkie. Orientation maps: Techniques for visualizing rotations (a consumer's guide). In *Proceedings of Visualization '93*, pages 183–188. IEEE Computer Society Press, 1993.

[2] S. L. Altmann. *Rotations, Quaternions, and Double Groups*. Oxford University Press, 1986.

[3] M.F. Atiyah, R. Bott, and A. Shapiro. Clifford modules. *Topology*, 3, Suppl. 1:3–38, 1986.

[4] T. Banchoff and J. Werner. *Linear Algebra through Geometry*. Springer-Verlag, 1983.

[5] T. F. Banchoff. Visualizing two-dimensional phenomena in four-dimensional space: A computer graphics approach. In E. Wegman and D. Priest, editors, *Statistical Image Processing and Computer Graphics*, pages 187–202. Marcel Dekker, Inc., New York, 1986.

[6] Thomas F. Banchoff. *Beyond the Third Dimension: Geometry, Computer Graphics, and Higher Dimensions*. Scientific American Library, New York, NY, 1990.

[7] David Banks. Interactive display and manipulation of two-dimensional surfaces in four dimensional space. In *Symposium on Interactive 3D Graphics*, pages 197–207, New York, 1992. ACM.

[8] David Banks. Interactive manipulation and display of two-dimensional surfaces in four-dimensional space. In David Zeltzer, editor, *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, volume 25, pages 197–207, March 1992.

[9] David C. Banks. Illumination in diverse codimensions. In *Computer Graphics*, pages 327–334, New York, 1994. ACM. Proceedings of SIGGRAPH 1994; Annual Conference Series 1994.

[10] 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.

[11] Richard L. Bishop. There is more than one way to frame a curve. *Amer. Math. Monthly*, 82(3):246–251, March 1975.

[12] Wilhelm Blaschke. *Kinematik und Quaternionen*. VEB Deutscher Verlag der Wissenschaften, Berlin, 1960.

[13] Jules Bloomenthal. Calculation of reference frames along a space curve. In Andrew Glassner, editor, *Graphics Gems*, pages 567–571. Academic Press, Cambridge, MA, 1990.

[14] Kenneth A. Brakke. The surface evolver. *Experimental Mathematics*, 1(2):141–165, 1992. The "Evolver" system, manual, and sample data files are available by anonymous ftp from geom.umn.edu, The Geometry Center, Minneapolis MN.

[15] D. W. Brisson, editor. *Hypergraphics: Visualizing Complex Relationships in Art, Science and Technology*, volume 24. Westview Press, 1978.

[16] S. A. Carey, R. P. Burton, and D. M. Campbell. Shades of a higher dimension. *Computer Graphics World*, pages 93–94, October 1987.

[17] Michael Chen, S. Joy Mountford, and Abigail Sellen. A study in interactive 3-d rotation using 2-d control devices. In *Proceedings of Siggraph 88*, volume 22, pages 121–130, 1988.

[18] H.S.M. Coxeter. *Regular Complex Polytopes*. Cambridge University Press, second edition, 1991.

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

[20] A.R. Edmonds. *Angular Momentum in Quantum Mechanics*. Princeton University Press, Princeton, New Jersey, 1957.

[21] N.V. Efimov and E.R. Rozendorn. *Linear Algebra and Multi-Dimensional Geometry*. Mir Publishers, Moscow, 1975.

[22] T. Eguchi, P. B. Gilkey, and A. J. Hanson. Gravitation, gauge theories and differential geometry. *Physics Reports*, 66(6):213–393, December 1980.

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

[24] S. Feiner and C. Beshers. Visualizing n-dimensional virtual worlds with n-vision. *Computer Graphics*, 24(2):37–38, March 1990.

[25] S. Feiner and C. Beshers. Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds. In *Proceedings of UIST '90, Snowbird, Utah*, pages 76–83, October 1990.

[26] Gerd Fischer. *Mathematische Modelle*, volume I and II. Friedr. Vieweg & Sohn, Braunschweig/Wiesbaden, 1986.

[27] H. Flanders. *Differential Forms*. Academic Press, New York, 1963.

[28] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes. *Computer Graphics, Principles and Practice*. Addison-Wesley, second edition, 1990. page 227.

[29] A. R. Forsyth. *Geometry of Four Dimensions*. Cambridge University Press, 1930.

[30] George K. Francis. *A Topological Picturebook*. Springer Verlag, 1987.

[31] Herbert Goldstein. *Classical Mechanics*. Addison-Wesley, 1950.

[32] F.S. Grassia. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools*, 3(3):29–48, 1998.

[33] Alfred Gray. *Modern Differential Geometry of Curves and Surfaces*. CRC Press, Inc., Boca Raton, FL, second edition, 1998.

[34] Cindy M. Grimm and John F. Hughes. Modeling surfaces with arbitrary topology using manifolds. In *Computer Graphics Proceedings, Annual Conference Series*, pages 359–368, 1995. Proceedings of SIGGRAPH '95.

[35] W.R. Hamilton. *Lectures on Quaternions*. Cambridge University Press, 1853.

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

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

[38] A. J. Hanson. Geometry for n-dimensional graphics. In Paul Heckbert, editor, *Graphics Gems IV*, pages 149–170. Academic Press, Cambridge, MA, 1994.

[39] A. J. Hanson. Rotations for n-dimensional graphics. In Alan Paeth, editor, *Graphics Gems V*, pages 55–64. Academic Press, Cambridge, MA, 1995.

[40] A. J. Hanson. Constrained optimal framings of curves and surfaces using quaternion gauss maps. In *Proceedings of Visualization '98*, pages 375–382. IEEE Computer Society Press, 1998.

[41] A. J. Hanson and R. A. Cross. Interactive visualization methods for four dimensions. In *Proceedings of Visualization '93*, pages 196–203. IEEE Computer Society Press, 1993.

[42] A. J. Hanson and P. A. Heng. Visualizing the fourth dimension using geometry and light. In *Proceedings of Visualization '91*, pages 321–328. IEEE Computer Society Press, 1991.

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

[44] A. J. Hanson and P. A. Heng. Foursight. In *Siggraph Video Review*, volume 85. ACM Siggraph, 1992. Scene 11, Presented in the Animation Screening Room at SIGGRAPH '92, Chicago, Illinois, July 28–31, 1992.

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

[46] A. J. Hanson and H. Ma. Visualizing flow with quaternion frames. In *Proceedings of Visualization '94*, pages 108–115. IEEE Computer Society Press, 1994.

[47] A. J. Hanson and H. Ma. Quaternion frame approach to streamline visualization. *IEEE Trans. on Visualiz. and Comp. Graphics*, 1(2):164–174, June 1995.

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

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

[50] A.J. Hanson. Quaternion gauss maps and optimal framings of curves and surfaces. Indiana University Computer Science Department Technical Report 518 (October, 1998).

[51] A.J. Hanson, K. Ishkov, and J. Ma. Meshview. A portable 4D geometry viewer written in OpenGL/Motif, available by anonymous ftp from ftp.cs.indiana.edu:pub/hanson.

[52] A.J. Hanson, K. Ishkov, and J. Ma. Meshview: Visualizing the fourth dimension. Overview of the MeshView 4D geometry viewer.

[53] John C. Hart, George K. Francis, and Louis H. Kauffman. Visualizing quaternion rotation. *ACM Trans. on Graphics*, 13(3):256–276, 1994.

[54] D. Hilbert and S. Cohn-Vossen. *Geometry and the Imagination*. Chelsea, New York, 1952.

[55] John G. Hocking and Gail S. Young. *Topology*. Addison-Wesley, 1961.

[56] C. Hoffmann and J. Zhou. Some techniques for visualizing surfaces in four-dimensional space. *Computer-Aided Design*, 23:83–91, 1991.

[57] S. Hollasch. Four-space visualization of 4D objects. Master's thesis, Arizona State University, August 1991.

[58] H.B. Lawson Jr. and M.L. Michelsohn. *Spin Geometry*. Princeton University Press, 1989.

[59] B. Jüttler. Visualization of moving objects using dual quaternion curves. *Computers and Graphics*, 18(3):315–326, 1994.

[60] B. Jüttler and M.G. Wagner. Computer-aided design with saptial rational B-spline motions. *Journal of Mechanical Design*, 118:193–201, June 1996.

[61] Myoung-Jun Kim, Myung-Soo Kim, and Sung Yong 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.

[62] F.. Klock. Two moving coordinate frames for sweeping along a 3d trajectory. *Computer Aided Geometric Design*, 3, 1986.

[63] J.B. Kuipers. *Quaternions and Rotation Sequences*. Princeton University Press, 1999.

[64] J. Milnor. *Topology from the Differentiable Viewpoint*. The University Press of Virginia, Charlottesville, 1965.

[65] Hans Robert Müller. *Sphärische Kinematik*. VEB Deutscher Verlag der Wissenschaften, Berlin, 1962.

[66] 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.

[67] Michael A. Noll. A computer technique for displaying n-dimensional hyperobjects. *Communications of the ACM*, 10(8):469–473, August 1967.

[68] Mark Phillips, Silvio Levy, and Tamara Munzner. Geomview: An interactive geometry viewer. *Notices of the Amer. Math. Society*, 40(8):985–988, October 1993. Available by anonymous ftp from geom.umn.edu, The Geometry Center, Minneapolis MN.

[69] D. Pletincks. Quaternion calculus as a basic tool in computer graphics. *The Visual Computer*, 5(1):2–13, 1989.

[70] Ravi Ramamoorthi and Alan H. Barr. Fast construction of accurate quaternion splines. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 287–292. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

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

[72] Uri Shani and Dana H. Ballard. Splines as embeddings for generalized cylinders. *Computer Vision, Graphics, and Image Processing*, 27:129–156, 1984.

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

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

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

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

[77] D.M.Y. Sommerville. *An Introduction to the Geometry of N Dimensions*. Reprinted by Dover Press, 1958.

[78] N. Steenrod. *The Topology of Fibre Bundles*. Princeton University Press, 1951. Princeton Mathematical Series 14.

[79] K. V. Steiner and R. P. Burton. Hidden volumes: The 4th dimension. *Computer Graphics World*, pages 71–74, February 1987.

[80] D. J. Struik. *Lectures on Classical Differential Geometry*. Addison-Wesley, 1961.

[81] P.G. Tait. *An Elementary Treatise on Quaternions*. Cambridge University Press, 1890.

[82] J. R. Weeks. *The Shape of Space*. Marcel Dekker, New York, 1985.

[83] S. Weinberg. *Gravitation and Cosmology: Principles and Applications of General Relativity*. John Wiley and Sons, 1972.

[84] E.T. Whittaker. *A Treatise on the Analytical Dynamics of Particles and Rigid Bodies*. Dover, New York, New York, 1944.

Visualizing Quaternions

Andrew J. Hanson
Computer Science Department
Indiana University

Siggraph 2001 Tutorial

GRAND PLAN

I: Fundamentals of Quaternions

II: Visualizing Quaternion Geometry

III: Quaternion Frames

IV: Clifford Algebras

I: Fundamentals of Quaternions

- **Motivation**
- **2D Frames:** Simple example, complex numbers.
- **3D Frames:** Rotations and quaternions.

II: Visualizing Quaternion Geometry

- **The Spherical Projection Trick:** Visualizing unit vectors.
- **Quaternion Frames**
- **Quaternion Curves**
- **Quaternion Splines**

III: Quaternion Frames

- **Quaternion Curves:** generalize the Frenet Frame
- **Quaternion Frame Evolution**
- **Quaternion Curve and Surface Optimization**

IV: Clifford Algebras

- **Clifford Algebras:** Generalize quaternion structure to N-dimensions
- **Reflections and Rotations:** New ways of looking at rotations
- **Pin(N), Spin(N), O(N), and SO(N)**

Visualizing Quaternions

Part I: Fundamentals of Quaternions

Andrew J. Hanson
Indiana University

Part I: OUTLINE

- **Motivation**
- **2D Frames:** Simple example, complex numbers.
- **3D Frames:** Rotations and quaternions.

Motivation

- Quaternion methods are now commonplace in graphics.
- Quaternions are very geometric, but we seldom attempt to visualize their properties geometrically.
- That's going to be our job today!

Basic Issues

- Basic fact number 1: Rotation matrices are **Coordinate Frame Axes.**
- Basic fact number 2: Rotation matrices form *groups*, which have *geometric properties*.

Task vs Strategy

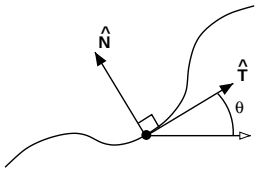
- **Our task: Understand Rotations**.
- Rotations don't just **act** on geometry, rotations **are** geometry.
- **Our strategy: the geometry should help us** to visualize the properties of rotations.

Simple Example: 2D Rotations

- 2D rotations $\Rightarrow$ **geometric origin** for *complex numbers.*
- **Complex numbers** are a special subspace of quaternions.
- Thus 2D rotations **introduce us to quaternions** and their geometric meaning.

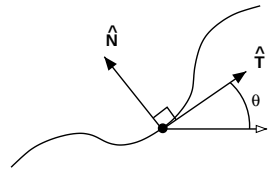
Frames in 2D

The tangent and normal to 2D curve move continuously along the curve:



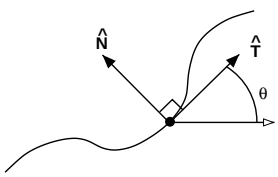
Frames in 2D

The tangent and normal to 2D curve move continuously along the curve:



Frames in 2D

The tangent and normal to 2D curve move continuously along the curve:



Frame Matrix in 2D

This motion is described at each point (or time) by the matrix:

$$R_2(\theta) = \begin{bmatrix} \hat{\mathbf{T}} & \hat{\mathbf{N}} \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} .$$

Another 2D Frame

If we did not know about $\cos^2\theta + \sin^2\theta = 1$, we might represent the frame differently, e.g., as:

$$R_2(A, B) = \begin{bmatrix} A & -B \\ B & A \end{bmatrix} .$$

with the constraint $A^2 + B^2 = 1$.

The Belt Trick:

Is There a Problem?

Demonstration: Rotations "want to be doubled" to get back where you started.

See: Hart, Francis, and Kauffman.

Half-Angle Transform:

A Fix for the Problem?

$$R_2(\theta) = \begin{bmatrix} \cos^2\frac{\theta}{2} - \sin^2\frac{\theta}{2} & -2\cos\frac{\theta}{2}\sin\frac{\theta}{2} \\ 2\cos\frac{\theta}{2}\sin\frac{\theta}{2} & \cos^2\frac{\theta}{2} - \sin^2\frac{\theta}{2} \end{bmatrix}$$

Half-Angle Transform:

A Fix for the Problem?

Or, with $a = \cos(\theta/2)$, $b = \sin(\theta/2)$,
(i.e., $A = a^2 - b^2$, $B = 2ab$),
we could parameterize as:

$$R_2(a,b) \;=\; \begin{bmatrix} a^2 - b^2 & -2ab \\ 2ab & a^2 - b^2 \end{bmatrix}.$$

where orthonormality implies

$$(a^2 + b^2)^2 = 1$$

which reduces back to $a^2 + b^2 = 1$.

Half-Angle Transform:

So the pair (a, b) provides an odd **double-valued** parameterization of the frame:

$$\begin{bmatrix} \hat{\mathbf{T}} & \hat{\mathbf{N}} \end{bmatrix} \;=\; \begin{bmatrix} a^2 - b^2 & -2ab \\ 2ab & a^2 - b^2 \end{bmatrix}.$$

where $(a,\, b)$ is precisely the *same frame* as $(-a,\, -b)$.

Frame Evolution in 2D

Examine time-evolution of 2D frame (on our way to 3D): First in $\theta(t)$ coordinates:

$$\begin{bmatrix} \hat{\mathbf{T}} & \hat{\mathbf{N}} \end{bmatrix} \;=\; \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

Differentiate to find frame equations:

$$\begin{aligned} \dot{\hat{\mathbf{T}}}(t) &= +\kappa\hat{\mathbf{N}} \\ \dot{\hat{\mathbf{N}}}(t) &= -\kappa\hat{\mathbf{T}}\,, \end{aligned}$$

where $\kappa(t) = d\theta/dt$ is the **curvature**.

Frame Evolution in 2D

Rearrange to make a "vector matrix:"

$$\begin{bmatrix} \dot{\hat{\mathbf{T}}}(t) \\ \dot{\hat{\mathbf{N}}}(t) \end{bmatrix} = \begin{bmatrix} 0 & +\kappa(t) \\ -\kappa(t) & 0 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{T}}(t) \\ \hat{\mathbf{N}}(t) \end{bmatrix}$$

Frame Evolution in (a, b):

Using the basis $(\hat{\mathbf{T}}, \hat{\mathbf{N}})$ we have **Four equations** with **Three constraints** from orthonormality, for **One** true degree of freedom.

Major Simplification occurs in (a, b) coordinates!!

$$\dot{\hat{\mathbf{T}}} = 2\begin{bmatrix} a\dot{a} - b\dot{b} \\ a\dot{b} + b\dot{a} \end{bmatrix} = 2\begin{bmatrix} a & -b \\ b & a \end{bmatrix}\begin{bmatrix} \dot{a} \\ \dot{b} \end{bmatrix}$$

Frame Evolution in (a, b):

But this formula for $\dot{\mathbf{T}}$ is just $\kappa\widehat{\mathbf{N}}$, where

$$\kappa\widehat{\mathbf{N}} = \kappa \begin{bmatrix} -2ab \\ a^2 - b^2 \end{bmatrix} = \kappa \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \begin{bmatrix} -b \\ a \end{bmatrix}$$

or

$$\kappa\widehat{\mathbf{N}} = \kappa \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

2D Quaternion Frames!

Rearranging terms, *both* $\dot{\mathbf{T}}$ and $\dot{\mathbf{N}}$ eqns reduce to

$$\begin{bmatrix} \dot{a} \\ \dot{b} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -\kappa \\ +\kappa & 0 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \end{bmatrix}$$

This is the square root *of frame equations.*

2D Quaternions . . .

So *one equation* in the two *"quaternion"* variables (a, b) with the constraint $a^2 + b^2 = 1$ contains *both* the frame equations

$$\dot{\mathbf{T}} = +\kappa\widehat{\mathbf{N}}$$

$$\dot{\mathbf{N}} = -\kappa\widehat{\mathbf{T}}$$

$\Rightarrow$ this is much better for computer implementation, etc.

Rotation as Complex Multiplication

If we let $(a + ib) = \exp(i\,\theta/2)$ we see that rotation is complex multiplication!

"Quaternion Frames" in 2D are just complex numbers, with

Evolution Eqns = derivative of $\exp(i\,\theta/2)$!

Rotation with no matrices!

This is the miracle:

$$a + ib = e^{i\theta/2}$$

represents rotations "more nicely" than the matrices $R(\theta)$.

$$(a' + ib')(a + ib) = e^{i(\theta' + \theta)/2} = A + iB$$

where if we *want* the matrix, we write:

$$R(\theta')R(\theta) = R(\theta' + \theta) = \begin{bmatrix} A^2 - B^2 & -2AB \\ 2AB & A^2 - B^2 \end{bmatrix}$$

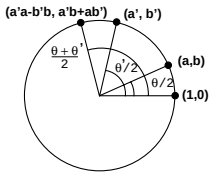
The Algebra of 2D Rotations

The algebra corresponding to 2D rotations is easy: just complex multiplication!!

$$\begin{aligned} (a', b') * (a, b) &\cong (a' + ib')(a + ib) \\ &= a'a - b'b + i(a'b + ab') \\ &\cong (a'a - b'b,\ a'b + ab') \\ &= (A,\ B) \end{aligned}$$

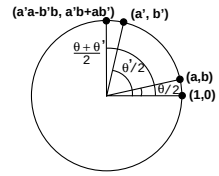
The Geometry of 2D Rotations

(a, b) with $a^2 + b^2 = 1$ is a **point on the unit circle**, also written S^1. Rotations are just **complex multiplication**, and take you around the unit circle like this:



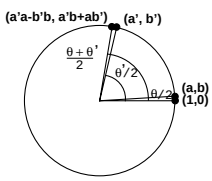
The Geometry of 2D Rotations

(a, b) with $a^2 + b^2 = 1$ is a **point on the unit circle**, also written S^1. Rotations are just **complex multiplication**, and take you around the unit circle like this:



The Geometry of 2D Rotations

(a, b) with $a^2 + b^2 = 1$ is a **point on the unit circle**, also written S^1. Rotations are just **complex multiplication**, and take you around the unit circle like this:



Quaternion Frames

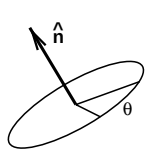
In 3D, repeat our trick: take square root of the frame, but now use *quaternions*:

- **Write down the 3D frame.**
- **Write as double-valued quadratic form.**
- **Rewrite linearly in the new variables.**

The Geometry of 3D Rotations

We begin with a basic fact:

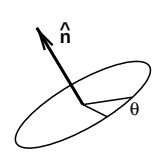
Euler theorem: *every* 3D frame can be written as a spinning by θ about a fixed axis $\hat{\mathbf{n}}$, the eigenvector of the rotation matrix:



The Geometry of 3D Rotations

We begin with a basic fact:

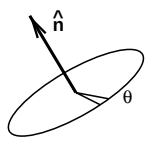
Euler theorem: *every* 3D frame can be written as a spinning by θ about a fixed axis $\hat{\mathbf{n}}$, the eigenvector of the rotation matrix:



The Geometry of 3D Rotations

We begin with a basic fact:

Euler theorem: *every* 3D frame can be written as a spinning by θ about a fixed axis $\hat{\mathbf{n}}$, the eigenvector of the rotation matrix:



Quaternion Frames . . .

Matrix giving 3D rotation by θ about axis $\hat{\mathbf{n}}$:

$$R_3(\theta, \hat{\mathbf{n}}) =$$

$$\begin{bmatrix} c + (n_1)^2(1-c) & n_1 n_2(1-c) - s n_3 & n_3 n_1(1-c) + s n_2 \\ n_1 n_2(1-c) + s n_3 & c + (n_2)^2(1-c) & n_3 n_2(1-c) - s n_1 \\ n_1 n_3(1-c) - s n_2 & n_2 n_3(1-c) + s n_1 & c + (n_3)^2(1-c) \end{bmatrix}$$

where $c = \cos\theta$, $s = \sin\theta$, and $\hat{\mathbf{n}} \cdot \hat{\mathbf{n}} = 1$.

Quaternion Frame Parameters

To find θ and axis $\hat{\mathbf{n}}$, given *any* rotation matrix or frame M, we need two steps:

$$\mathrm{Tr} M = 1 + 2\cos\theta$$

$\Rightarrow$ solve for θ.

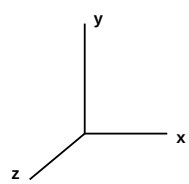
$$M - M^t =$$

$$\begin{bmatrix} 0 & -2n_3 \sin\theta & +2n_2 \sin\theta \\ +2n_3 \sin\theta & 0 & -2n_1 \sin\theta \\ -2n_2 \sin\theta & +2n_1 \sin\theta & 0 \end{bmatrix}$$

$\Rightarrow$ solve for $\hat{\mathbf{n}}$ as long as $\theta \neq 0$.

Quaternions and Rotations

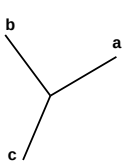
Some set of axes can be chosen as the identity matrix:



$$= \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

Quaternions and Rotations

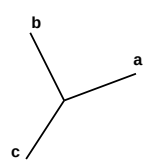
Any arbitrary set of axes forms the columns of an orthogonal rotation matrix:



$$= \begin{vmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{vmatrix}$$

Quaternions and Rotations

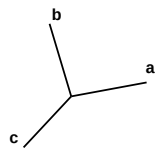
Any arbitrary set of axes forms the columns of an orthogonal rotation matrix:



$$= \begin{vmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{vmatrix}$$

Quaternions and Rotations

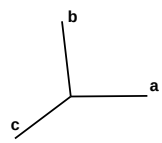
Any arbitrary set of axes forms the columns of an orthogonal rotation matrix:



$$= \begin{vmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{vmatrix}$$

Quaternions and Rotations

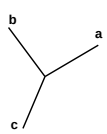
Any arbitrary set of axes forms the columns of an orthogonal rotation matrix:



$$= \begin{vmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{vmatrix}$$

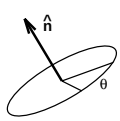
Quaternions and Rotations

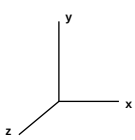
By Euler's theorem, that matrix has an *eigenvector* $\hat{\mathbf{n}}$, and so is representable as a single rotation about $\hat{\mathbf{n}}$ applied to the identity:



$$= \begin{vmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{vmatrix}$$

$$= \quad \times \quad = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$





Rotations and Quadratic Polynomials

Remember $R_2(\theta) = \begin{bmatrix} a^2 - b^2 & -2ab \\ 2ab & a^2 - b^2 \end{bmatrix}$?

What if we try a 3×3 matrix R_3 instead of 2×2?

$$\begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

Hint: set $q_1 = q_2 = 0$ or any other $(i \neq j)$ pair to see a familiar sight!

Quaternions and Rotations

Why does this matrix parameterize a rotation? Because *Columns* of $R_3(q_0, q_1, q_2, q_3)$ are orthogonal:

$$\mathrm{col}_i \cdot \mathrm{col}_j = 0 \text{ for } i \neq j$$

What is LENGTH of 3-vector column?

$$\mathrm{col}_i \cdot \mathrm{col}_i = (q_0^2 + q_1^2 + q_2^2 + q_3^2)^2$$

Quaternions and Rotations . . .

So if we require $\boxed{q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1}$, orthonormality is assured and $R_3(q_0, q_1, q_2, q_3)$ is a rotation.

This implies q **is a point on 3-sphere in 4D**.

NOTE: $q \Rightarrow -q$ gives *same* $R_3()$.

Quaternions and Rotations . . .

HOW does $q = (q_0, \mathbf{q})$ represent rotations?

LOOK at

$$R_3(\theta, \hat{\mathbf{n}}) \stackrel{?}{=} R_3(q_0, q_1, q_2, q_3)$$

NOTICE: Choosing

$$q(\theta, \hat{\mathbf{n}}) = (\cos\frac{\theta}{2},\ \hat{\mathbf{n}}\sin\frac{\theta}{2})$$

makes the R_3 equation an *IDENTITY*.

Quaternions and Rotations . . .

WHAT happens if you do **TWO** rotations?

EXAMINE the action of two rotations

$$R(q')R(q) = R(Q)$$

EXPRESS in quadratic forms in q and LOOK FOR an analog of complex multiplication:

Quaternions and Rotations . . .

RESULT: the following multiplication rule $q' * q = Q$ yields **exactly** the correct 3×3 rotation matrix $R(Q)$:

$$\begin{bmatrix} Q_0 = [q' * q]_0 \\ Q_1 = [q' * q]_1 \\ Q_2 = [q' * q]_2 \\ Q_3 = [q' * q]_3 \end{bmatrix} = \begin{bmatrix} q'_0 q_0 - q'_1 q_1 - q'_2 q_2 - q'_3 q_3 \\ q'_0 q_1 + q'_1 q_0 + q'_2 q_3 - q'_3 q_2 \\ q'_0 q_2 + q'_2 q_0 + q'_3 q_1 - q'_1 q_3 \\ q'_0 q_3 + q'_3 q_0 + q'_1 q_2 - q'_2 q_1 \end{bmatrix}$$

This is Quaternion Multiplication.

Algebra of Quaternions = 3D Rotations!

2D rotation matrices are represented by **complex multiplication**

3D rotation matrices are represented by **quaternion multiplication**

Algebraic 2D/3D Rotations

Therefore in 3D, the 2D complex multiplication

$$(a', b') * (a, b) = (a'a - b'b,\ a'b + ab')$$

is replaced by 4D quaternion multiplication:

$$\begin{aligned} q' * q \ = \ & (q'_0 q_0 - q'_1 q_1 - q'_2 q_2 - q'_3 q_3, \\ & q'_0 q_1 + q'_1 q_0 + q'_2 q_3 - q'_3 q_2, \\ & q'_0 q_2 + q'_2 q_0 + q'_3 q_1 - q'_1 q_3, \\ & q'_0 q_3 + q'_3 q_0 + q'_1 q_2 - q'_2 q_1) \end{aligned}$$

Algebra of Quaternions . . .

The is easier to remember by dividing it into the *scalar* piece q_0 and the *vector* piece $\vec{\mathbf{q}}$:

$$\begin{aligned} q' * q \ = \ & (q'_0 q_0 - \vec{\mathbf{q}'} \cdot \vec{\mathbf{q}}, \\ & q'_0 \vec{\mathbf{q}} + q_0 \vec{\mathbf{q}'} + \vec{\mathbf{q}'} \times \vec{\mathbf{q}}) \end{aligned}$$

Quaternions and Rotations

Another miracle: let us generalize the 2D equation

$$a + ib = e^{i\theta/2}$$

How? We set

$$\begin{aligned} q &= (q_0, q_1, q_2, q_3) \\ &= q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3 \\ &= e^{(\mathbf{I}\cdot\hat{\mathbf{n}}\theta/2)} \end{aligned}$$

with $q_0 = \cos(\theta/2)$ and $\vec{\mathbf{q}} = \hat{\mathbf{n}}\sin(\theta/2)$ and $\mathbf{I} = (\mathbf{i}, \mathbf{j}, \mathbf{k})$.

Quaternions and Rotations . . .

Then if we take $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1$, and $\mathbf{i} * \mathbf{j} = \mathbf{k}$ (cyclic), quaternion multiplication rule is automatic!

$\Rightarrow$ $q = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3$ is the standard representation for a *quaternion*, and we can also use 2×2 Pauli matrices in place of $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ if we want.

Key to Quaternion Intuition

Fundamental Intuition: We know

$$q_0 = \cos(\theta/2), \quad \vec{\mathbf{q}} = \hat{\mathbf{n}}\sin(\theta/2)$$

We also know that *any coordinate frame* M can be written as $M = R(\theta, \hat{\mathbf{n}})$.

Therefore

$\vec{\mathbf{q}}$ points exactly along the axis we have to rotate around to go from identity I to M, and the length of $\vec{\mathbf{q}}$ tells us how much to rotate.

Summarize Quaternion Properties

- **Unit four-vector.** Take $q = (q_0, q_1, q_2, q_3) = (q_0, \vec{\mathbf{q}})$ to obey constraint $q \cdot q = 1$.

- **Multiplication rule.** The quaternion product q and p is $q * p = (q_0 p_0 - \vec{\mathbf{q}} \cdot \vec{\mathbf{p}},\ q_0\vec{\mathbf{p}} + p_0\vec{\mathbf{q}} + \vec{\mathbf{q}} \times \vec{\mathbf{p}})$, or, alternatively,

$$\begin{bmatrix} [q*p]_0 \\ [q*p]_1 \\ [q*p]_2 \\ [q*p]_3 \end{bmatrix} = \begin{bmatrix} q_0p_0 - q_1p_1 - q_2p_2 - q_3p_3 \\ q_0p_1 + q_1p_0 + q_2p_3 - q_3p_2 \\ q_0p_2 + q_2p_0 + q_3p_1 - q_1p_3 \\ q_0p_3 + q_3p_0 + q_1p_2 - q_2p_1 \end{bmatrix}$$

Quaternion Summary . . .

- **Rotation Correspondence.** The unit quaternions q and $-q$ correspond to a single 3D rotation R_3:

$$\begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

Quaternion Summary . . .

- **Rotation Correspondence.** Let

$$q = (\cos\frac{\theta}{2}, \hat{\mathbf{n}}\sin\frac{\theta}{2}),$$

with $\hat{\mathbf{n}}$ a unit 3-vector, $\hat{\mathbf{n}} \cdot \hat{\mathbf{n}} = 1$. Then $R(\theta, \hat{\mathbf{n}})$ is usual 3D rotation by θ in the plane $\perp$ to $\hat{\mathbf{n}}$.

- **Inversion.** Any 3×3 matrix R can be inverted for q *up to a sign*. Carefully treat singularities! Can choose sign, e.g., by local consistency, to get continuous frames.

SUMMARY

- **Complex numbers** represent 2D frames.
- **Complex multiplication represents 2D rotation.**
- **Quaternions** represent 3D frames.
- **Quaternion multiplication represents 3D rotation.**

Visualizing Quaternions

Part II: Visualizing Quaternion Geometry

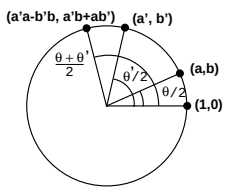
Andrew J. Hanson
Indiana University

Part II: OUTLINE

- **The Spherical Projection Trick:** Visualizing unit vectors.
- **Quaternion Frames**
- **Quaternion Curves**
- **Quaternion Splines**

The Geometry of Quaternions

Recall (a, b) with $a^2 + b^2 = 1$ is a unit-length **complex number** or a **point on the unit circle** S^1 .

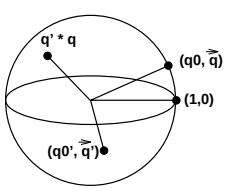


The Geometry of Quaternions . . .

Similarly, $q = (q_0, \vec{q})$ with ${q_0}^2 + \vec{q}^2 = 1$ is a unit-length **quaternion** or a **point on the unit 3-sphere** S^3.

The Geometry of Quaternions . . .

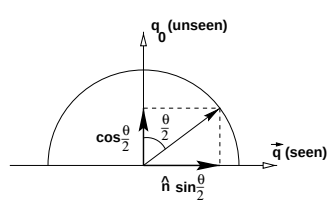
Rotations combine by taking the quaternion product of the *geometric* values of 4D points on S^3:



Visualizing a Quaternion??

Learn how to *Visualize* a quaternion by starting with a visualization of a point on S^1**,** the circle:

$$q_0 = \sqrt{1 - (q_1)^2}$$

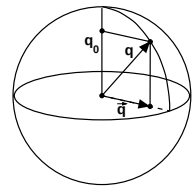


Demo: 2D Unit Vector Projection

Visualizing a Quaternion?? . . .

Next, visualize a point on S^2, the ordinary sphere using only the projection $\vec{\mathrm{q}}$:

$$q_0 = \sqrt{1 - (q_1)^2 - (q_2)^2}$$

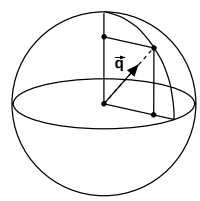


Demo: 3D Unit Vector Projection

Visualizing a Quaternion?? . . .

Finally, visualize a point on S^3, the quaternion space: **DISPLAY** only $\vec{\mathrm{q}}$, but **INFER**

$$q_0 = \sqrt{1 - (q_1)^2 - (q_2)^2 - (q_3)^2}$$



Demo: 4D Unit Vector Projection

Visualize Quaternion Rotations

Each 4D quaternion point $q = (\cos\frac{\theta}{2}, \hat{\mathrm{n}}\sin\frac{\theta}{2})$ is a frame — a 3×3 rotation matrix generated by applying $R(\theta, \hat{\mathrm{n}})$ to the identity frame.

Identity Matrix is the quaternion $q = (1, 0, 0, 0)$.

Visualize q using only the VECTOR part $\vec{\mathrm{q}}$, so Identity is the zero vector.

Visualize Quaternion Rotations . . .

The quaternion rotation by θ about $\hat{\mathrm{n}}$:

$$q = (q_0, \mathrm{q}) = (\cos(\theta/2), \hat{\mathrm{n}}\sin(\theta/2))$$

represents the matrix $R(\theta, \hat{\mathrm{n}})$.

Action of rotating Identity by θ about $\hat{\mathrm{n}}$:
$q * (1, 0, 0, 0)$ gives Vector part:

$$\vec{0} \Rightarrow \hat{\mathrm{n}}\sin(\theta/2)$$

Quaternion Interpretation

Rotation axis is Fundamental 3-vector: We know

$$q_0 = \cos(\theta/2), \quad \vec{\mathrm{q}} = \hat{\mathrm{n}}\sin(\theta/2)$$

We also know that *any coordinate frame* M can be written as $M = R(\theta, \hat{\mathrm{n}})$.

Therefore

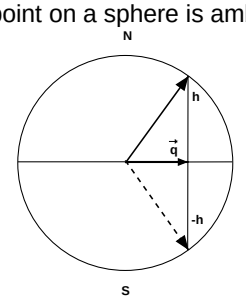
$\vec{\mathrm{q}}$ points exactly along the axis we have to rotate around to go from identity I to M, and the length of $\vec{\mathrm{q}}$ tells us how much to rotate.

Demo: Growing Quaternion = Rotation about Axis

$\hat{\mathrm{q}}$ = required Euler theorem axis.

Displaying spherical points

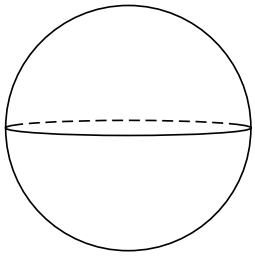
Displaying a point on a sphere is ambiguous:



The same horizontal projection is shared by the North vector $(h, \vec{\mathrm{q}})$ and the South vector $(-h, \vec{\mathrm{q}})$.

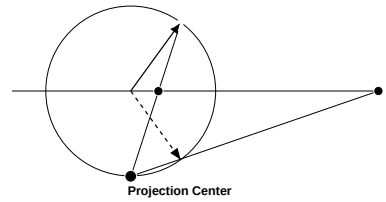
Orbit is Front/Back Line

2D projection of orbit around sphere has two half circles that line up in 2D, match at two edge points only in 3D.



Projective spherical points

Polar Projection removes ambiguity, destroys symmetry:



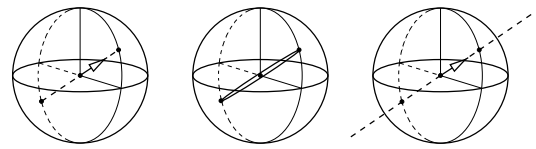
the North vector $(h, \vec{\mathrm{q}})$ and the South vector $(-h, \vec{\mathrm{q}})$ project to *different points* on the line.

Displaying S^3

A quaternion point can be displayed in

- **Parallel Projection:** so $q = (h, \vec{\mathrm{q}})$ lines up with $q = (-h, \vec{\mathrm{q}})$,
- **Polar projection:** so only the "north pole" projects within the unit sphere, and "south pole" is at ∞ of R^3.

Displaying S^3 . . .



(a) (b) (c)

(a) Usual vector quaternion point. (b) Orbits through northern and southern hemispheres. (c) Polar projection: north pole at origin, south pole at infinity.

Quaternion Curves

Represent Families of Frames

Each orientation is a *4D point* on the 3-sphere representing a quaternion.

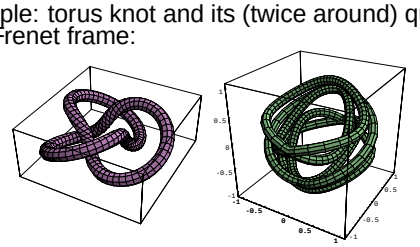
Thus families of frames, which are really rotation matrices, become **curves** on the 3-sphere.

$\Rightarrow$ treat these curves just like any other curve. . .

Demo: Simple Quaternion Curves

Families of Quaternion Frames, . . .

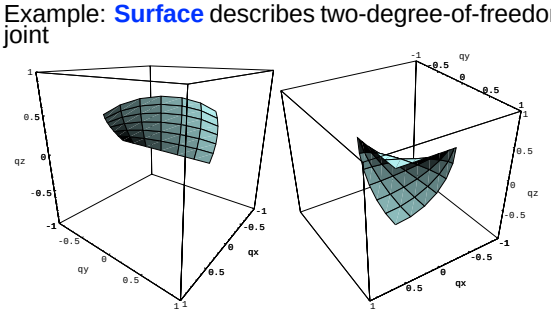
Example: torus knot and its (twice around) quaternion Frenet frame:



see: Hanson and Ma, "Quaternion Frame Approach to Streamline Visualization," *IEEE Trans. on Visualiz. and Comp. Graphics*, **1**, No. 2, pp. 164–174 (June, 1995).

Families of Quaternion Frames, . . .

Example: **Surface** describes two-degree-of-freedom joint



Remark: Group theory *demands* that x and y rotations combine to give some z rotation!

Quaternion Interpolations

- Shoemake (Siggraph '85) proposed using quaternions instead of Euler angles to get smooth frame interpolations without **Gimbal Lock**:

  BEST CHOICE: Animate objects and cameras using rotations represented on S^3 by quaternions

Interpolating on Spheres

N-dimensional spherical interpolation employs the "SLERP," a constant angular velocity transition between two directions, $\hat{\mathbf{n}}_1$ and $\hat{\mathbf{n}}_2$:

$$\begin{aligned}\hat{\mathbf{n}}_{12}(t) &= \text{Slerp}(\hat{\mathbf{n}}_1, \hat{\mathbf{n}}_2, t) \\ &= \hat{\mathbf{n}}_1 \frac{\sin((1-t)\theta)}{\sin(\theta)} + \hat{\mathbf{n}}_2 \frac{\sin(t\theta)}{\sin(\theta)}\end{aligned}$$

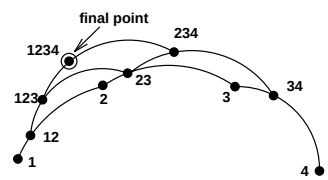
where $\cos\theta = \hat{\mathbf{n}}_1 \cdot \hat{\mathbf{n}}_2$.

Remark on Spherical Interpolation

(The SLERP formula is simply the result of applying a **Gram-Schmidt decomposition** to the two vectors while enforcing unit norm in *any dimension*.)

Quaternion Interpolations

Many variations have been proposed since then; simplest is simply to apply the formula iteratively to give analog of the de Casteljau spline construction:



Spline Families

Schlag (in Graphics Gems II (1991)) gives recursive form for several cubic splines:

$$\begin{aligned}&S(x_1, x_2, x_3, x_4, t) = \\ &L(L(L(x_1, x_2, f_{12}(t)), \quad L(x_2, x_3, f_{23}(t)), \quad f_{123}(t)), \\ &\quad L(L(x_2, x_3, f_{23}(t)), \quad L(x_3, x_4, f_{34}(t)), \quad f_{234}(t)), \\ &\qquad f(t))\end{aligned}$$

Spline Families . . .

For **Euclidean space**, the interpolator is

$$L(a, b, t) = a(1-t) + bt$$

while for **Spherical space**, the interpolator is

$$L(a, b, t) = a\frac{\sin((1-t)\theta)}{\sin\theta} + b\frac{\sin(t\theta)}{\sin\theta}$$

where $a \cdot b = \cos\theta$.

Spline Families . . .

Catmull-Rom

$f_{12} = t + 1$ $\quad f_{23} = t$ $\quad f_{34} = t - 1$

$f_{123} = \frac{(t+1)}{2}$ $\quad f_{234} = \frac{t}{2}$

$f = t$

Spline Families . . .

Bezier

$f_{12} = t$ $\quad f_{23} = t$ $\quad f_{34} = t$

$f_{123} = t$ $\quad f_{234} = t$

$f = t$

Spline Families . . .

Uniform B-spline

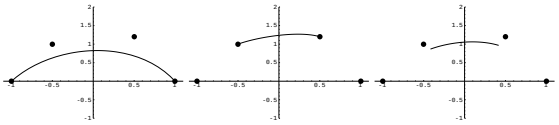
$f_{12} = \frac{(t+2)}{3}$ $\quad f_{23} = \frac{(t+1)}{3}$ $\quad f_{34} = \frac{t}{3}$

$f_{123} = \frac{(t+1)}{2}$ $\quad f_{234} = \frac{t}{2}$

$f = t$

Plane Interpolations

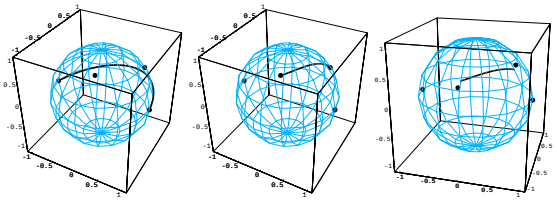
In Euclidean space, these three basic cubic splines look like this:



Bezier Catmull-Rom Uniform B

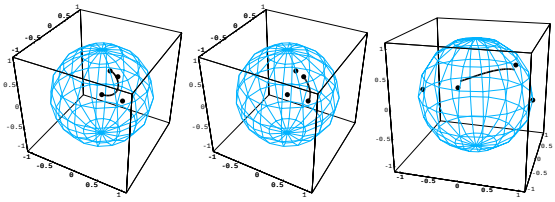
The differences are in the derivatives: Bezier has to start matching all over at every fourth point; Catmull-Rom matches the first derivative; and B-spline is the cadillac, matching all derivatives but no *control points*.

Spherical Interpolations



Bezier Catmull-Rom Uniform B

Quaternion Interpolations



Bezier Catmull-Rom Uniform B

Demo: Interpolated Keyframe Quaternion Curves

Optional:

Exponential Quaternion Map

Exponential map (Kim, Kim, and Shin; see also Grassia) has these advantages:

- **Derivative Computation Simpler.** Initial velocity conditions work.
- **Derivative Matching Simpler.** Easier to match neighboring derivatives.
- **No Renormalization Req'd.**
  4 vars, 3 DOF $\Rightarrow$ 3 independent vars, constraints automatic.

Exponential Quaternion Map . . .

METHOD: use only three "vector" degrees of freedom $\vec{\mathbf{n}} = \hat{\mathbf{n}}\,\theta/2$: exponentiating gives four-component quaternion obeying unit 3-sphere constraint:

$$\begin{aligned} q(\vec{\mathbf{n}}) &= e^{(\mathbf{I}\,\hat{\mathbf{n}}\,\theta/2)} \\ &= (\cos(\theta/2),\ \hat{\mathbf{n}}\sin(\theta/2)) \end{aligned}$$

Note (Grassia): the quaternion frame evolution equations can be then be written *directly* in terms of the 3-vector $\vec{\mathbf{n}}$!

Exponential Quaternion Map . . .

Then, using *logarithms* of quaternion control points, e.g.,

$$\vec{\mathbf{n}} = \log q = \left(\frac{\theta}{2}\right)\frac{\mathbf{q}}{|\mathbf{q}|} = \cos^{-1}(q_0)\frac{\mathbf{q}}{|\mathbf{q}|}$$

where $\vec{\mathbf{n}}$ is *not* normalized, we can use splines in the reduced *Euclidean* 3D space.

Exponential Quaternion Map . . .

The trick:

Wherever you would see $x_0 + t(x_1 - x_0)$ in a Euclidean spline, replace $(x_1 - x_0)$ by

$$\omega = \log((q_0)^{-1} * q_1)$$

(This is easy: quaternion multiplication gives a new $Q = (Q_0, \mathbf{Q})$, and $\omega = \log Q$ is computed as usual.)

Exponential Quaternion Map . . .

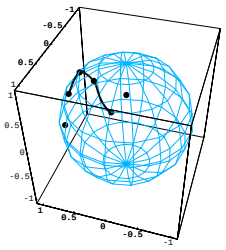
Then, with $\omega_i = \log((q_{i-1})^{-1} * q_i)$, let

$$q(t) = q_0 \prod_{i=1}^{k} \exp(\omega_i \alpha_i(t))$$

$\Rightarrow$ Euclidean spline $\alpha_i(t)$ methods can be used, and derivative computations are simplified.

Exponential Quaternion Map . . .

Example of a many-control-point Catmull-Rom spline done using the exponential method:



Even More Quaternion Interpolations . . .

A number of other approaches can be found in the bibliography. Other literature includes:

- **Barr et al.** Global optimization emulating vanishing 4th derivative of Euclidean cubic splines, but on quaternion three-sphere.
- **Jüttler et al.:** Generalized rational splines, including quaternion component as well as a spatial component.

SUMMARY

- **The Spherical Projection Trick:** Visualizing unit vectors.
- **Quaternion Frames:** $\hat{\mathbf{n}}$ in quaternion tells how to make frame.
- **Quaternion Curves:** are like any other curve; spline applications.

Visualizing Quaternions

Part III: Quaternion Frames

Andrew J. Hanson
Indiana University

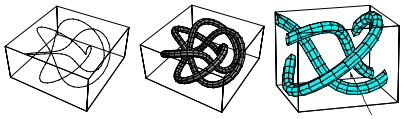
Part III: OUTLINE

- **Quaternion Curves:** generalize the Frenet Frame
- **Quaternion Frame Evolution**
- **Quaternion Curve and Surface Optimization**

What are Frames used For?

- Move objects and object parts in an animated scene.
- Move the camera generating the rendered viewpoint of the scene.
- Attach tubes and textures to thickened lines, oriented textures to surfaces.
- Compare shapes of similar curves.

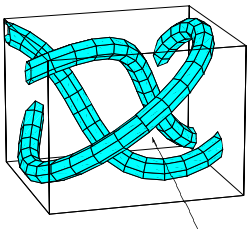
Motivating Problem: Framing Curves



The (3,5) torus knot.

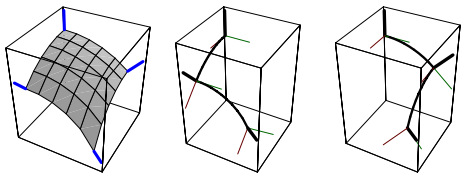
- Line drawing $\approx$ useless.
- Tubing based on parallel transport, **not periodic.**
- Closeup of the non-periodic mismatch.

Motivating Problems: Curves



Closeup of the non-periodic mismatch.
Can't apply texture.

Motivating Problems: Surfaces



A smooth 3D surface patch: two ways to get bottom frame.

No unique orthonormal frame is derivable from the parameterization.

3D Curves: Frenet and PT Frames

Now give more details of 3D frames: Classic Moving Frame:

$$\begin{bmatrix}\mathbf{T}'(t)\\ \mathbf{N}'(t)\\ \mathbf{B}'(t)\end{bmatrix} = \begin{bmatrix}0 & k_1(t) & k_2(t)\\ -k_1(t) & 0 & \sigma(t)\\ -k_2(t) & -\sigma(t) & 0\end{bmatrix}\begin{bmatrix}\mathbf{T}(t)\\ \mathbf{N}(t)\\ \mathbf{B}(t)\end{bmatrix} .$$

Serret-Frenet frame: $k_2 = 0$, $k_1 = \kappa(t)$ is the curvature, and $\sigma(t) = \tau(t)$ is the classical torsion. LOCAL.

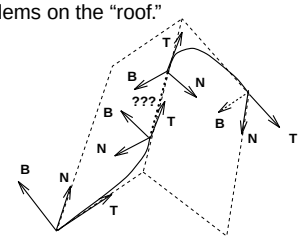
Parallel Transport frame (Bishop): $\sigma = 0$ to get minimal turning. NON-LOCAL = an INTEGRAL.

3D curve frames, contd

Frenet frame is *locally* defined, e.g., by

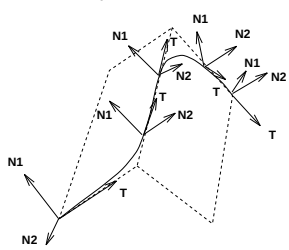
$$\mathbf{B}(t) = \frac{\mathbf{x}'(t) \times \mathbf{x}''(t)}{\|\mathbf{x}'(t) \times \mathbf{x}''(t)\|}$$

but has problems on the "roof."

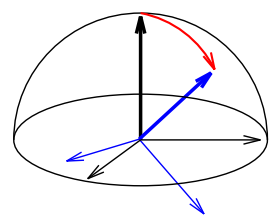


3D curve frames, contd

Bishop's Parallel Transport frame is *integrated over whole curve*, non-local, but no problems on "roof:"

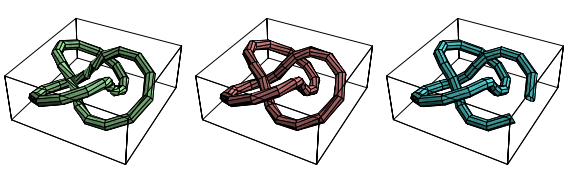


3D curve frames, contd



Geodesic Reference Frame is the frame found by tilting North Pole of "canonical frame" along a great circle until it points in desired direction (tangent for curves, normal for surfaces).

Sample Curve Tubings and their Frames



Tubings based on Frenet, Geodesic Reference, and Parallel Transport frames.
Easily see PT has least "Twist," but lacks periodicity.

3D Frames to Quaternion Frames

- **Unit four-vector.** Take $q = (q_0, q_1, q_2, q_3) = (q_0, \mathbf{q})$ to obey constraint $q \cdot q = 1$.
- **Multiplication rule.** Let $q * p$ be the quaternion product of two quaternions q and p, where

$$\begin{bmatrix}[q*p]_0\\ [q*p]_1\\ [q*p]_2\\ [q*p]_3\end{bmatrix} = \begin{bmatrix}q_0p_0 - q_1p_1 - q_2p_2 - q_3p_3\\ q_0p_1 + q_1p_0 + q_2p_3 - q_3p_2\\ q_0p_2 + q_2p_0 + q_3p_1 - q_1p_3\\ q_0p_3 + q_3p_0 + q_1p_2 - q_2p_1\end{bmatrix}$$

. . . to Quaternion Frames . . .

- **Quaternion Correspondence.** The unit quaternions q and $-q$ correspond to a single 3D rotation $R_3(q)$:

$$\begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2q_1q_2-2q_0q_3 & 2q_1q_3+2q_0q_2 \\ 2q_1q_2+2q_0q_3 & q_0^2-q_1^2+q_2^2-q_3^2 & 2q_2q_3-2q_0q_1 \\ 2q_1q_3-2q_0q_2 & 2q_2q_3+2q_0q_1 & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix}$$

- **Rotation Correspondence.**
Let $q = (\cos\frac{\theta}{2}, \hat{\mathbf{n}}\sin\frac{\theta}{2})$, with $\hat{\mathbf{n}}$ a unit 3-vector, $\hat{\mathbf{n}}\cdot\hat{\mathbf{n}} = 1$. Then $R(\theta, \hat{\mathbf{n}})$ is usual 3D rotation by θ in the plane perpendicular to $\hat{\mathbf{n}}$.

Quaternion Frame Evolution

Just as in 2D, let columns of $R_3(q)$ be a **frame**: $(\mathbf{T}, \mathbf{N}, \mathbf{B})$; this system has *nine components*.

Derivatives of the i-th column R_i in quaternion coordinates have the form:

$\dot{R}_i = 2W_i \cdot [\dot{q}(t)]$ where $i = 1, 2, 3$ and, e.g.,

$$W_1 = \begin{bmatrix} q_0 & q_1 & -q_2 & -q_3 \\ q_3 & q_2 & q_1 & q_0 \\ -q_2 & q_3 & -q_0 & q_1 \end{bmatrix}$$

(rows form mutually orthonormal basis).

Quaternion Frame Evolution . . .

When we simplify by eliminating W_i . . .
we find the *square root* of the 3D frame eqns!

Tait (1890) derived the quaternion equation that makes **all 9 3D frame equations reduce to:** $\boxed{\dot{q} = (1/2)q * (0, \mathrm{k})}$ or:

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 0 & k_2 & -k_1 & -\sigma \\ -k_2 & 0 & \sigma & -k_1 \\ k_1 & -\sigma & 0 & -k_2 \\ \sigma & k_1 & k_2 & 0 \end{bmatrix} \cdot \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

Quaternion Frames . . .

Properties of Tait's quaternion frame equations:

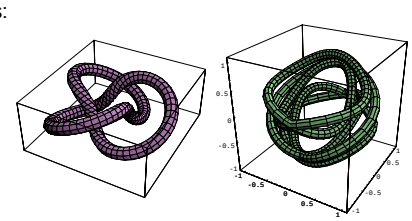
- Antisymmetry $\Rightarrow$ $q(t) \cdot \dot{q}(t) = 0$ as required to keep constant unit radius on 3-sphere.
- *Nine equations and six constraints* become *four equations and one constraint*, keeping quaternion on the 3-sphere. $\Rightarrow$ **Good for computer implementation.**
- MATHEMATICA code implementing this differential equation is provided.

Quaternion Frames . . .

- Analogous treatment (given in Hanson Tech Note in Course Notes) applies also to the Weingarten equations, allowing a direct quaternion treatment of the classical differential geometry of *surfaces* as well.

Example of a Quaternion Frame Curve

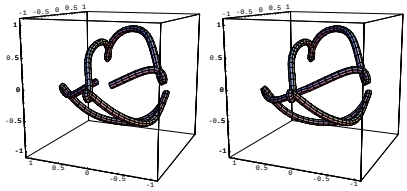
Left Curve = torus knot tubed with Frenet frame; Right Curve is projection from 4D of (twice around) quaternion Frenet frames:



see Notes: Hanson and Ma, "Quaternion Frame Approach to Streamline Visualization," *IEEE Trans. on Visualiz. and Comp. Graphics*, **1**, No. 2, pp. 164–174 (June, 1995).

Minimizing Quaternion Length Solves Periodic Tube

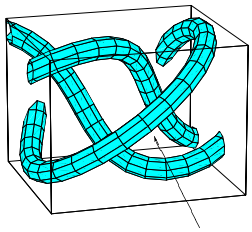
Quaternion space optimization of the non-periodic parallel transport frame of the (3,5) torus knot.



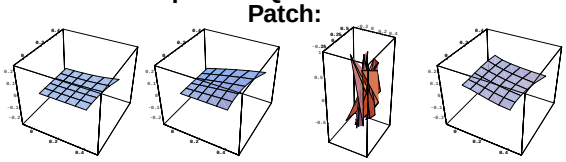
see Notes: "Constrained Optimal Framings of Curves and Surfaces using Quaternion Gauss Maps," *Proceedings of Visualization '98*, pp. 375–382 (1998).

Minimizing Quaternion Length Works

Result of Quaternion space optimization of the (3,5) torus knot frame.



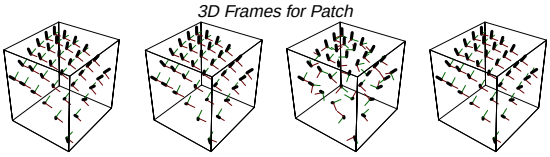
Can also Optimize Quaternion Frames on Patch:



(a) (b) (c) (d)

Quaternion frames for (a) Geodesic Ref. (b) One edge Parallel Transport. (c) Random. (d) Minimal area result.

3D Frames for Patch



(a) (b) (c) (d)

Quaternion frames for (a) Geodesic Ref. (b) One edge Parallel Transport. (c) Random. (d) Minimal area result.

SUMMARY

- **Quaternions can represent frames.**
- **Curve frames ⇒ quaternion curves.**
- **Surface patch frames ⇒ quaternion surface patches.**
- **Minimizing quaternion length or area finds parallel transport "minimal turning" set of frames.**

Visualizing Quaternions

Part IV: Clifford Algebras

Andrew J. Hanson
Indiana University

Part IV: OUTLINE

- **Clifford Algebra:** Introduction to generalizing complex numbers and quaternions.
- **Reflections vs Rotations:** How to make a rotation.
- **Pin(N), Spin(N), O(N), and SO(N):** Double coverings of N-dimensional rotations.

Motivation

- **Quaternions are too special.** Complex numbers and quaternions run out of steam after dimensions 2,3,4.
- **Search for some generalizable idea.** How does the $(a^2 - b^2)$, etc., form generalize to N-dim rotation matrices?
- **Clifford Algebra:** Clifford found the generalization, but the *really* interesting relation to spin 1/2 elementary particles came much later.

Foundations

- N-dimensional space is described formally by basis vectors e_i, $i = 1, \ldots, N$ and real numbers v_i, so a vector looks like

$$V = \sum_i v_i e_i$$

- The *length* is computed from the inner product:

$$\|V\|^2 = < V, V > = \sum_{ij} v_i g_{ij} v_j$$

where g_{ij} is the identity matrix in Euclidean space.

Foundations . . .

- But the basis vectors obey a strange multiplication rule:

$$e_i e_j + e_j e_i = -2 g_{ij}$$

- This is the CLIFFORD ALGEBRA.
- (Note: physicists would recognize these formulas as those obeyed by the *Pauli matrices* or the *Dirac matrices*.)

Clifford Algebra . . .

- How does this odd product concern us??

1. Because it contains *in any dimension* a way of expressing rotations as *multiple reflections about a plane*.
2. Because these expressions of rotations are natural *square roots* of the familiar $N \times N$ orthogonal matrix approach to writing rotations.

Clifford Algebra implements reflections

If $A = \Sigma a_i e_i$ is any vector with $\|A\| = 1$, then, using the Clifford multiplication rule,

$$V' = A * V * A = V - 2A < A, V >$$

This is just a *reflection* of the component of V lying in the direction of A about the *plane*

$$< A, X >= 0$$

Clifford Algebra – sample calculation

With $A = \Sigma a_i e_i$ and $\|A\| = 1$, compute V':

$$\begin{aligned} A * V * A &= \sum_{i,j,k} a_i v_j a_k e_i e_j e_k \\ &= \sum_{i,j,k} a_i v_j a_k \left(-e_j e_i - 2g_{ij}\right) e_k \\ &= \sum_{i,j,k} a_i v_j a_k \left(-e_j \{e_i e_k\} - 2g_{ij} e_k\right) \\ &= -V \sum_{i,j} a_i a_k \frac{1}{2}(e_i e_k + e_k e_i) - 2A < A, V > \end{aligned}$$

Clifford Algebra – sample calculation, contd

Use *symmetrization trick* — only one-half of the symmetrized sum multiplying $\Sigma a_i a_k$ can survive; that in turn is just the Clifford rule right hand side!

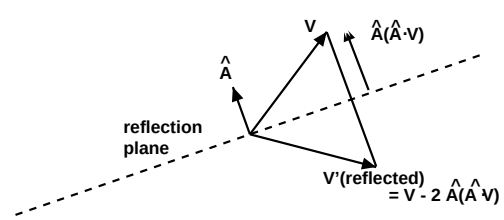
$$\begin{aligned} A * V * A &= -V \sum_{i,j} a_i a_k \frac{1}{2}(-2g_{ik}) - 2A < A, V > \\ &= +V < A, A > -2A < A, V > \end{aligned}$$

QED.

Note: The MINUS sign in $-2g_{ij}$ in the original definition makes the sign of V positive.

Clifford Algebra reflections

$$V' = A * V * A = V - 2A < A, V >$$



Clifford Algebra rotations

Let $B = \Sigma b_i e_i$ be *another* vector with $\|B\| = 1$:

Repeating the Clifford multiplication rule,

$$\begin{aligned} V'' &= B * V' * B \\ &= B * A * V * A * B \\ &= V' - 2B < B, V' > \\ &= V - 2A < A, V > \\ -2B < B, V > &+ 4B < B, A >< A, V > \end{aligned}$$

Clifford Algebra rotations

This can be shown (e.g, in Mathematica) to be a *proper rotation* of the vector V, that is

$$V'' = B * A * V * A * B = \sum_{ij} R_{ij} v_j e_i$$

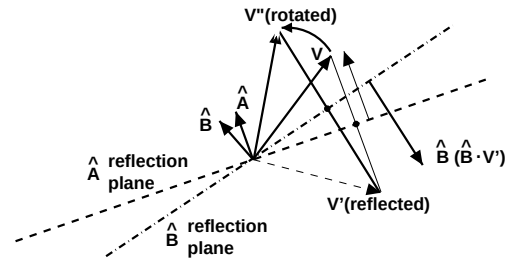
where R_{ij} is an orthonormal matrix of unit determinant.

Clifford Algebra rotations

Graphically, we have this:

$$\begin{aligned} V'' &= B * A * V * A * B \\ &= \sum_{ij} R_{ij} v_j e_i \end{aligned}$$

Clifford Algebra rotation graph



Clifford Algebra rotations

NOTE that in higher dimensions, you may need *more* than a single (A, B) pair to exhaust all possible rotations:

N	pairs	params	constraints	freedom
2	1	$2 * 2$	3	1
3	1	$2 * 3$	3	3
4	2	$4 * 4$	10	6
5	2	$4 * 5$	10	10
6	3	$6 * 6$	21	15
7	3	$6 * 7$	21	21
.	.	.	.	.
N	$p = \lfloor \frac{N}{2} \rfloor$	$2p * N$	$p(2p+1)$	$\frac{N(N-1)}{2}$

Examples of Clifford Algebras

- $N = 1$: The basis $(1, e_1)$ with $(e_1)^2 = -1$ is just the **complex numbers.** But be careful: there is only one dimension, so the only possible reflection is $x \to -x$. This is *not* enough to do 2D rotations!

Examples of Clifford Algebras . . .

- $N = 2$: The basis $(1, e_1, e_2, e_1 e_2)$ exhausts all possible Clifford products.
  Since $e_1 e_2 e_1 e_2 = -1$, we can identify this basis with the **quaternions** $(1, \mathrm{i}, \mathrm{j}, \mathrm{k})$! But be careful: there are only *two* dimensions, so this is *not* enough to do 3D rotations!
- **Where is 2D Rot?** *True* basis of rotations is the **even part** of the family of all Clifford products, or $\boxed{(1, e_1 e_2)}$!!

2D Rotations done right

- **What is i?** What we *called* $i = \sqrt{-1}$ is *really* $\boxed{i = e_1 e_2}$
- **How do we rotate in 2D?** Let $R = a + b e_1 e_2, \quad R^\dagger = a - b e_1 e_2$:

$$\boxed{R * V * R^\dagger = V' = v_1' e_1 + v_2' e_2}$$

where V' now means a rotation, and

$$\begin{bmatrix} v_1' \\ v_2' \end{bmatrix} = \begin{bmatrix} a^2 - b^2 & -2ab \\ 2ab & a^2 - b^2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

2D Rotations done right . . .

So the half-angle formula is mandatory! Our 2D transformation was not so silly after all; nothing else generalizes to N-dimensions. The Clifford algebra for $N = 2$ automatically produces:

$$R_2(a, b) = \begin{bmatrix} a^2 - b^2 & -2ab \\ 2ab & a^2 - b^2 \end{bmatrix},$$

where $a^2 + b^2 = 1$, and we have the solution $a = \cos(\theta/2), \; b = \sin(\theta/2)$.

3D Rotations done right

3D is of course a little trickier: here the full basis of all Clifford products is 8-dimensional:

$$\boxed{(1, e_1, e_2, e_3, e_2 e_3, e_3 e_1, e_1 e_2, e_1 e_2 e_3)}$$

- **Even part is rotations.** To exclude reflections, we keep only the even part:

$$\boxed{(1, e_2 e_3, e_3 e_1, e_1 e_2)}$$

- **These are the quaternions:** identify these with $(1, \mathrm{i}, \mathrm{j}, \mathrm{k})$.

3D Rotations done right . . .

- **General 3D Rotation:** with

$$R = q_0 + q_1 e_2 e_3 + q_2 e_3 e_1 + q_3 e_1 e_2$$

we have

$$R * V * R^\dagger = \sum_{i=1}^{4} v_i' e_i$$

where the coefficients of v_i in v_i' are *precisely* our old quaternion formula.

Higher dimensions:

As one might expect, higher dimensions are *much* more complicated, and do not work out so neatly, except for a convenient accident in $N = 4$, which allows a "double-quaternion" form (see "Rotations for N-dimensional Graphics").

Higher dimensions:

But we can do a little counting to see what is going on in N dimensions, where we know that the number of rotational degrees of freedom is

$$\boxed{N(N-1)/2}$$

Degrees of freedom in higher dimensional Spin representations:

N	Dim(even Clifford)	Dim(Rotations)	Constraints (the difference)
1	1	0	1
2	2	1	1
3	4	3	1
4	8	6	2
5	16	10	6
6	32	15	17
7	64	21	43
8	128	28	100
.	.	.	.
N	2^{N-1}	$N(N-1)/2$	$\frac{2^N - N^2 + N}{2}$

Pin(N), Spin(N), O(N), SO(N) and all that . . .

Spin representations of orthogonal groups follow from the Clifford Algebra Cl(N): (So do **spinors** — but another time...)

- **Pin(N).** G is "Pin" if it's a general reflection; G includes **all** elements of Cl(N).
- **Spin(N).** G is "Spin" if it's a general rotation; G is only **even** elements of Cl(N).
- **O(N).** $G * V * G^\dagger$ is "O" if G is in Pin and result is a vector reflection.
- **SO(N).** $G * V * G^\dagger$ is "SO" if G is in Spin and result is a vector rotation.

GRAND CONCLUSION

- **Rotation Matrices:** can be represented by a "square root" object with simpler geometric properties than rotations (= quaternions for $N = 2, 3, 4$).
- **Visualization of Quaternions:** is possible using sphere projection trick and a solid unit sphere.
- **Quaternion Curves, Surfaces, Volumes:** embedded in that sphere represent animations, flows, curve tubings, joint ranges, etc.
- **Clifford Algebras:** form the rigorous basis for the whole set of concepts.

◊ II.6

Geometry for N-Dimensional Graphics

Andrew J. Hanson
Computer Science Department
Indiana University
Bloomington, IN 47405
hanson@cs.indiana.edu

◊ Introduction ◊

Textbook graphics treatments commonly use special notations for the geometry of 2 and 3 dimensions that are not obviously generalizable to higher dimensions. Here we collect a family of geometric formulas frequently used in graphics that are easily extendible to N dimensions as well as being helpful alternatives to standard 2D and 3D notations.

What use are such formulas? In mathematical visualization, which commonly must deal with higher dimensions — 4 real dimensions, 2 complex dimensions, etc. — the utility is self-evident (see, e.g., (Banchoff 1990, Francis 1987, Hanson and Heng 1992b, Phillips et al. 1993)). The visualization of statistical data also frequently utilizes techniques of N-dimensional display (see, e.g., (Noll 1967, Feiner and Beshers 1990a, Feiner and Beshers 1990b, Brun et al. 1989, Hanson and Heng 1992a)). We hope that publicizing some of the basic techniques will encourage further exploitation of N-dimensional graphics in scientific visualization problems.

We classify the formulas we present into the following categories: basic notation and the N-simplex; rotation formulas; imaging in N-dimensions; N-dimensional hyperplanes and volumes; N-dimensional cross-products and normals; clipping formulas; the point-hyperplane distance; barycentric coordinates and parametric hyperplanes; N-dimensional ray-tracing methods. An appendix collects a set of obscure Levi-Civita symbol techniques for computing with determinants. For additional details and insights, we refer the reader to classic sources such as (Sommerville 1958, Coxeter 1991, Hocking and Young 1961) and (Banchoff and Werner 1983, Efimov and Rozendorn 1975).

◊ Definitions — What is a Simplex, Anyway? ◊

In a nutshell, an N-simplex is a set of $(N+1)$ points that together specify the simplest non-vanishing N-dimensional volume element (e.g., two points delimit a line segment in 1D, 3 points a triangle in 2D, 4 points a tetrahedron in 3D, etc.). From a mathematical point of view,

Copyright © 1994 by Academic Press, Inc.
All rights of reproduction in any form reserved.
ISBN 0-12-336156-7

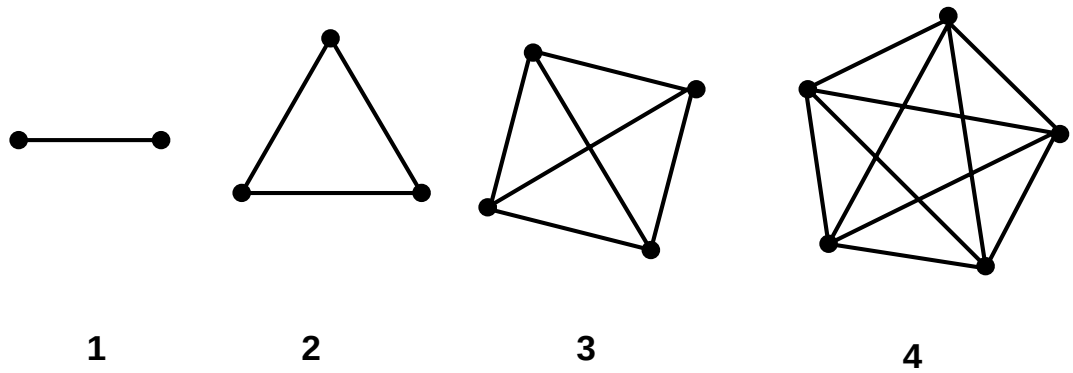


Figure 1. 2D projections of simplexes with dimension 1–4. An N-simplex is defined by $(N+1)$ linearly independent points and generalizes the concept of a line segment or a triangular surface patch.

there are lots of different N-dimensional spaces: here we will restrict ourselves to ordinary flat, real Euclidean spaces of N dimensions with global orthogonal coordinates that we can write as

$$\vec{x} = (x, y, z, \ldots, w)$$

or more pedantically as

$$\vec{x} = (x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(N)}) \ .$$

We will use the first, less cumbersome, notation whenever it seems clearer.

Our first type of object in N-dimensions, the 0-dimensional *point* $\vec{x}$, may be thought of as a vector from the origin to the designated set of coordinate values. The next type of object is the 1-dimensional *line*, which is determined by giving two points $(\vec{x}_0, \vec{x}_1)$; the line segment from $\vec{x}_0$ to $\vec{x}_1$ is called a 1-*simplex*. If we now take three noncollinear points $(\vec{x}_0, \vec{x}_1, \vec{x}_2)$, these uniquely specify a *plane*; the triangular area delineated by these points is a 2-*simplex*. A 3-simplex is a solid tetrahedron formed by a set of four noncoplanar points, and so on. In figure 1, we show schematic diagrams of the first few simplexes projected to 2D.

Starting with the $(N+1)$ points $(\vec{x}_0, \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N)$ defining a simplex, one then connects all possible pairs of points to form edges, all possible triples to form faces, and so on, resulting in the structure of component "parts" given in table 1. The next higher object uses its predecessor as a building block: a triangular face is built from three edges, a tetrahedron is built from four triangular faces, a 4-simplex is built from 5 tetrahedra.

The general idea should now be clear: $(N+1)$ linearly independent points define a *hyperplane* of dimension N and specify the boundaries of an N-dimensional coordinate patch comprising an N-*simplex* (Hocking and Young 1961). Just as the surfaces modeling a 3D object may be broken up (or *tessellated*) into triangular patches, N-dimensional objects may be tessellated into $(N-1)$-dimensional simplexes that define their geometry.

Table 1. Numbers of component structures making up an N-simplex. For example, in 2D, the basic simplex is the triangle with 3 points, 3 edges, and one 2D face.

	Dimension of Space					
Type of Simplex	$N=1$	$N=2$	$N=3$	$N=4$	$\ldots$	N
Points (0D)	2	3	4	5	$\ldots$	$\binom{N+1}{1} = N+1$
Edges (1D simplex)	1	3	6	10	$\ldots$	$\binom{N+1}{2}$
Faces (2D simplex)	0	1	4	10	$\ldots$	$\binom{N+1}{3}$
Volumes (3D simplex)		0	1	5	$\ldots$	$\binom{N+1}{4}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$(N-2)$D simplex					$\ldots$	$\binom{N+1}{N-1}$
$(N-1)$D simplex					$\ldots$	$\binom{N+1}{N} = N+1$
ND simplex				1	$\ldots$	$\binom{N+1}{N+1} = 1$

♢ Rotations ♢

In N Euclidean dimensions, there are $\binom{N}{2} = N(N-1)/2$ degrees of rotational freedom corresponding to the free parameters of the group $SO(N)$. In 2D, that means we only have one rotational degree of freedom given by the angle used to mix the x and y coordinates. In 3D, there are 3 parameters, which can be thought of as corresponding either to three Euler angles or to the three independent quaternion coordinates that remain when we represent rotations in terms of unit quaternions. In 4D, there are 6 degrees of freedom, and the familiar 3D picture of "rotating about an axis" is no longer valid; each rotation leaves an entire plane fixed, not just one axis.

General rotations in N dimensions may be viewed as a sequence of elementary rotations. Each elementary rotation acts in the plane of a particular pair, say (i, j), of coordinates, leaving an $(N-2)$-dimensional subspace unchanged; we may write any such rotation in the form

$$\begin{aligned} x'^{(i)} &= x^{(i)} \cos\theta \pm x^{(j)} \sin\theta \\ x'^{(j)} &= \mp x^{(i)} \sin\theta + x^{(j)} \cos\theta \\ x'^{(k)} &= x^{(k)} \quad (k \neq i, j) \ . \end{aligned}$$

It is important to remember that *order matters* when doing a sequence of nested rotations; for example, two sequences of small 3D rotations, one consisting of a $(2, 3)$-plane rotation followed

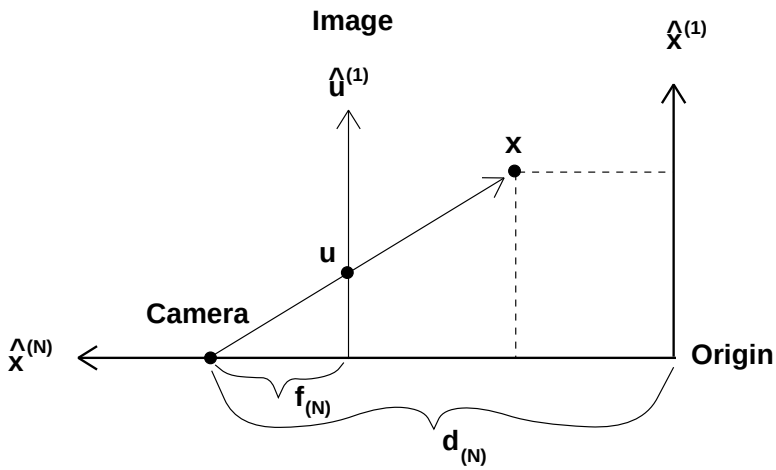


Figure 2. Schematic view of the projection process for an N-dimensional pinhole camera.

by a $(3,1)$-plane rotation, and the other with the order reversed, will differ by a rotation in the $(1,2)$-plane. (See any standard reference such as (Edmonds 1957).)

We then have a number of options for controlling rotations in N-dimensional Euclidean space. Among these are the following:

- (i,j)**-space pairs.** A brute-force choice would be just to pick a sequence of (i,j) planes in which to rotate using a series of matrix multiplications.
- (i,j,k)**-space triples.** A more interesting choice for an interactive system is to provide the user with a family of (i,j,k) triples having a 2D controller like a mouse coupled to two of the degrees of freedom, and having the 3rd degree of freedom accessible in some other way — with a different button, from context using the "virtual sphere" algorithm of (Chen et al. 1988), or implicitly using a context-free method like the "rolling-ball" algorithm (Hanson 1992). The simplest example is $(1,2,3)$ in 3D, with the mouse coupled to rotations about the $\hat{\mathbf{x}}$-axis $(2,3)$ and the $\hat{\mathbf{y}}$-axis $(3,1)$, giving $\hat{\mathbf{z}}$-axis $(1,2)$ rotations as a side-effect. In 4D, one would have four copies of such a controller, $(1,2,3)$, $(2,3,4)$, $(3,1,4)$, and $(1,2,4)$, or two copies exploiting the decomposition of $SO(4)$ infinitesimal rotations into two independent copies of ordinary 3D rotations. In N dimensions, $\begin{pmatrix} N \\ 3 \end{pmatrix}$ sets of these controllers (far too many when N is large!) could in principle be used.

♢ N-dimensional Imaging ♢

The general concept of an "image" is a projection of a point $\vec{x} = (x^{(1)}, x^{(2)}, \ldots, x^{(N)})$ from dimension N to a point $\vec{u}$ of dimension $(N-1)$ along a line. That is, the image of a 2D world

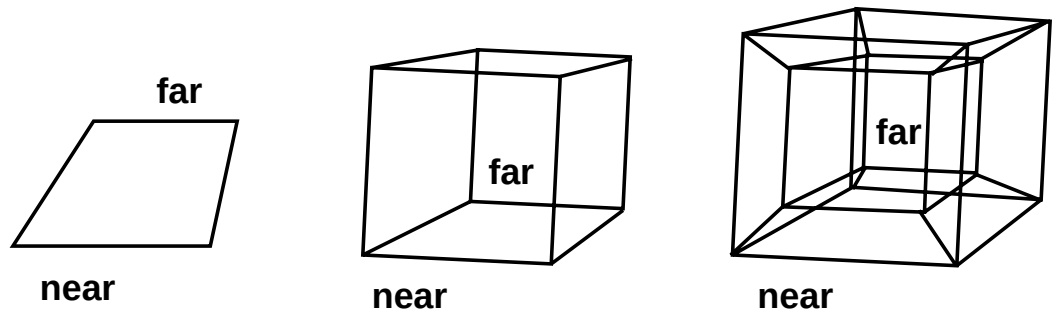


Figure 3. Qualitative results of perspective projection of a wire-frame square, a cube, and a hypercube in 2D, 3D, and 4D, respectively.

is a projection to 1D film, 3D worlds project to 2D film, 4D worlds project to 3D film, and so on. Since we can rotate our coordinate system as we please, we lose no generality if we assume this projection is along the N-th coordinate axis. An orthographic or parallel projection results if we simply throw out the N-th coordinate $x^{(N)}$ of each point. A pinhole camera perspective projection (see figure 2) results when, in addition, we scale the first $(N-1)$ coordinates by dividing by $(d_N - x^{(N)})/f_N$, where d_N is the distance along the positive N-th axis to the camera focal point and f_N is the focal length. One may need to project this first image to successively lower dimensions to make it displayable on a 2D graphics screen; thus a hierarchy of up to $(N-2)$ parameter sets $\{(f_N, d_N), \ldots, (f_3, d_3)\}$ may be introduced if desired.

In the familiar 3D case, we replace a vertex (x, y, z) of an object by the 2D coordinates $(xf/(d-z), yf/(d-z))$, so that more distant objects (in the negative z direction) are shrunk in the 2D image. In 4D, entire solid objects are shrunk, thus giving rise to the familiar wire-frame hypercube shown in figure 3 that has the more distant cubic hyperfaces actually lying *inside* the projection of the nearest cube.

As we will see a bit later when we discuss normals and cross-products, the usual shading approaches allow only $(N-1)$-manifolds to interact uniquely with a light ray. That is, the generalization of a viewable "object" to N dimensions is a manifold of dimension $(N-1)$ that bounds an N-dimensional volume; only this boundary is visible in the projected image if the object is opaque. For example, curves in 2D reflect light toward the focal point to form images on a "film line," surface patches in 3D form area images on a 2D film plane, volume patches in 4D form volume images in the 3D film volume, etc. The image of this $(N-1)$-dimensional patch may be ray traced or scan converted. Objects are typically represented as tessellations which consist of a collection of $(N-1)$-dimensional simplexes; for example, triangular surface patches form models of the visible parts of 3D objects, while tetrahedral volumes form models of the visible parts of 4D objects. (An interesting side issue is how to display meaningful illuminated images of lower dimensional manifolds — lines in 3D, surfaces and lines in 4D, etc.; see (Hanson and Heng 1992b) for further discussion.)

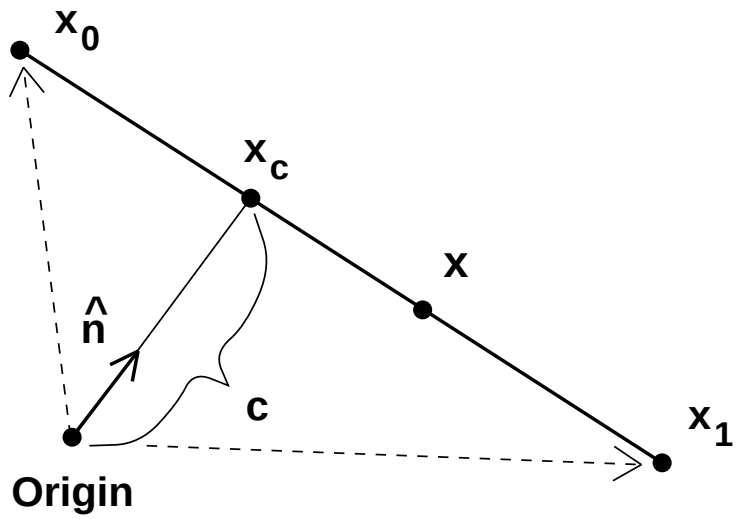


Figure 4. The line from $\vec{x}_0$ to $\vec{x}_1$ whose points obey the equation $\hat{\mathbf{n}} \cdot (\vec{x} - \vec{x}_0) = 0$. The constant c is just $\hat{\mathbf{n}} \cdot \vec{x}_0$.

♢ Hyperplanes and Volume Formulas ♢

Implicit Equation of a Hyperplane. In 2D, a special role is played by the single linear equation defining a line; in 3D, the analogous single linear equation defines a plane. In N-dimensions, the following implicit linear equation describes a set of points belonging to an $(N-1)$-dimensional hyperplane:

$$\hat{\mathbf{n}} \cdot (\vec{x} - \vec{x}_0) = 0\,. \tag{1}$$

Here $\vec{x}_0$ is any point on the hyperplane and conventionally $\hat{\mathbf{n}} \cdot \hat{\mathbf{n}} = 1$. The geometric interpretation of this equation in 2D is the 1D line shown in figure 4. In general, $\hat{\mathbf{n}}$ is a normalized unit vector that is perpendicular to the hyperplane, and $\hat{\mathbf{n}} \cdot \vec{x}_0 = c$ is simply the (signed) distance from the origin to the hyperplane. The point $\vec{x}_c = c\hat{\mathbf{n}}$ is the point on the hyperplane closest to the origin; the point closest to some other point $\vec{P}$ is $\vec{x}_c = \vec{P} + \hat{\mathbf{n}}\{\hat{\mathbf{n}} \cdot (\vec{x}_0 - \vec{P})\}$.

Simplex Volumes and Subvolumes. The volume (by which we always mean the N-dimensional hypervolume) of an N-simplex is determined in a natural way by a determinant of its $(N+1)$ defining points (Sommerville 1958):

$$V_N = \frac{1}{N!} \det \begin{bmatrix} x_1 & x_2 & \cdots & x_N & x_0 \\ y_1 & y_2 & \cdots & y_N & y_0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ w_1 & w_2 & \cdots & w_N & w_0 \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix}\,. \tag{2}$$

The bottom row of 1's in eq. (2) corresponds to the familiar homogeneous coordinate used with 4×4 projection matrices in 3D graphics. We will attempt to convince the reader in a moment that disastrous sign inconsistencies result unless the global origin $\vec{x}_0$ of the N-simplex's coordinate system is in the last column as shown.

The expression for the volume in eq. (2) is *signed*, which means that it implicitly defines the N-dimensional generalization of the *Right-Hand Rule* typically adopted to determine triangle orientation in 3D geometry. For example, we observe that if $\vec{x}_0 = (0, 0, \ldots, 0)$ is the origin and we choose $\vec{x}_1 = (1, 0, \ldots, 0)$, $\vec{x}_2 = (0, 1, 0, \ldots, 0)$, and so on, the value of the determinant is $+1$. If we had put $\vec{x}_0$ in the first row in eq. (2), the sign would alternate from dimension to dimension! We will exploit this signed determinant shortly to define N-dimensional normal vectors, and again later to formulate N-dimensional clipping.

First, we use the standard column-subtraction identity for determinants to reduce the dimension of the determinant in eq. (2) by one, expressing it in a form that is manifestly *translation-invariant*:

$$\begin{aligned} V_N &= \frac{1}{N!} \det \begin{bmatrix} (x_1 - x_0) & (x_2 - x_0) & \cdots & (x_N - x_0) & x_0 \\ (y_1 - y_0) & (y_2 - y_0) & \cdots & (y_N - y_0) & y_0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ (w_1 - w_0) & (w_2 - w_0) & \cdots & (w_N - w_0) & w_0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \\ &= \frac{1}{N!} \det \begin{bmatrix} (x_1 - x_0) & (x_2 - x_0) & \cdots & (x_N - x_0) \\ (y_1 - y_0) & (y_2 - y_0) & \cdots & (y_N - y_0) \\ \vdots & \vdots & \ddots & \vdots \\ (w_1 - w_0) & (w_2 - w_0) & \cdots & (w_N - w_0) \end{bmatrix} . \end{aligned} \tag{3}$$

These formulas for V_N can be intuitively understood as generalizations of the familiar 3D triple scalar product,

$$[(\vec{x}_1 - \vec{x}_0) \times (\vec{x}_2 - \vec{x}_0)] \cdot (\vec{x}_3 - \vec{x}_0) ,$$

which gives the volume of the parallelepiped with sides $((\vec{x}_1 - \vec{x}_0), (\vec{x}_2 - \vec{x}_0), (\vec{x}_3 - \vec{x}_0))$. The corresponding tetrahedron with vertices at the points $(\vec{x}_0, \vec{x}_1, \vec{x}_2, \vec{x}_3)$ has one-sixth the volume of the parallelepiped. The analogous observation in N dimensions is that the factor of $1/N!$ in eq. (3) is the proportionality factor between the volume of the N-simplex and the volume of the parallelepiped whose edges are given by the matrix columns.

Invariance. The volume determinant is invariant under rotations. To see this explicitly, let $|X|$ be the matrix in eq. (3) and let $|R|$ be any orthonormal rotation matrix (i.e., one whose columns are of unit length and are mutually perpendicular, with unit determinant); then, letting $|X'| = |R| \cdot |X|$, we find

$$\det |X'| = \det(|R| \cdot |X|) = \det |R| \det |X| = \det |X| = N!\, V_N ,$$

since the determinant of a product is the product of the determinants.

A manifestly translation *and* rotation invariant form for the square of the volume element is

$$\begin{aligned}(V_N)^2 &= \left(\frac{1}{N!}\right)^2 \det|X^t \cdot X| \\ &= \left(\frac{1}{N!}\right)^2 \det\begin{bmatrix} v(1,1) & v(1,2) & \cdots & v(1,N) \\ v(2,1) & v(2,2) & \cdots & v(2,N) \\ \vdots & \vdots & \ddots & \vdots \\ v(N,1) & v(N,2) & \cdots & v(N,N) \end{bmatrix}, \end{aligned} \tag{4}$$

where $v(i,j) = (\vec{x}_i - \vec{x}_0) \cdot (\vec{x}_j - \vec{x}_0)$.

This invariant form is not presented as an idle observation; we now exploit it to show how to construct volume forms for *subspaces* of N-dimensional spaces, for which the defining vertices of the desired simplex cannot form square matrices!

The trick here is to note that while V_K, for $K < N$, is not expressible in terms of a square matrix of coordinate differences the way V_N is, we may write V_K as the determinant of a square matrix in one particular coordinate frame, and multiply this matrix by its transpose to get a form like eq. 4, which does not depend on the frame. Since the form is invariant, we can transform back to an arbitrary frame to find the following expression for V_K in terms of its K basis vectors $(\vec{x}_k - \vec{x}_0)$ of dimension N:

$$\begin{aligned}(V_K)^2 &= \left(\frac{1}{K!}\right)^2 \det\begin{bmatrix} \vec{x}_1 - \vec{x}_0 \\ \vec{x}_2 - \vec{x}_0 \\ \vdots \\ \vec{x}_K - \vec{x}_0 \end{bmatrix} \cdot \begin{bmatrix} \vec{x}_1 - \vec{x}_0 & \vec{x}_2 - \vec{x}_0 & \cdots & \vec{x}_K - \vec{x}_0 \end{bmatrix} \\ &= \left(\frac{1}{K!}\right)^2 \det\begin{bmatrix} v(1,1) & v(1,2) & \cdots & v(1,K) \\ v(2,1) & v(2,2) & \cdots & v(2,K) \\ \vdots & \vdots & \ddots & \vdots \\ v(K,1) & v(K,2) & \cdots & v(K,K) \end{bmatrix}. \end{aligned} \tag{5}$$

That is, to compute a volume of dimension K in N dimensions, find the K independent basis vectors spanning the subspace, and form a square $K \times K$ matrix of dot products related to V_K^2 by multiplying the $N \times K$ matrix of column vectors by its transpose on the left. When $K = 1$, we see that we have simply the squared Euclidean distance in N dimensions, $v(1,1) = (\vec{x}_1 - \vec{x}_0) \cdot (\vec{x}_1 - \vec{x}_0)$.

♢ Normals and the Cross-Product ♢

A frequently asked question in N-dimensional geometry concerns how to define a normal vector as a cross-product of edges for use in geometry and shading calculations. To begin with,

you must have an $(N-1)$-manifold (a line in 2D, surface in 3D, volume in 4D) in order to have a well-defined normal *vector*; otherwise, you may have a normal *space* (a plane, a volume, etc.). Suppose you have an ordered set of $(N-1)$ edge vectors $(\vec{x}_k - \vec{x}_0)$ tangent to this $(N-1)$-manifold at a point $\vec{x}_0$; typically these vectors are the edges of one of the $(N-1)$-simplexes in the tessellation. Then the normal $\vec{N}$ at the point is a *generalized cross-product* whose components are cofactors of the last column in the following (notationally abusive!) determinant:

$$\begin{aligned}\vec{N} &= N_x\hat{\mathbf{x}} + N_y\hat{\mathbf{y}} + N_z\hat{\mathbf{z}} + \cdots + N_w\hat{\mathbf{w}} \\ &= \det\begin{bmatrix} (x_1-x_0) & (x_2-x_0) & \cdots & (x_{N-1}-x_0) & \hat{\mathbf{x}} \\ (y_1-y_0) & (y_2-y_0) & \cdots & (y_{N-1}-y_0) & \hat{\mathbf{y}} \\ (z_1-z_0) & (z_2-z_0) & \cdots & (z_{N-1}-z_0) & \hat{\mathbf{z}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ (w_1-w_0) & (w_2-w_0) & \cdots & (w_{N-1}-w_0) & \hat{\mathbf{w}} \end{bmatrix}. \qquad (6)\end{aligned}$$

As usual, we can normalize using $\|\vec{N}\|$, the square root of the sum of the squares of the cofactors, to form the normalized normal $\hat{\mathbf{n}} = \vec{N}/\|\vec{N}\|$. A quick check shows that if the vectors $(\vec{x}_k - \vec{x}_0)$ are assigned to the first $(N-1)$ coordinate axes in order, this normal vector points in the direction of the positive N-th axis. For example, in 2D, we want the normal to the vector $(x_1 - x_0,\ y_1 - y_0)$ to be $\vec{N} = (-(y_1 - y_0),\ (x_1 - x_0))$ so that a vector purely in the x direction has a normal in the positive y direction; placing the column of unit vectors $(\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}, \ldots, \hat{\mathbf{w}})$ in the *first* column fails this test. The 3D case can be done either way because an even number of columns are crossed! It is tempting to move the column of unit vectors to the first column instead of the last, but one must resist: the choice given here is the one to use for consistent behavior across different dimensions!

The qualitative interpretation of eq. (6) can now be summarized as follows:

- **2D:** Given two points $(\vec{x}_0, \vec{x}_1)$ determining a line in 2D, the cross-product of a *single vector* is the normal to the line.
- **3D:** Given three points defining a plane in 3D, the cross-product of the two 3D vectors outlining the resulting triangle is the familiar formula $(\vec{x}_1 - \vec{x}_0) \times (\vec{x}_2 - \vec{x}_0)$ for the normal $\vec{N}$ to the plane.
- **4D:** In four dimensions, we use four points to construct the three vectors $(\vec{x}_1 - \vec{x}_0), (\vec{x}_2 - \vec{x}_0), (\vec{x}_3 - \vec{x}_0)$; the cross product of these vectors is a *four-vector* that is perpendicular to each vector and thus is interpretable as the normal to the tetrahedron specified by the original four points.

From this point on, the relationship to standard graphics computations should be evident: If, in N-dimensional space, the $(N-1)$-manifold to be rendered is tessellated into $(N-1)$-simplexes, use eq. (6) to compute the normal of each simplex for flat shading. For interpolated shading, compute the normal at each vertex (e.g., by averaging the normals of all neighboring

simplexes and normalizing or by computing the gradient of an implicit function specifying the vertex). Compute the intensity at a point for which you know the normal by taking the dot product of the appropriate illumination vector with the normal (e.g, by plugging it into the last column of eq. (6)). If appropriate, set the dot product to zero if it is negative (pointing away from the light). Back face culling, to avoid rendering simplexes pointing away from the camera, is accomplished in exactly the same way: plug the camera view vector into the last column of eq. (6) and discard the simplex if the result is negative.

Dot Products of Cross Products. We conclude this section with the remark that sometimes computing the dot product between a normal and a simple vector is not enough; if we need to know the relative orientation of two face normals (e.g., to determine whether a finer tessellation is required), we must compute the dot products of normals. In principle, this can be done by brute force directly from eq. (6). Here we note an alternative formulation that is the N-dimensional generalization of the 3D formula for the decomposition of the dot product of two cross products; in the 3D case, if one normal is given by the cross product $\vec{X} = \vec{A} \times \vec{B}$ and the other by $\vec{Y} = \vec{C} \times \vec{D}$, we can write

$$\vec{X} \cdot \vec{Y} = (\vec{A} \times \vec{B}) \cdot (\vec{C} \times \vec{D}) = (\vec{A} \cdot \vec{C})(\vec{B} \cdot \vec{D}) - (\vec{A} \cdot \vec{D})(\vec{B} \cdot \vec{C}) \ . \tag{7}$$

We note that the degenerate case for the square of a cross product is

$$(\vec{A} \times \vec{B}) \cdot (\vec{A} \times \vec{B}) = (\vec{A} \cdot \vec{A})(\vec{B} \cdot \vec{B}) - (\vec{A} \cdot \vec{B})^2 \ ,$$

which, if θ is the angle between $\vec{A}$ and $\vec{B}$, reduces to the identity $\|\vec{A}\|^2 \|\vec{B}\|^2 \sin^2\theta = \|\vec{A}\|^2 \|\vec{B}\|^2 - \|\vec{A}\|^2 \|\vec{B}\|^2 \cos^2\theta$.

The generalization of this expression to N dimensions can be derived from the product of two Levi-Civita symbols (see the Appendix). If $\vec{X}$ and $\vec{Y}$ are two cross products formed from the sets of vectors $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{N-1}$ and $\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_{N-1}$, then

$$\begin{aligned} \vec{X} \cdot \vec{Y} \quad = \quad & \sum_{\text{all indices}} x_1^{(i_1)} x_2^{(i_2)} \ldots x_{N-1}^{(i_{N-1})} y_1^{(j_1)} y_2^{(j_2)} \ldots y_{N-1}^{(j_{N-1})} \\ & \det \begin{bmatrix} \delta_{i_1 j_1} & \delta_{i_1 j_2} & \cdots & \delta_{i_1 j_{N-1}} \\ \delta_{i_2 j_1} & \delta_{i_2 j_2} & \cdots & \delta_{i_2 j_{N-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{i_{N-1} j_1} & \delta_{i_{N-1} j_2} & \cdots & \delta_{i_{N-1} j_{N-1}} \end{bmatrix} \ , \end{aligned} \tag{8}$$

where the Kronecker delta, δ_{ij}, is defined as

$$\begin{aligned} \delta_{ij} \quad &= \quad 1 \qquad\qquad i = j \\ &= \quad 0 \qquad\qquad i \neq j \ . \end{aligned}$$

It is easy to verify that for $N = 3$ this reduces to eq. (7).

More remarkable, however, is the fact that this formula shows that the square magnitude of the normal $\vec{N}$ of a hyperplane given in eq. (6) is the *subvolume* of the corresponding parallelepiped specified by eq. (5). That is, not only does the *direction* of eq. (6) have an important geometric meaning with respect to the $(N-1)$-simplex specifying the hyperplane, but so does its *magnitude*! We find

$$\vec{N}\cdot\vec{N} = \det\begin{bmatrix} v(1,1) & v(1,2) & \cdots & v(1,N-1) \\ v(2,1) & v(2,2) & \cdots & v(2,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ v(N-1,1) & v(N-1,2) & \cdots & v(N-1,N-1) \end{bmatrix} = \left((N-1)!\,V_{N-1}\right)^2 .$$

♢ Clipping Tests in N Dimensions ♢

Now we can exploit the properties of the volume formula to define clipping ("which side") tests in any dimension. If we replace $(\vec{x}_N - \vec{x}_0)$ by $(\vec{x} - \vec{x}_0)$, eq. (3) becomes a *function* $V_N(\vec{x})$. Furthermore, this function has the remarkable property that it is an alternative form for the hyperplane equation, eq. (1), when $V_N(\vec{x}) = 0$.

We can furthermore determine *on which side* of the $(N-1)$-dimensional hyperplane determined by $(\vec{x}_0, \vec{x}_1, \ldots, \vec{x}_{N-1})$ an arbitrary point $\vec{x}$ lies simply by checking the sign of $V_N(\vec{x})$. That is,

- $V_N(\vec{x}) = 0 \Rightarrow$ the point $\vec{x}$ lies on a hyperplane and solves an equation of the form eq. (1).
- $V_N(\vec{x}) > 0 \Rightarrow$ the point $\vec{x}$ lies above the hyperplane.
- $V_N(\vec{x}) < 0 \Rightarrow$ the point $\vec{x}$ lies below the hyperplane.

Note: The special case $V_N = 0$ is of course just the general criterion for discovering *linear dependence* among a set of $(N+1)$ vector variables. This has the following elegant geometric interpretation: In 2D, we use the formula to compute the area of the triangle formed by 3 points $(\vec{x}_0, \vec{x}_1, \vec{x})$; if the area vanishes, the 3 points lie on a single line. In 3D, if the volume of the tetrahedron formed by 4 points $(\vec{x}_0, \vec{x}_1, \vec{x}_2, \vec{x})$ vanishes, all 4 points are coplanar, and so on. Vanishing N-volume means the points lie in a hyperplane of dimension no greater than $(N-1)$.

These relationships between the sign of $V_N(\vec{x})$ and the relative position of $\vec{x}$ are precisely those we are accustomed to examining when we *clip* vectors (e.g., edges of a triangle) to lie on one side of a plane in a viewing frustum or within a projected viewing rectangle. For example, a 2D clipping line defined by the vector $\vec{x}_1 - \vec{x}_0 = (x_1 - x_0,\, y_1 - y_0)$ has a right-handed (unnormalized) normal $\vec{N} = (-(y_1 - y_0),\, (x_1 - x_0))$. Writing the 2D volume as the area A, eq. (3) becomes

$$A(\vec{x}) = \frac{1}{2}\det\begin{bmatrix} (x_1 - x_0) & (x - x_0) \\ (y_1 - y_0) & (y - y_0) \end{bmatrix} = \frac{1}{2}\left[\vec{N}\cdot(\vec{x} - \vec{x}_0)\right]$$

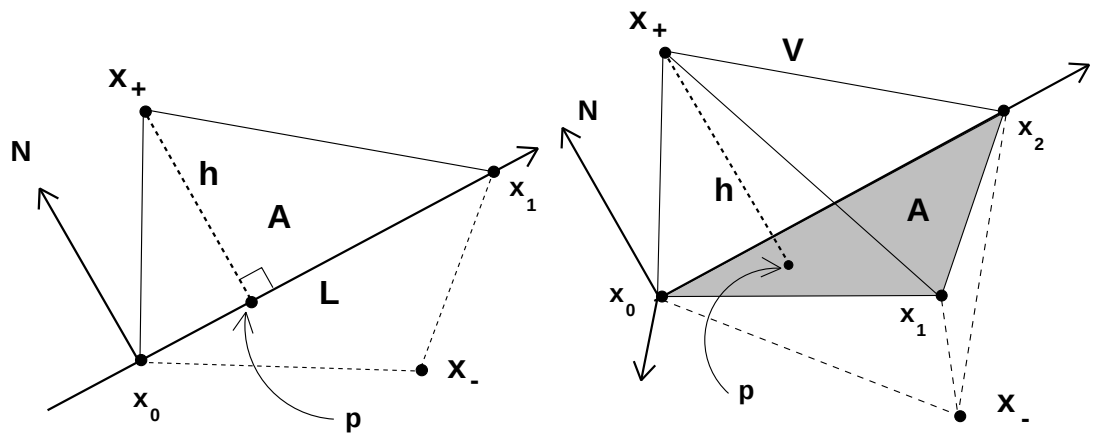


Figure 5. In 2D, the line through $\vec{x}_0$ to $\vec{x}_1$ defined by $\hat{\mathbf{n}} \cdot (\vec{x} - \vec{x}_0) = 0$ partitions the plane into two regions, one where this expression is positive (e.g., for $\vec{x}_+$) and another where it is negative (e.g., for $\vec{x}_-$). In 3D, the analogous procedure uses the plane defined by $(\vec{x}_0, \vec{x}_1, \vec{x}_2)$ to divide 3-space into two half spaces. The same pictures serve to show how the distance h from a point to a hyperplane is computable from the ratio of the simplex volume to the lower-dimensional volume of its base, i.e., $2A/L$ or $3V/A$.

for some arbitrary point $\vec{x}$, and so we recover the form of eq. (1) as

$$\hat{\mathbf{n}} \cdot (\vec{x} - \vec{x}_0) = \frac{2A}{\|\vec{x}_1 - \vec{x}_0\|} ,$$

where $\hat{\mathbf{n}} = \vec{N}/\|\vec{N}\|$; the relationship of $\vec{x}$ to the clipping line is determined by the sign.

In 3D, when clipping a line against a plane, everything reduces to the traditional form, namely the dot product between a 3D cross-product and a vector from a point $\vec{x}_0$ in the clipping plane to the point $\vec{x}$ being clipped. The normal to the plane through $(\vec{x}_0, \vec{x}_1, \vec{x}_2)$ is

$$\begin{aligned} \vec{N} &= (\vec{x}_1 - \vec{x}_0) \times (\vec{x}_2 - \vec{x}_0) \qquad (9) \\ &= \Bigg(+\det \begin{bmatrix} (y_1 - y_0) & (y_2 - y_0) \\ (z_1 - z_0) & (z_2 - z_0) \end{bmatrix}, \\ &\qquad -\det \begin{bmatrix} (x_1 - x_0) & (x_2 - x_0) \\ (z_1 - z_0) & (z_2 - z_0) \end{bmatrix}, +\det \begin{bmatrix} (x_1 - x_0) & (x_2 - x_0) \\ (y_1 - y_0) & (y_2 - y_0) \end{bmatrix} \Bigg) , \end{aligned}$$

and we again find the same general form,

$$\hat{\mathbf{n}} \cdot (\vec{x} - \vec{x}_0) = \frac{6V}{\|\vec{N}\|} ,$$

whose sign determines where $\vec{x}$ falls. Figure 5 summarizes the relationship of the signed volume to the clipping task in 2D and 3D.

Hyperplanes for clipping applications in any dimension are therefore easily defined and checked by choosing $\vec{x}_N$ to be the test point $\vec{x}$ and checking the sign of eq. (3). If $\vec{N}$ and a point $\vec{x}_0$ are easy to determine directly, then the procedure reduces to checking the sign of the left hand side of eq. (1).

The final step is to find the desired point on the truncated, clipped line. Since the clipped form of a triangle, tetrahedron, etc., can be determined from the clipped forms of the component lines, we need only consider the point at which a line straddling the clipping hyperplane intersects this hyperplane. If the line to be clipped is given parametrically as $\vec{x}(t) = \vec{x}_a + t(\vec{x}_b - \vec{x}_a)$, where $\vec{x}_a$ and $\vec{x}_b$ are on opposite sides of the clipping hyperplane so $0 \le t \le 1$, then we simply plug $\vec{x}(t)$ into $V(\vec{x}) = 0$ and solve for t:

$$t = \frac{\det\left[\begin{array}{cccc} \vec{x}_1 - \vec{x}_0 & \vec{x}_2 - \vec{x}_0 & \cdots & \vec{x}_a - \vec{x}_0 \end{array}\right]}{\det\left[\begin{array}{cccc} \vec{x}_1 - \vec{x}_0 & \vec{x}_2 - \vec{x}_0 & \cdots & \vec{x}_a - \vec{x}_b \end{array}\right]} = \frac{\hat{\mathbf{n}} \cdot (\vec{x}_a - \vec{x}_0)}{\hat{\mathbf{n}} \cdot (\vec{x}_a - \vec{x}_b)} \ . \tag{10}$$

Here $\hat{\mathbf{n}}$ is of course just the normal to the clipping hyperplane, discussed in detail above.

♢ Point-Hyperplane Distance ♢

The general formula for the volume of a parallelepiped is the product of the base and the height, $W = Bh$. In N dimensions, if we take $W_N = N!\, V_N$ to be the volume of the parallelepiped with edges $(\vec{x}_1 - \vec{x}_0), (\vec{x}_2 - \vec{x}_0), \ldots, (\vec{x}_{N-1} - \vec{x}_0), (\vec{x} - \vec{x}_0)$, this generalizes to

$$W_N = hW_{N-1} \ ,$$

where h is the perpendicular distance from the point $\vec{x}$ to the $(N-1)$-dimensional parallelepiped with volume $W_{N-1} = (N-1)!\, V_{N-1}$ and edges $(\vec{x}_1 - \vec{x}_0), (\vec{x}_2 - \vec{x}_0), \ldots, (\vec{x}_{N-1} - \vec{x}_0)$. We may thus immediately compute the distance h from a point to a hyperplane as

$$h = \frac{W_N}{W_{N-1}} = \frac{N!\, V_N}{(N-1)!\, V_{N-1}} = \frac{N\, V_N}{V_{N-1}} \ . \tag{11}$$

Note! Here one must use the trick of eq. 4 to express W_{N-1} in terms of the square root of a square determinant given by the product of two non-square matrices.

Thus in 2D, the area of a triangle $(\vec{x}_0, \vec{x}_1, \vec{x})$ is

$$A = V_2 = \frac{1}{2} W_2 = \frac{1}{2} \det\left[\begin{array}{cc} (x_1 - x_0) & (x - x_0) \\ (y_1 - y_0) & (y - y_0) \end{array}\right]$$

and the length-squared of the base is $L^2 = (\vec{x}_1 - \vec{x}_0) \cdot (\vec{x}_1 - \vec{x}_0)$ so, with $A = (1/2)hL$, the height becomes $h = 2A/L = W_2/L = W_2/W_1$. In 3D, the volume of the tetrahedron $(\vec{x}_0, \vec{x}_1, \vec{x}_2, \vec{x})$ is $V = V_3 = (1/6)W_3$ and the area $A = (1/2)W_2$ of the triangular base may be written

$$(2A)^2 = (W_2)^2 = \det\left[\begin{array}{cc} (\vec{x}_1 - \vec{x}_0) \cdot (\vec{x}_1 - \vec{x}_0) & (\vec{x}_1 - \vec{x}_0) \cdot (\vec{x}_2 - \vec{x}_0) \\ (\vec{x}_2 - \vec{x}_0) \cdot (\vec{x}_1 - \vec{x}_0) & (\vec{x}_2 - \vec{x}_0) \cdot (\vec{x}_2 - \vec{x}_0) \end{array}\right] \ .$$

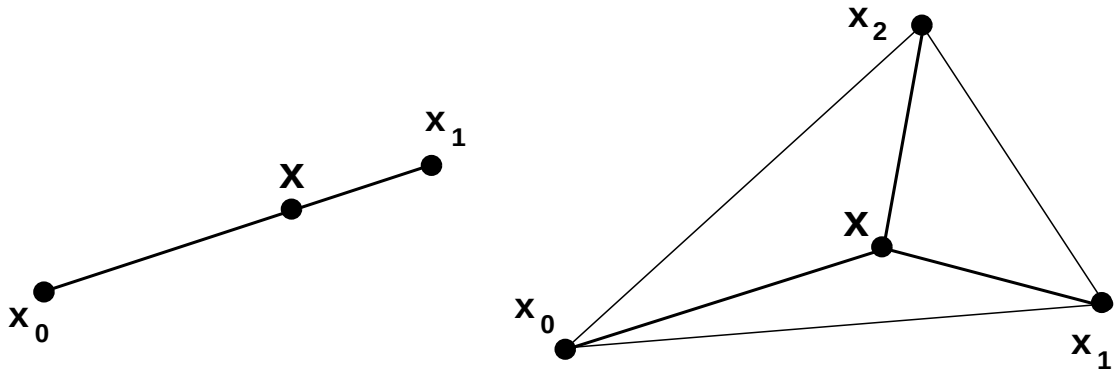


Figure 6. Barycentric coordinates in N dimensions.

We know $V = (1/3)hA$, and so $h = 3V/A = 6V/2A = W_3/W_2$. (See figure 5.) We note for reference that, as we showed earlier, the base $(N-1)$-volume is related to its normal by $\vec{N} \cdot \vec{N} = W_{N-1}^2$.

Here we also typically need to answer one last question, namely *where* is the point $\vec{p}$ on the base hyperplane closest to the point $\vec{x}$ whose distance h we just computed? This can be found by parameterizing the line from $\vec{x}$ to the base hyperplane along the normal $\hat{\mathbf{n}}$ to the hyperplane as $\vec{x}(t) = \vec{x} + t\hat{\mathbf{n}}$, writing the implicit equation for the hyperplane as $\hat{\mathbf{n}} \cdot (\vec{x}(t) - \vec{x}_0) = 0$, and solving for the mutual solution $t_p = \hat{\mathbf{n}} \cdot (\vec{x}_0 - \vec{x}) = -h$. Thus

$$\begin{aligned} \vec{p} &= \vec{x} + \hat{\mathbf{n}}(\hat{\mathbf{n}} \cdot (\vec{x}_0 - \vec{x})) \\ &= \vec{x} - h\hat{\mathbf{n}} \ . \end{aligned}$$

♢ Barycentric Coordinates ♢

Barycentric coordinates (see, e.g., (Hocking and Young 1961), chapter 5) are a practical way to parameterize lines, surfaces, etc., for applications that must compute where various geometric objects intersect. In practice, the barycentric coordinate method reduces to specifying two points $(\vec{x}_0, \vec{x}_1)$ on a line, three points $(\vec{x}_0, \vec{x}_1, \vec{x}_2)$ on a plane, four points $(\vec{x}_0, \vec{x}_1, \vec{x}_2, \vec{x}_3)$ in a volume, etc., and parameterizing the line segment, enclosed triangular area, and enclosed tetrahedral volume, etc., respectively, by

$$\vec{x}(t) = \vec{x}_0 + t(\vec{x}_1 - \vec{x}_0) \tag{12}$$

$$\vec{x}(t_1, t_2) = \vec{x}_0 + t_1(\vec{x}_1 - \vec{x}_0) + t_2(\vec{x}_2 - \vec{x}_0) \tag{13}$$

$$\vec{x}(t_1, t_2, t_3) = \vec{x}_0 + t_1(\vec{x}_1 - \vec{x}_0) + t_2(\vec{x}_2 - \vec{x}_0) + t_3(\vec{x}_3 - \vec{x}_0) \tag{14}$$

$$\cdots \ .$$

The line and plane geometries are shown in figure 6. The interpolated point then lies within the N-simplex defined by the specified points provided

$$0 \le t \le 1$$
$$0 \le t_1 \le 1,\ 0 \le t_2 \le 1,\ 0 \le (1 - t_1 - t_2) \le 1$$
$$0 \le t_1 \le 1,\ 0 \le t_2 \le 1,\ 0 \le t_3 \le 1,\ 0 \le (1 - t_1 - t_2 - t_3) \le 1$$
$$\ldots$$

Center of What? However, this is really only half the story of barycentric coordinates. For the other half, we seek a geometric interpretation of the parameters t_i when we are *given* the value of $\vec{x}$.

First let us look at the simple case when $\vec{x}$ lies on the line segment between $\vec{x}_0$ and $\vec{x}_1$. Solving eq. (12) for t directly gives

$$t = \frac{(\vec{x} - \vec{x}_0) \cdot (\vec{x}_1 - \vec{x}_0)}{(\vec{x}_1 - \vec{x}_0) \cdot (\vec{x}_1 - \vec{x}_0)} \ .$$

That is, t is the fraction of the distance that $\vec{x}$ has traveled along the line, the *ratio* between the length from $\vec{x}_0$ to $\vec{x}$ and the total length. But, since $\vec{x}_1 - \vec{x}_0 = \vec{x}_1 - \vec{x} + \vec{x} - \vec{x}_0$, we easily see that an alternative parameterization would be to take $t_1 = t$ and

$$t_0 = \frac{(\vec{x}_1 - \vec{x}) \cdot (\vec{x}_1 - \vec{x}_0)}{(\vec{x}_1 - \vec{x}_0) \cdot (\vec{x}_1 - \vec{x}_0)}$$

so that $1 = t_0 + t_1$ and eq. (12) for $\vec{x}$ becomes

$$\vec{x}(t_0, t_1) = t_0 \vec{x}_0 + t_1 \vec{x}_1 \ .$$

If $t_0 = 1$, then the entire fraction of the distance from $\vec{x}_1$ to $\vec{x}$ is assigned to t_0 and $\vec{x} = \vec{x}_0$. If $t_1 = 1$, then the entire fraction of the distance from $\vec{x}_0$ to $\vec{x}$ is assigned to t_1 and $\vec{x} = \vec{x}_1$.

Next, suppose we know $\vec{x}$ in a plane and wish to compute its barycentric coordinates by solving eq. (13) for $(t_1,\ t_2)$. Once we realize that $(\vec{x}_1 - \vec{x}_0)$ and $(\vec{x}_2 - \vec{x}_0)$ form the basis for an affine coordinate system for the plane specified by $(\vec{x}_0, \vec{x}_1, \vec{x}_2)$ in *any* dimension, we see that we may measure the relative barycentric coordinates by taking the dot product with each basis vector:

$$\begin{aligned}
(\vec{x} - \vec{x}_0) \cdot (\vec{x}_1 - \vec{x}_0) &= t_1 \|\vec{x}_1 - \vec{x}_0\|^2 + t_2 (\vec{x}_2 - \vec{x}_0) \cdot (\vec{x}_1 - \vec{x}_0) \\
(\vec{x} - \vec{x}_0) \cdot (\vec{x}_2 - \vec{x}_0) &= t_1 (\vec{x}_1 - \vec{x}_0) \cdot (\vec{x}_2 - \vec{x}_0) + t_2 \|\vec{x}_2 - \vec{x}_0\|^2 \ .
\end{aligned}$$

Extending the previously introduced abbreviation to the form $v(x, j) = (\vec{x} - \vec{x}_0) \cdot (\vec{x}_j - \vec{x}_0)$ and solving this pair of equations by Cramer's rule, we get

$$t_1 = \frac{\det \begin{bmatrix} v(x,1) & v(1,2) \\ v(x,2) & v(2,2) \end{bmatrix}}{\det \begin{bmatrix} v(1,1) & v(1,2) \\ v(2,1) & v(2,2) \end{bmatrix}}$$

$$t_2 \quad = \quad \frac{\det \begin{bmatrix} v(1,1) & v(x,1) \\ v(1,2) & v(x,2) \end{bmatrix}}{\det \begin{bmatrix} v(1,1) & v(1,2) \\ v(2,1) & v(2,2) \end{bmatrix}} \, .$$

The denominator is clearly proportional to the *square* of the area of the triangle $(\vec{x}_0, \vec{x}_1, \vec{x}_2)$, and the numerators have the form of squared areas as well. In N dimensions, the numerators reduce to determinants of products of non-square matrices, and so may *not* be expressed as two separate determinants! However, if we transform to a coordinate system that contains the triangle within the plane of two coordinate axes, or if $N = 2$, an effectively square matrix is recovered; one factor of area in the denominator then cancels out, giving the intuitively expected result that the barycentric coordinates are ratios of two areas: $t_1 = A(\vec{x}, \vec{x}_0, \vec{x}_1)/A(\vec{x}_0, \vec{x}_1, \vec{x}_2)$, $t_2 = A(\vec{x}, \vec{x}_2, \vec{x}_0)/A(\vec{x}_0, \vec{x}_1, \vec{x}_2)$. This leads us to introduce the generalized version of t_0 for the line, namely,

$$\begin{aligned} t_0 \quad &= \quad 1 - t_1 - t_2 = \frac{A(\vec{x}_1, \vec{x}_2, \vec{x})}{A(\vec{x}_0, \vec{x}_1, \vec{x}_2)} \\ &= \quad \frac{\det \begin{bmatrix} (\vec{x}_1 - \vec{x}_0)\cdot(\vec{x}_1 - \vec{x}) & (\vec{x}_1 - \vec{x}_0)\cdot(\vec{x}_2 - \vec{x}) \\ (\vec{x}_2 - \vec{x}_0)\cdot(\vec{x}_1 - \vec{x}) & (\vec{x}_2 - \vec{x}_0)\cdot(\vec{x}_2 - \vec{x}) \end{bmatrix}}{\det \begin{bmatrix} v(1,1) & v(1,2) \\ v(2,2) & v(2,2) \end{bmatrix}} \, . \end{aligned}$$

Here we used the squaring argument given above to extend t_0 from its special-coordinate-system interpretation as the fraction of the area contributed by the triangle $(\vec{x}, \vec{x}_1, \vec{x}_2)$ to the invariant form. This form obviously has the desired property that $t_0 = 1$ when $\vec{x} = \vec{x}_0$, and we finally have the sought equation (with $1 = t_0 + t_1 + t_2$)

$$\vec{x}(t_0, t_1, t_2) = t_0\vec{x}_0 + t_1\vec{x}_1 + t_2\vec{x}_2 \, .$$

It is amusing to note that the determinant identity $1 = t_0 + t_1 + t_2$ and its higher analogs, which are nontrivial to derive, generalize the simple identity $\vec{x}_1 - \vec{x}_0 = \vec{x}_1 - \vec{x} + \vec{x} - \vec{x}_0$ that we used in the 1D case.

Thus we can construct barycentric coordinates in any dimension which intuitively correspond to *fractions of hypervolumes*; each barycentric coordinate is the hypervolume of an N-simplex defined by the point $\vec{x}$ and all but one of the other simplex-defining points divided by the volume of the whole simplex. The actual computation, however, is best done using the squared-volume form because only that form is independent of the chosen coordinate system.

Note: The volumes are *signed*; even if $\vec{x}$ lies outside the N-simplex volume, the ratios remain correct due to the cancellation between the larger volumes and the negative volumes. We also remark that the generalized formulas for t_i in any dimension, with $1 = \sum_{i=0}^{N} t_i$, give an elegant geometric interpretation of Cramer's rule as ratios of simplex volumes.

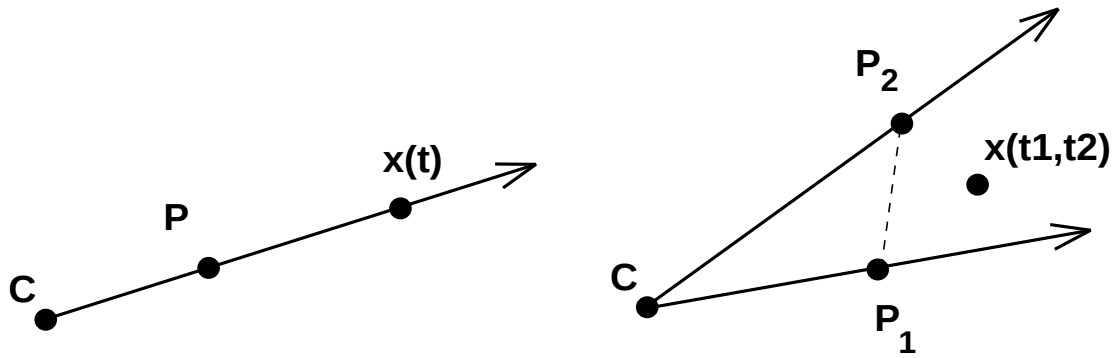


Figure 7. Schematic diagram comparing an ordinary camera ray and a planar "thick ray" used in N-dimensional ray-tracing methods.

♢ Ray Tracing ♢

It is often useful to compute the intersection of a ray passing through two points (typically the camera focal point $\vec{C}$ and an image point $\vec{P}$) with a geometrical object. In N dimensions, this object will typically be an $(N-1)$-simplex defining an oriented visible "face" with a normal vector computable as described above. We need to do several things: compute the intersection of the ray with the hyperplane containing the "face," check to see whether the point lies within the simplex's boundaries (observe that this is a clipping problem), and see whether the normal vector points in the direction of the ray (making it visible).

We formulate this procedure by first writing

$$\vec{X}(t) = \vec{C} + t(\vec{P} - \vec{C})$$

for the position of a point on the camera ray, as illustrated in figure 7. Then we consider a single $(N-1)$-simplex of the tessellation to be described either by a known normal or by using the set of N points giving its vertices to define its normal via eq. (6); in either case, we can write the equation of any *other* point $\vec{x}$ lying within the simplex as

$$\hat{\mathbf{n}} \cdot (\vec{x} - \vec{x}_0) = 0 \ .$$

Plugging in the parametric ray equation, we solve for the point $\vec{X}(t)$ in the simplex that lies on the ray:

$$t = \frac{\hat{\mathbf{n}} \cdot \left(\vec{x}_0 - \vec{C}\right)}{\hat{\mathbf{n}} \cdot \left(\vec{P} - \vec{C}\right)} \ .$$

A useful generalization of ray-tracing to N-dimensions follows from the observation that a "thick ray" is cast into space by an open-ended simplex that is essentially a barycentric coordinate form with the restriction $0 \leq (1 - t_1 - t_2 - \ldots) \leq 1$ relaxed (see, e.g., (Hanson and Cross 1993)).

A planar ray such as that shown in figure 7 then has two parameters,

$$\vec{X}(t_1, t_2) = \vec{C} + t_1(\vec{P}_1 - \vec{C}) + t_2(\vec{P}_2 - \vec{C}) \,,$$

with obvious generalizations to volume rays, etc. Intersecting such a planar ray with an $(N-2)$-dimensional manifold (describable using $(N-2)$ barycentric parameters) results in N equations with N unknown parameters, and thus a unique *point* is determined as the mutual solution. In 3D, a plane intersects a line in one point, in 4D two planes intersect in a single point, while in 5D a plane intersects a volume in a point. Other generalizations, including rays that intersect particular geometries in lines and surfaces, can easily be constructed. For example, the intersection of a planar ray with the single hyperplane equation for a 3-manifold in 4D leaves one undetermined parameter, and is therefore a line.

♢ Conclusion ♢

Geometry is an essential tool employed in the creation of computer graphics images of everyday objects. Statistical data analysis, mathematics, and science, on the other hand, provide many problems where N-dimensional generalizations of the familiar 2D and 3D formulas are required. The N-dimensional formulas and insights into the nature of geometry that we have presented here provide a practical guide for extending computer graphics into these higher-dimensional domains.

♢ Appendix: Determinants and the Levi-Civita Symbol ♢

One of the unifying features that has permitted us throughout this treatment to extend formulas to arbitrary dimensions has been the use of *determinants*. But what if you encounter an expression involving determinants that has not been given here and you wish to work out its algebraic properties for yourself? In this appendix, we outline for the reader a useful mathematical tool for treating determinants, the Levi-Civita symbol. References for this are hard to locate; the author learned these techniques by apprenticeship while studying general relativity, but even classic texts like Møller (Møller 1972) contain only passing mention of the methods; somewhat more detail is given in hard-to-find sources such as (Efimov and Rozendorn 1975).

First we define two basic objects, the Kronecker delta, δ_{ij},

$$\begin{aligned} \delta_{ij} &= 1 \qquad & i = j \\ &= 0 \qquad & i \neq j \end{aligned}$$

and the Levi-Civita symbol, $\epsilon_{ijk\ldots}$, which is the totally antisymmetric pseudotensor with the properties

$$\begin{aligned} \epsilon_{ijk\ldots} &= 1 \qquad & i, j, k, \ldots \text{ in an even permutation of cyclic order} \\ &= -1 \qquad & i, j, k, \ldots \text{ in an odd permutation of cyclic order} \\ &= 0 \qquad & \text{when any two indices are equal.} \end{aligned}$$

All indices are assumed to range from 1 to N, e.g., $i = \{1, 2, \ldots, (N-1), N\}$, so that, for example, (1234,1342,4132,4321), are even permutations and (1324,2134,1243,4312) are odd permutations.

We can use the Kronecker delta to write the dot product between two N-dimensional vectors as a matrix product with the Kronecker delta representing the unit matrix,

$$\vec{A} \cdot \vec{B} = \sum_{i=1}^{N} \sum_{j=1}^{N} A_i \delta_{ij} B_j = \sum_{i=1}^{N} A_i \left(\sum_{j=1}^{N} \delta_{ij} B_j \right) = \sum_{i=1}^{N} A_i B_i \ , \tag{15}$$

and the Levi-Civita symbol to write the determinant of a matrix $|M|$ as

$$\det[M] \quad = \sum_{\text{all } i_k \text{ indices}} \epsilon_{i_1 i_2 \ldots i_N} M_{1,i_1} M_{2,i_2} \cdots M_{N,i_N} \ .$$

The fundamental formula for the product of two Levi-Civita symbols is:

$$\epsilon_{i_1 i_2 \ldots i_N} \epsilon_{j_1 j_2 \ldots j_N} \quad = \quad \det \begin{bmatrix} \delta_{i_1 j_1} & \delta_{i_1 j_2} & \cdots & \delta_{i_1 j_N} \\ \delta_{i_2 j_1} & \delta_{i_2 j_2} & \cdots & \delta_{i_2 j_N} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{i_N j_1} & \delta_{i_N j_2} & \cdots & \delta_{i_N j_N} \end{bmatrix} \ .$$

(Note that if we set $\{j_1 j_2 \ldots j_N\} = \{1, 2, \ldots, N\}$, the second Levi-Civita symbol reduces to $+1$, and the resulting determinant is an explicit realization of the antisymmetry of the Levi-Civita symbol itself as a determinant of Kronecker deltas!)

With this notation, the generalized cross product $\vec{N}$ of eq. (6), simplified by setting $\vec{x}_0 = 0$, can be written

$$\vec{N} \quad = \sum_{\text{all indices}} \epsilon_{i_1 i_2 \ldots i_{N-1} i_N} x_1^{(i_1)} x_2^{(i_2)} \cdots x_{N-1}^{(i_{N-1})} \hat{\mathbf{x}}^{(i_N)} \ ,$$

where $\hat{\mathbf{x}}^{(i_N)}$ are the unit vectors $(\hat{\mathbf{x}}, \hat{\mathbf{y}}, \ldots, \hat{\mathbf{w}})$ of the coordinate system. The dot product between the normal and another vector simply becomes

$$\vec{N} \cdot \vec{L} \quad = \sum_{\text{all indices}} \epsilon_{i_1 i_2 i_3 \ldots i_{N-1} i_N} x_1^{(i_1)} x_2^{(i_2)} x_3^{(i_3)} \cdots x_{N-1}^{(i_{N-1})} L^{(i_N)} \ .$$

The reader can verify that, in 2D, $N_k = \sum_{i=1}^{2} x^{(i)} \epsilon_{ik} = (-y, +x)$, and so on. We conclude with two examples of applications:

Rotations of Normals. Is the normal $\vec{N}$ a vector? *Almost*. To check this, we must rotate each column vector in the cross product formula using $x'^{(i)} = \sum_{j=1}^{N} R_{ij} x^{(j)}$ and compute the behavior of $\vec{N}$. Using the identity ((Efimov and Rozendorn 1975), p. 203),

$$\epsilon_{i_1 i_2 \ldots i_{N-1} i_N} \det[R] = \sum_{\text{all } j_k \text{ indices}} \epsilon_{j_1 j_2 \ldots j_{N-1} j_N} R_{j_1 i_1} R_{j_2 i_2} \cdots R_{j_{N-1} i_{N-1}} R_{j_N i_N} ,$$

we find

$$\begin{aligned} N'^{(i)} &= \sum_{\substack{\text{all indices} \\ \text{except } i}} \epsilon_{i_1 i_2 \ldots i_{N-1} i} R_{i_1 j_1} x_1^{(j_1)} R_{i_2 j_2} x_2^{(j_2)} \cdots R_{i_{N-1} j_{N-1}} x_{N-1}^{(j_{N-1})} \\ &= \sum_{j=1}^{N} R_{ij} N^{(j)} \det[R] . \end{aligned}$$

Therefore $\vec{N}$ is a *pseudotensor*, and behaves as a vector for ordinary rotations (which have $\det[R] = 1$), but changes sign if $[R]$ contains an odd number of reflections.

Contraction Formula. The contraction of two partial determinants of $(N-K)$ N-dimensional vectors can expanded in terms of products of Kronecker deltas as follows:

$$\sum_{i_{N-K+1} \ldots i_N} \epsilon_{i_1 i_2 \ldots i_{N-K} i_{N-K+1} \ldots i_N} \epsilon_{j_1 j_2 \ldots j_{N-K} i_{N-K+1} \ldots i_N} = K! \det \begin{bmatrix} \delta_{i_1 j_1} & \delta_{i_1 j_2} & \cdots & \delta_{i_1 j_{N-K}} \\ \delta_{i_2 j_1} & \delta_{i_2 j_2} & \cdots & \delta_{i_2 j_{N-K}} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{i_{N-K} j_1} & \delta_{i_{N-K} j_2} & \cdots & \delta_{i_{N-K} j_{N-K}} \end{bmatrix} .$$

The expression eq. (8) for the dot product of two normals is a special case of this formula.

Acknowledgment. This work was supported in part by NSF grant IRI-91-06389.

Bibliography

(Banchoff and Werner 1983) T. Banchoff and J. Werner. *Linear Algebra through Geometry*. Springer-Verlag, 1983.

(Banchoff 1990) Thomas F. Banchoff. *Beyond the Third Dimension: Geometry, Computer Graphics, and Higher Dimensions*. Scientific American Library, New York, NY, 1990.

(Brun et al. 1989) R. Brun, O. Couet, C. Vandoni, and P. Zanarini. *PAW – Physics Analysis Workstation, The Complete Reference*. CERN, Geneva, Switzerland, October 1989. Version 1.07.

(Chen et al. 1988) Michael Chen, S. Joy Mountford, and Abigail Sellen. A study in interactive 3-d rotation using 2-d control devices. In *Proceedings of Siggraph 88*, volume 22, pages 121–130, 1988.

(Coxeter 1991) H.S.M. Coxeter. *Regular Complex Polytopes*. Cambridge University Press, second edition, 1991.

(Edmonds 1957) A.R. Edmonds. *Angular Momentum in Quantum Mechanics*. Princeton University Press, Princeton, New Jersey, 1957.

(Efimov and Rozendorn 1975) N.V. Efimov and E.R. Rozendorn. *Linear Algebra and Multi-Dimensional Geometry*. Mir Publishers, Moscow, 1975.

(Feiner and Beshers 1990a) S. Feiner and C. Beshers. Visualizing n-dimensional virtual worlds with n-vision. *Computer Graphics*, 24(2):37–38, March 1990.

(Feiner and Beshers 1990b) S. Feiner and C. Beshers. Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds. In *Proceedings of UIST '90, Snowbird, Utah*, pages 76–83, October 1990.

(Francis 1987) G.K. Francis. *A Topological Picturebook*. Springer-Verlag, New York, 1987.

(Hanson and Cross 1993) A.J. Hanson and R.A. Cross. Interactive visualization methods for four dimensions. In *IEEE Visualization '93*, 1993. To be published.

Copyright © 1994 by Academic Press, Inc.
All rights of reproduction in any form reserved.
ISBN 0-12-336156-7

(Hanson and Heng 1992a) Andrew J. Hanson and Pheng A. Heng. Four-dimensional views of 3-D scalar fields. In Arie E. Kaufman and Gregory M. Nielson, editors, *Proceedings of Visualization 92*, pages 84–91, Los Alamitos, CA, October 1992. IEEE Computer Society Press.

(Hanson and Heng 1992b) Andrew J. Hanson and Pheng A. Heng. Illuminating the fourth dimension. *IEEE Computer Graphics and Applications*, 12(4):54–62, July 1992.

(Hanson 1992) Andrew J. Hanson. The rolling ball. In David Kirk, editor, *Graphics Gems III*, pages 51–60. Academic Press, San Diego, CA, 1992.

(Hocking and Young 1961) John G. Hocking and Gail S. Young. *Topology*. Addison-Wesley, 1961.

(Møller 1972) C. Møller. *The Theory of Relativity*. Oxford, Clarendon Press, 1972.

(Noll 1967) Michael A. Noll. A computer technique for displaying n-dimensional hyperobjects. *Communications of the ACM*, 10(8):469–473, August 1967.

(Phillips et al. 1993) Mark Phillips, Silvio Levy, and Tamara Munzner. Geomview: An interactive geometry viewer. *Notices of the Amer. Math. Society*, 40(8):985–988, October 1993. Available by anonymous ftp from geom.umn.edu, The Geometry Center, Minneapolis MN.

(Sommerville 1958) D.M.Y. Sommerville. *An Introduction to the Geometry of N Dimensions*. Reprinted by Dover Press, 1958.

◊ II.4

Rotations for N-Dimensional Graphics

Andrew J. Hanson
Computer Science Department
Indiana University
Bloomington, IN 47405
hanson@cs.indiana.edu

◊ Introduction ◊

In a previous Gem (Hanson 1994), "Geometry for N-Dimensional Graphics," we described a family of techniques for dealing with the geometry of N-dimensional models in the context of graphics applications. Here, we build on that framework to look in more detail at rotations in N-dimensional Euclidean space. In particular, we give a natural N-dimensional extension of the 3D rolling ball technique described in an earlier Gem (Hanson 1992), along with the corresponding analog of the Virtual Sphere method (Chen et al. 1988). Next, we touch on practical methods for specifying and understanding the parameters of N-dimensional rotations. Finally, we give the explicit 4D extension of 3D quaternion orientation splines.

For additional details and insights, we refer the reader to classic sources such as (Sommerville 1958,Coxeter 1991,Hocking and Young 1961,Efimov and Rozendorn 1975).

◊ The Rolling Ball in N Dimensions ◊

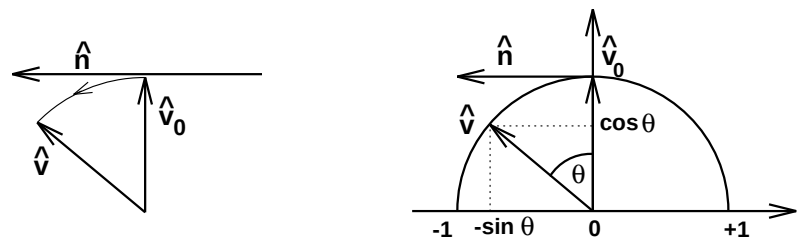
Basic Intuition of the Rolling Ball. The basic intuitive property of a rolling ball (or *tangent space*) rotation algorithm in any dimension is that it takes a unit vector $\hat{\mathbf{v}}_0 = (0, 0, \ldots, 0, 1)$ pointing purely in the N-th direction (towards the "north pole" of the ball) and tips it in the direction of an arbitrary unit vector $\hat{\mathbf{n}} = (n_1, n_2, \ldots, n_{N-1}, 0)$ lying in the $(N-1)$-plane tangent to the ball at the north pole, thus producing a new, rotated unit vector $\hat{\mathbf{v}}$, where

$$\hat{\mathbf{v}} = M_N \cdot \hat{\mathbf{v}}_0 = \hat{\mathbf{n}} \sin\theta + \hat{\mathbf{v}}_0 \cos\theta \ ,$$

as indicated schematically in Figure 1a. (Note: for notational simplicity, we choose to write the components of column vectors as horizontal lists.)

If we choose the convention that positive rotations are right-handed and progress counterclockwise, a positive rotation of the north pole actually tilts it into the negative direction of the remaining axis of the rotation plane. That is, if the 2D "rolling circle" acts on $\hat{\mathbf{v}}_0 = (0, 1)$ and

Copyright © 1995 by Academic Press, Inc.
All rights of reproduction in any form reserved.
ISBN 0-12-543457-X



(a) (b)

Figure 1. Tilting the "north pole" vector $\hat{\mathbf{v}}_0$ in the direction of the tangent vector $\hat{\mathbf{n}}$, as though rolling a ball by placing one's finger directly on the north pole and pulling in the direction $\hat{\mathbf{n}}$.

$\hat{\mathbf{n}} = (-1, 0)$ as shown in Figure 1b, then

$$\hat{\mathbf{v}} = M_2 \cdot \hat{\mathbf{v}}_0 = \hat{\mathbf{n}} \sin\theta + \hat{\mathbf{v}}_0 \cos\theta = (-\sin\theta,\, \cos\theta)\,,$$

where the rotation matrix M_2 can be written

$$\begin{aligned} M_2 &= \begin{bmatrix} \cos\theta & -\sin\theta \\ +\sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} c & -s \\ +s & c \end{bmatrix} \\ &= \begin{bmatrix} c & +n_x s \\ -n_x s & c \end{bmatrix}. \end{aligned} \tag{1}$$

If we choose a right-handed overall coordinate frame, the sign of $\hat{\mathbf{n}}$ will automatically generate the correct sign convention.

> *Synopsis:* Qualitatively speaking, if we imagine looking straight down at the north pole, the rolling ball *pulls* the unseen N-th component of a vector along the direction $\hat{\mathbf{n}}$ of the $(N-1)$-dimensional controller motion, bringing the unseen component gradually into view.

Implementation. In practice, we choose a radius R for the ball containing the object or scene to be rotated and move our controller (slider, 2D mouse, 3D mouse, ...) a distance r in the tangent direction $\hat{\mathbf{n}}$, as indicated in Figure 2a. Working from the simplified diagram in Figure 2b, we define $D^2 = R^2 + r^2$ and choose the rotation parameters $c = \cos\theta = R/D$ and $s = \sin\theta = r/D$.

For interactive systems, this choice has the particular advantage that, however rapidly the user moves the controller, $0 \le (r/D) < +1$, so $0 \le \theta < \pi/2$. Depending upon the desired interface behavior, an alternative choice would be to take $\theta = r/R$. This requires computing a trigonometric function instead of a square root, and may cause large discontinuities in orientation for large controller motion.

3D. The explicit 3D rolling ball formula can be derived starting from an arbitrary 2D mouse displacement $\vec{\mathbf{r}} = (x, y, 0) = (rn_x, rn_y, 0)$, where $n_x^2 + n_y^2 = 1$. Then one replaces Equation

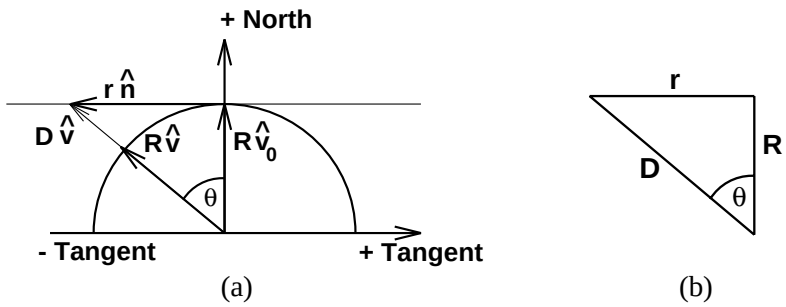


Figure 2. The notation used in implementing the rolling ball rotation model for N dimensions.

(1) with $n_x = +1$ by the analogous 3×3 matrix R_0 for (x, z) rotations and encloses this in a conjugate pair of rotations R_{xy} that transform the 2D mouse displacement $\vec{\mathbf{r}}$ into the strictly positive x-direction and back. Since even $\vec{\mathbf{r}} = (-1, 0, 0)$ is rotated to the positive x-direction before R_0 acts, all signs are correct. With the explicit matrices

$$R_{xy} = \begin{bmatrix} n_x & -n_y & 0 \\ n_y & n_x & 0 \\ 0 & 0 & 1 \end{bmatrix}, \; R_0 = \begin{bmatrix} c & 0 & +s \\ 0 & 1 & 0 \\ -s & 0 & c \end{bmatrix},$$

we find an alternative derivation of the formula in our earlier Gem (Hanson 1992):

$$\begin{aligned} M_3 &= R_{xy} R_0 (R_{xy})^{-1} \\ &= \begin{bmatrix} c + (n_y)^2(1-c) & -n_x n_y(1-c) & n_x s \\ -n_x n_y(1-c) & c + (n_x)^2(1-c) & n_y s \\ -n_x s & -n_y s & c \end{bmatrix} \\ &= \begin{bmatrix} 1 - (n_x)^2(1-c) & -n_x n_y(1-c) & n_x s \\ -n_x n_y(1-c) & 1 - (n_y)^2(1-c) & n_y s \\ -n_x s & -n_y s & c \end{bmatrix}. \end{aligned} \tag{2}$$

4D. The 4D case takes as input a 3D mouse motion $\vec{\mathbf{r}} = (x, y, z, 0) = (rn_x, rn_y, rn_z, 0)$, with $n_x^2 + n_y^2 + n_z^2 = 1$. Then one first transforms (n_y, n_z) into a pure y-component, rotates that result to yield a pure x-component, performs a rotation by θ in the (x, w)-plane, and reverses the first two rotations. Defining the required matrices as

$$R_{yz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{n_y}{r_{yz}} & -\frac{n_z}{r_{yz}} & 0 \\ 0 & \frac{n_z}{r_{yz}} & \frac{n_y}{r_{yz}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, R_{xy} = \begin{bmatrix} n_x & -r_{yz} & 0 & 0 \\ r_{yz} & n_x & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, R_0 = \begin{bmatrix} c & 0 & 0 & +s \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s & 0 & 0 & c \end{bmatrix}, \tag{3}$$

where $r_{yz}^2 = n_y^2 + n_z^2$, we find

$$\begin{aligned} M_4 &= R_{yz} R_{xy} R_0 (R_{xy})^{-1} (R_{yz})^{-1} \\ &= \begin{bmatrix} 1-(n_x)^2(1-c) & -(1-c)n_x n_y & -(1-c)n_x n_z & s n_x \\ -(1-c)n_x n_y & 1-(n_y)^2(1-c) & -(1-c)n_y n_z & s n_y \\ -(1-c)n_x n_z & -(1-c)n_y n_z & 1-(n_z)^2(1-c) & s n_z \\ -s n_x & -s n_y & -s n_z & c \end{bmatrix} . \end{aligned} \tag{4}$$

ND. The extension of this procedure to any dimension is accomplished by having the controller interface supply an $(N-1)$-dimensional vector $\vec{\mathbf{r}} = (r n_1, r n_2, \ldots, r n_{N-1}, 0)$ with $\vec{\mathbf{r}} \cdot \vec{\mathbf{r}} = r^2$ and $\hat{\mathbf{n}} \cdot \hat{\mathbf{n}} = 1$ and applying the rotation

$$\begin{aligned} M_N &= R_{N-2,N-1} R_{N-3,N-2} \cdots R_{1,2} R_0 (R_{1,2})^{-1} \cdots (R_{N-3,N-2})^{-1} (R_{N-2,N-1})^{-1} \\ &= \begin{bmatrix} 1-(n_1)^2(1-c) & -(1-c)n_2 n_1 & \cdots & -(1-c)n_{N-1} n_1 & s n_1 \\ -(1-c)n_1 n_2 & 1-(n_2)^2(1-c) & \cdots & -(1-c)n_{N-1} n_2 & s n_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -(1-c)n_1 n_{N-1} & -(1-c)n_2 n_{N-1} & \cdots & 1-(n_{N-1})^2(1-c) & s n_{N-1} \\ -s n_1 & -s n_2 & \cdots & -s n_{N-1} & c \end{bmatrix} \end{aligned} \tag{5}$$

Recall that the controller input $\vec{\mathbf{r}} = r\hat{\mathbf{n}}$ that selects the direction to "pull" also determines $c = \cos\theta = R/D$, $s = \sin\theta = r/D$, with $D^2 = R^2 + r^2$, or, alternatively, $\theta = r/R$.

♢ Controlling the Remaining Rotational Degrees of Freedom ♢

There are $N(N-1)/2$ parameters in a general N-dimensional orthogonal rotation matrix, one parameter for each possible pair of axes specifying a *plane of rotation* (the 3D intuition about "axes of rotation" does not extend simply to higher dimensions). The matrix M_N in Equation (5) has only $(N-1)$ independent parameters: we must now understand what happened to the other $(N-1)(N-2)/2$ degrees of freedom needed for arbitrary rotations.

In fact, the non-commutativity of the rotation group allows us to generate all the other rotations by *small circular motions* of the controller in the $(N-1)$-dimensional subspace of $\vec{\mathbf{r}} = r\hat{\mathbf{n}}$. Moving the controller in circles in the $(1,2)$-plane, $(1,3)$-plane, etc., of the $(N-1)$-dimensional controller space exactly generates the missing $(N-1)(N-2)/2$ rotations required to exhaust the full parameter space. In mathematical terms, the additional motions are generated by the commutation relations of the $SO(N)$ Lie algebra for $i, j = 1, \ldots, N-1$,

$$\begin{aligned} [R_{iN}, R_{jN}] &= \delta_{ij} R_{NN} - \delta_{jN} R_{iN} + \delta_{iN} R_{jN} - \delta_{NN} R_{ij} \\ &= -R_{ij} \ . \end{aligned}$$

The minus sign in the above equation means that *clockwise* controller motions in the (i, j)-plane inevitably produce *counterclockwise* rotations of the object, and vice-versa. Thus the philosophy (Hanson 1992) of achieving the full set of context-free rotation group transformations with a limited set of controller moves extends perfectly to N-dimensions. *Implementation Note*: In practice, the effectiveness of this technique varies considerably with the application; the size of the counter-rotation experienced may be relatively small for parameters that give appropriate spatial motion sensitivity with current 3D mouse technology.

Alternative Context Philosophies. The rolling ball interface is a *context-free* interface that allows the user of a virtual reality application to ignore the absolute position of the controller and requires no supplementary cursor context display; thus one may avoid distractions that may disturb stereography and immersive effects in a virtual reality environment. However some applications are better adapted to *context-sensitive* interfaces like the Arcball method (Shoemake 1994) or the Virtual Sphere approach (Chen et al. 1988). The Virtual Sphere approach in particular can be straightforwardly extended to higher dimensions by using the rolling ball equations inside a displayed spatial context (typically a sphere) and changing over to an $(N-1)$-dimensional rolling ball outside the context; that is, as the controller approaches and passes the displayed inner domain context sphere, the rotation action changes to one that leaves the N-th coordinate fixed but changes the remaining $(N-1)$ coordinates as though an $(N-1)$-dimensional rolling ball controller were attached to the nearest point on the sphere. Similar flexibility can be achieved by using a different controller state to signal a discrete rather than a continuous context switch to the $(N-1)$-dimensional controller.

♢ Handy Formulas for N-Dimensional Rotations ♢

For some applications the incremental orientation control methods described above are not as useful as knowing a single matrix for the entire N-dimensional orientation frame for an object. We note three ways to represent such an orientation frame:

Columns are new axes. One straightforward construction simply notes that if the default coordinate frame is represented by the orthonormal set of unit vectors $\hat{\mathbf{x}}_1 = (1, 0, \ldots, 0)$, $\hat{\mathbf{x}}_2 = (0, 1, 0, \ldots, 0)$, ..., $\hat{\mathbf{x}}_N = (0, \ldots, 0, 1)$, and the desired axes of the new (orthonormal) coordinate frame are known to be $\hat{\mathbf{a}}_1 = (a_1^{(1)}, a_1^{(2)}, \ldots, a_1^{(N)})$, $\hat{\mathbf{a}}_2$, ..., $\hat{\mathbf{a}}_N$, then the rotation matrix that transforms any vector to that frame just has the new axes as its columns:

$$M = \left[\begin{array}{cccc} \hat{\mathbf{a}}_1 & \hat{\mathbf{a}}_2 & \cdots & \hat{\mathbf{a}}_N \end{array}\right] .$$

The orthonormality constraints give M the required $N(N-1)/2$ degrees of freedom.

Concatenated subplane rotations. Rotations in the plane of a pair of coordinate axes $(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j)$, $i, j = 1, \ldots, N$ can be written as the block matrix

$$R_{ij}(\theta_{ij}) = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cos\theta_{ij} & 0 & \cdots & 0 & -\sin\theta_{ij} & \cdots & 0 \\ 0 & \cdots & 0 & 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & \sin\theta_{ij} & 0 & \cdots & 0 & \cos\theta_{ij} & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{6}$$

and thus the $N(N-1)/2$ distinct $R_{ij}(\theta_{ij})$ may be concatenated in some order to produce a rotation matrix such as

$$M = \prod_{i<j} R_{ij}(\theta_{ij})$$

with $N(N-1)/2$ degrees of freedom parametrized by $\{\theta_{ij}\}$. However, since the matrices R_{ij} do not commute, different orderings give different results and it is difficult to intuitively understand the global rotation. In fact, as is the case for 3D Euler angles, one may even repeat some matrices (with distinct parameters) and omit others, and still not miss any degrees of freedom.

Quotient Space Decomposition. Another useful decomposition relies on the classic quotient property of the topological spaces of the orthogonal groups (Helgason 1962),

$$SO(N)/SO(N-1) = S^{N-1}\,, \tag{7}$$

where S^K is a K-dimensional topological sphere. In practical terms, this means that the $N(N-1)/2$ parameters of $SO(N)$, the mathematical group of N-dimensional orthogonal rotations, can be viewed as a nested family of points on spheres. The 2D form is the matrix (1) parameterizing the points on the circle S^1; the 3D form reduces to the standard matrix

$$M_3(\theta, \hat{\mathbf{n}}) = \begin{bmatrix} c + (n_1)^2(1-c) & n_1 n_2(1-c) - s n_3 & n_3 n_1(1-c) + s n_2 \\ n_1 n_2(1-c) + s n_3 & c + (n_2)^2(1-c) & n_3 n_2(1-c) - s n_1 \\ n_1 n_3(1-c) - s n_2 & n_2 n_3(1-c) + s n_1 & c + (n_3)^2(1-c) \end{bmatrix} \tag{8}$$

where the two free parameters of $\hat{\mathbf{n}} \cdot \hat{\mathbf{n}} = (n_1)^2 + (n_2)^2 + (n_3)^2 = 1$ describe a point on the 2-sphere. These two parameters plus a third from the S^1 described by $c^2 + s^2 = 1$ (i.e., $c = \cos\theta$, $s = \sin\theta$) yield the required total of three free parameters equivalent to the three Euler angles. The 4D and higher forms are already too unwieldy to be conveniently written as single matrices.

♢ Interpolating N-Dimensional Orientation Frames ♢

To define a uniform-angular-velocity interpolation between two N-dimensional orientation frames, we might consider independently interpolating each angle in Equation (6), or we might take the quotient space decomposition given by the hierarchy of points on the spheres $(S^{N-1}, \ldots, S^2, S^1)$ and apply a constant angular velocity spherical interpolation to each spherical point in each successive dimension using the "Slerp"

$$\hat{\mathbf{n}}_{12}(t) = \text{Slerp}(\hat{\mathbf{n}}_1, \hat{\mathbf{n}}_2, t) = \hat{\mathbf{n}}_1 \frac{\sin((1-t)\theta)}{\sin(\theta)} + \hat{\mathbf{n}}_2 \frac{\sin(t\theta)}{\sin(\theta)}$$

where $\cos\theta = \hat{\mathbf{n}}_1 \cdot \hat{\mathbf{n}}_2$. (This formula is simply the result of applying a Gram-Schmidt decomposition while enforcing unit norm in any dimension.)

Either of these often achieves the goal of smooth appearance, but the solutions are neither unique nor mathematically compelling, since the curve is not guaranteed to be a geodesic in $SO(N)$.

The specification of geodesic curves in $SO(N)$ is a difficult problem in general (Barr et al. 1992); fortunately, the two most important cases for interactive systems, $N = 3$ and $N = 4$, have elegant solutions using the covering or "Spin" groups. For $SO(3)$, geodesic interpolations and suitable corresponding splines are definable using Shoemake's quaternion splines (Shoemake 1985), which can be simply formulated using Slerps on S^3 as follows: let $\hat{\mathbf{n}}$ be a unit 3-vector, so that

$$q_0 = \cos(\theta/2), \quad \vec{\mathbf{q}} = \hat{\mathbf{n}} \sin(\theta/2)$$

is automatically a point on S^3 due to the constraint $(q_0)^2 + (q_1)^2 + (q_2)^2 + (q_3)^2 = 1$. Then each point on S^3 corresponds to an $SO(3)$ rotation matrix

$$R_3 = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{bmatrix} \tag{9}$$

which the reader can verify reduces exactly to the nested-sphere form in Equation (8). Note that the quaternions q and $-q$ each correspond to the same 3D rotation. Slerping q generates sequences of matrices $R_3(t)$ that are geodesic interpolations. Arbitrary splines can be defined using the method of Schlag (Schlag 1991).

Quaternions in Four Dimensions. In four dimensions, the correspondence between the rotation group $SO(4)$ and the spin group Spin(4) that is its double covering may be computed by extending quaternion multiplication to act not just on 3-vectors ("pure" quaternions) $v = (0, \vec{\mathbf{V}})$, but on full 4-vector quaternions v^μ in the following way:

$$\sum_{\nu=0}^{3} R^\mu_\nu v^\nu = q \cdot v^\mu \cdot p^{-1} .$$

We thus find that the general double-quaternion parameterization for 4D rotation matrices takes the form

$$R_4 = \begin{bmatrix} q_0p_0 + q_1p_1 + q_2p_2 + q_3p_3 & q_1p_0 - q_0p_1 - q_3p_2 + q_2p_3 & q_2p_0 - q_0p_2 - q_1p_3 + q_3p_1 & q_3p_0 - q_0p_3 - q_2p_1 + q_1p_2 \\ -q_1p_0 + q_0p_1 - q_3p_2 + q_2p_3 & q_0p_0 + q_1p_1 - q_2p_2 - q_3p_3 & q_1p_2 + p_1q_2 - p_0q_3 - q_0p_3 & q_1p_3 + p_1q_3 + p_0q_2 + q_0p_2 \\ -q_2p_0 + q_0p_2 - q_1p_3 + q_3p_1 & q_1p_2 + q_2p_1 + q_0p_3 + q_3p_0 & q_0p_0 + q_2p_2 - q_1p_1 - q_3p_3 & q_2p_3 + q_3p_2 - q_0p_1 - q_1p_0 \\ -q_3p_0 + q_0p_3 - q_2p_1 + q_1p_2 & q_1p_3 + q_3p_1 - q_0p_2 - q_2p_0 & q_2p_3 + q_3p_2 + q_1p_0 + p_0q_1 & q_0p_0 + q_3p_3 - q_1p_1 - q_2p_2 \end{bmatrix} . \quad (10)$$

One may check that Equation (9) is just the lower right-hand corner of the degenerate $p = q$ case of Equation (10).

Shoemake-style interpolation between two distinct 4D frames is now achieved by applying the desired Slerp-based interpolation method independently to a set of quaternion coordinates $q(t)$ on one three-sphere, and to a separate set of quaternion coordinates $p(t)$ on another. The resulting matrix $R_4(t)$ gives geodesic interpolations for simple Slerps, and can be used as the basis for corresponding spline methods (Schlag 1991,Barr et al. 1992). Analogs of the $N = 3$ and $N = 4$ approaches for general N involve computing Spin(N) geodesics and thus are quite complex.

Controls. As pointed out in (Shoemake 1994), the Arcball controller can be adapted with complete faithfulness of spirit to the 4D case, since one can pick *two* points in a three-sphere to specify an initial 4D frame, and then pick *two more* points in the three-sphere to define the current 4D frame. Equation (10) gives the complete form of the effective 4D rotation. Alternately, one can replace the 4D rolling ball or Virtual Sphere controls described earlier by a pair (or more) of 3D controllers (Hanson 1992).

Acknowledgment. This work was supported in part by NSF grant IRI-91-06389.

♢ Bibliography ♢

(Barr et al. 1992) Alan H. Barr, Bena Currin, Steven Gabriel, and John F. Hughes. Smooth interpolation of orientations with angular velocity constraints using quaternions. *Computer Graphics*, 26(2):313–320, 1992.

(Chen et al. 1988) Michael Chen, S. Joy Mountford, and Abigail Sellen. A study in interactive 3-D rotation using 2-D control devices. In *Proceedings of Siggraph 88*, volume 22, pages 121–130, 1988.

(Coxeter 1991) H.S.M. Coxeter. *Regular Complex Polytopes*. Cambridge University Press, second edition, 1991.

(Efimov and Rozendorn 1975) N.V. Efimov and E.R. Rozendorn. *Linear Algebra and Multi-Dimensional Geometry*. Mir Publishers, Moscow, 1975.

(Hanson 1992) Andrew J. Hanson. The rolling ball. In David Kirk, editor, *Graphics Gems III*, pages 51–60. Academic Press, 1992.

(Hanson 1994) Andrew J. Hanson. Geometry for n-dimensional graphics. In Paul Heckbert, editor, *Graphics Gems IV*, pages 149–170. Academic Press, 1994.

(Helgason 1962) Sigurdur Helgason. *Differential Geometry and Symmetric Spaces*. Academic Press, New York, 1962.

(Hocking and Young 1961) John G. Hocking and Gail S. Young. *Topology*. Addison-Wesley, 1961.

(Schlag 1991) John Schlag. Using geometric constructions to interpolate orientations with quaternions. In James Arvo, editor, *Graphics Gems II*, pages 377–380. Academic Press, 1991.

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

(Shoemake 1994) Ken Shoemake. Arcball rotation control. In Paul Heckbert, editor, *Graphics Gems IV*, pages 172–192. Academic Press, 1994.

(Sommerville 1958) D.M.Y. Sommerville. *An Introduction to the Geometry of N Dimensions*. Reprinted by Dover Press, 1958.

Visualizing Quaternion Rotation

JOHN C. HART
Washington State University

GEORGE K. FRANCIS
University of Illinois, Urbana
and
LOUIS H. KAUFFMAN
University of Illinois, Chicago

Quaternions play a vital role in the representation of rotations in computer graphics, primarily for animation and user interfaces. Unfortunately, quaternion rotation is often left as an advanced topic in computer graphics education due to difficulties in portraying the four-dimensional space of the quaternions. One tool for overcoming these obstacles is the quaternion demonstrator, a physical visual aid consisting primarily of a belt. Every quaternion used to specify a rotation can be represented by fixing one end of the belt and rotating the other. Multiplication of quaternions is demonstrated by the composition of rotations, and the resulting twists in the belt depict visually how quaternions interpolate rotation.

This article introduces to computer graphics the exponential notation that mathematicians have used to represent unit quaternions. Exponential notation combines the angle and axis of the rotation into a concise quaternion expression. This notation allows the article to present more clearly a mechanical quaternion demonstrator consisting of a ribbon and a tag, and develop a computer simulation suitable for interactive educational packages. Local deformations and the belt trick are used to minimize the ribbon's twisting and simulate a natural-appearing interactive quaternion demonstrator.

Categories and Subject Descriptors: I.3.5 [**Computational Geometry and Object Modeling**]: Hierarchy and Geometric Transformations; I.3.6 [**Methodology and Techniques**]: Graphics Data Structures and Data Types

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Deformation, education, orientation interpolation, quaternions, rotation, visualization

---

J. C. Hart is supported in part by the NSF under a Research Initiation Award CCR-9309210. Equipment was provided by the Imaging Research Laboratory, which is supported by the NSF under grant CDA-9121675.
Authors' addresses: J. C. Hart, School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752; email: hart@eecs.wsu.edu; G. K. Francis, Department of Mathematics, University of Illinois at Urbana-Champaign, Urbana, IL 61801; email: gfrancis@math.uiuc.edu; L. H. Kauffman, Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago, Chicago, IL 60680; email: u10451@uicvm.bitnet.
Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.
© 1994 ACM 0730-0301/94/0700-0256 $03.50

ACM Transactions on Graphics, Vol. 13, No. 3, July 1994, Pages 256–276.

1. INTRODUCTION

The method of specifying rotations and orientations of coordinate systems via unit quaternions was formally introduced to the computer graphics community by the publication of Shoemake [1985]. Quaternions were used in graphics programming informally mostly by geometers because Sir William Rowan Hamilton's [Hamilton 1866] beautiful invention is not regularly taught in college. Quaternions encode rotations by four real numbers (or two complex numbers), whereas the linear representation of these transformations as 3×3 matrices requires nine. Moreover, Hamilton impressed explicit geometrical meaning into every detail of his algebraic system, which guides intuition and facilitates implementation [Francis and Kauffman 1994].

Interpolating the quaternionic representation of a sequence of rotations is more natural than doing so for the familiar Euler angles, such as yaw, pitch, and roll. The quaternions occupy a smooth, seamless, isotropic space which is a generalization of the surface of a sphere. Thus, there is no need for special care to avoid singularities, such as gimbal lock, where two rotation axes collapse into one and thus make the interpolation irreversible.

Bezier curves were used in Shoemake [1985] to spline the quaternions representing rotations, while Barr et al. [1992] used energy-minimizing curves for demonstrably smoother motions. Quaternions provide an easy mechanism for specifying an arbitrary rotation about an arbitrary axis. This has long been exploited in keyboard user interfaces, and most recently for specifying 3-dimensional rotations with a 2-dimensional mouse [Hanson 1992; Shoemake 1992].

1.1 Overview

This article builds on previous work in quaternion rotation to derive an implementation of the quaternion demonstrator. The first half of the article summarizes various recent works on the quaternions. Section 2 reviews the quaternion representation of three-dimensional rotation, based on Shoemake [1985] and Francis and Kauffman [1994], and describes the quaternion demonstrator, as devised originally in Kauffman [1987; 1991]. Section 3 describes the belt trick, summarizing Kauffman [1991] and Francis and Kauffman [1994] and explaining the mathematics behind the animation "Air on the Dirac Strings" [Sandin et al. 1993].

These sections form the basis for this article's original contribution, found in the second half of the article. Beginning with Section 4, techniques from differential geometry model the quaternion demonstrator, regulating the twists and motions of the belt. Section 5 describes the resulting implementation and outlines directions for further research.

1.2 Background

An object $\mathcal{O}$ is assumed to be defined with respect to some canonical coordinate frame. The *orientation* of $\mathcal{O}$ is represented by a rotation that takes the object from its canonical coordinate frame to its current state.

This article relies heavily on the deformation techniques developed in Barr [1984]. We use globally and locally specified deformations. A *globally specified deformation* alters explicitly the positions of points in an object whereas a *locally specified deformation* affects the tangent space of an object, and new positions result only after an integration over the deformed tangent space.

2. THE QUATERNIONS

The four-dimensional space, $\mathbb{H}$, of quaternions is spanned by the real axis, and three further orthogonal axes, spanned by vectors $\mathbf{i}, \mathbf{j}, \mathbf{k}$, called the *principal imaginaries*, which obey Hamilton's rules

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1. \tag{1}$$

These imaginaries signify the three-dimensional vectors

$$\begin{aligned} \mathbf{i} &= (1,0,0), \\ \mathbf{j} &= (0,1,0), \\ \mathbf{k} &= (0,0,1). \end{aligned}$$

Multiplication of these imaginaries resembles a cross product

$$\begin{aligned} &\mathbf{ij} = \mathbf{k}, \quad \mathbf{jk} = \mathbf{i}, \quad \mathbf{ki} = \mathbf{j}, \\ &\mathbf{ji} = -\mathbf{k}, \quad \mathbf{kj} = -\mathbf{i}, \quad \mathbf{ik} = -\mathbf{j} \end{aligned} \tag{2}$$

and is clearly noncommutative. Quaternion multiplication causes rotation: multiplication on the right by $\mathbf{j}$ causes a 90 degree rotation in four-dimensional space, rotating the $\mathbf{i}$ axis into the $\mathbf{k}$ axis, and rotating the $\mathbf{k}$ axis into the $-\mathbf{i}$ axis. Quaternion multiplication differs from the cross product in that $\mathbf{ii} = \mathbf{jj} = \mathbf{kk} = -1$ whereas $\mathbf{i} \times \mathbf{i} = \mathbf{j} \times \mathbf{j} = \mathbf{k} \times \mathbf{k} = 0$.

A quaternion $q = r + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ consists of a real part r and a pure part $x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ [Hamilton 1866]. We will call quaternions with zero real part ($r = 0$) *pure quaternions*. Pure quaternions will also be simultaneously represented as a column vector

$$\mathbf{v} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}. \tag{3}$$

Under this notation, the same symbol can simultaneously represent both a vector and a pure quaternion, depending on its context. For example,

$$\mathbf{v}^2 = -\mathbf{v} \cdot \mathbf{v} \tag{4}$$

because the LHS of (4) treats $\mathbf{v}$ as a pure quaternion whereas the RHS of (4) treats $\mathbf{v}$ as a vector.

Let $q_1 = a_1 + \mathbf{v}_1$ and $q_2 = a_2 + \mathbf{v}_2$ be two quaternions. Their sum is

$$q_1 + q_2 = (a_1 + a_2) + (\mathbf{v}_1 + \mathbf{v}_2),$$

and their product is

$$q_1 q_2 = a_1 a_2 - \mathbf{v}_1 \cdot \mathbf{v}_2 + a_1 \mathbf{v}_2 + a_2 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2.$$

The quaternion $q = a + \mathbf{v}$ also decomposes into $a + b\mathbf{u}$ which resembles a complex number, where the imaginary $\mathbf{u}$ is a unit three-vector

$$\mathbf{u} = \begin{pmatrix} x/b \\ y/b \\ z/b \end{pmatrix} = \frac{x}{\|\mathbf{v}\|}\mathbf{i} + \frac{y}{\|\mathbf{v}\|}\mathbf{j} + \frac{z}{\|\mathbf{v}\|}\mathbf{k},$$

such that $\|\mathbf{u}\| = 1$, and x, y, z are the same coordinates used in (3). The pure unit-magnitude quaternion $\mathbf{u}$ resembles the imaginary $\mathbf{i}$ from the complex plane in that $\mathbf{u}^2 = -1$.

Let $q = a + b\mathbf{u}$ be a quaternion. Its *conjugate* is $\bar{q} = a - b\mathbf{u}$, and its *magnitude* is

$$\|q\| = q\bar{q} = \bar{q}q = \sqrt{a^2 + b^2}. \tag{5}$$

2.1 Quaternion Rotation

Rotations in computer graphics are typically represented by quaternions of unit magnitude [Shoemake 1985], which we will call *unit quaternions*. The unit quaternions $\{q : \|q\| = 1\}$ form a hypersphere $\mathbb{S}^3 \subset \mathbb{H}$. In particular, for any unit quaternion $q \in \mathbb{S}^3$, (5) implies

$$q^{-1} = \bar{q}. \tag{6}$$

In other words, to invert a unit quaternion, we simply negate its pure part.

In Shoemake [1985], a rotation of θ about the axis $\mathbf{u}$ was represented as the unit quaternion

$$q = \cos\frac{1}{2}\theta + \sin\frac{1}{2}\theta\mathbf{u}$$

which matches the complex-like form of a quaternion $q = a + b\mathbf{u}$, where a is the real component and b the imaginary component along the new imaginary axis specified by the unit vector (pure quaternion) $\mathbf{u}$. The abbreviation $e^{\mathbf{i}\theta} = \cos\theta + \mathbf{i}\sin\theta$, borrowed from complex analysis, has a long history of use in the engineering sciences. We can similarly represent the aforementioned unit quaternion q more concisely using exponential notation as

$$q = e^{\frac{1}{2}\theta\mathbf{u}}.$$

In the same manner that engineers read the expression $e^{\mathbf{i}\theta}$, the reader should likewise understand the notation $e^{(1/2)\theta\mathbf{u}}$ not as e to some imaginary power but simply as the quaternion that represents a rotation of θ about the axis $\mathbf{u}$.

Exponential notation was chosen to represent quaternion rotations herein to promote consistency between the converging fields of computer graphics and mathematics. Such quaternion exponential notation has a lengthy history in mathematics (e.g., as the so-called exponential map in different geometry [Spivak 1965]). However, since quaternion multiplication is noncommutative, likewise quaternion exponentiation does not in general follow the rules of real or complex exponentiation (e.g., $e^{(1/2)\theta_1\mathbf{u}_1}e^{(1/2)\theta_2\mathbf{u}_2} \neq$

$e^{(1/2)(\theta_1 \mathbf{u}_1 + \theta_2 \mathbf{u}_2)}$). Quaternion exponentiation is formally defined in Francis and Kauffman [1994], along with a discussion of its properties illustrated by several examples.

Given a unit quaternion q that represents a rotation, the question remains of how to apply this rotation to an arbitrary vector (pure quaternion) $\mathbf{v} \in \mathbb{R}^3$. From Shoemake [1985], we find two results: a function $R(q)$ that returns the 3×3 (nonhomogeneous) transformation matrix corresponding to the rotation represented by q, and a two-to-one correspondence between unit quaternions $\mathbb{S}^3$ and the space of all rotations $\mathbb{SO}(3)$ (the group of special-orthogonal 3×3 matrices). As a consequence of these two results, we have

$$R(q)\mathbf{v} \equiv q\mathbf{v}q^{-1}. \tag{7}$$

The LHS of (7) treats $\mathbf{v}$ as a column vector and yields a new column vector of the same length by left-multiplying the special-orthogonal matrix returned by the function R. The RHS of (7) treats $\mathbf{v}$ as a pure quaternion and yields a new pure quaternion of the same magnitude. In fact, both apply the rotation represented by the unit quaternion q to $\mathbf{v}$. We denote rotations with the notation on the LHS of (7), but implement rotations more efficiently using the formula on the RHS of (7). (Since $q \in \mathbb{S}^3$, q^{-1} simplifies to $\bar{q}$.)

Hence, the otherwise complicated procedure of rotating a vector about an arbitrary axis simplifies in our notation to

$$R(e^{\frac{1}{2}\theta \mathbf{u}})\mathbf{v}$$

which rotates $\mathbf{v} \in \mathbb{R}^3$ about the $\mathbf{u} \in \mathbb{S}^2$ axis by an angle of θ [Francis and Kauffman 1994].

2.2 The Quaternion Demonstrator

The *quaternion demonstrator* [Kauffman 1987; 1991] is a mechanical unit quaternion multiplier. It consists primarily of a ribbon, called the *belt*, with one end fixed and the other end free. Fastened to the free end of the belt is a rectangle, called the *tag*. (An alternative demonstrator was discovered by Kauffman and E. Oshins [Kauffman 1991] that uses only one human arm.)

The orientations of the tag, along with the twists in the belt, represent the unit quaternions. The tag is inscribed with labels indicating the quaternion it currently represents. The top side of the tag is inscribed with a 1 and an upside-down **j**. Its bottom side is inscribed with a **k** and an upside-down **i**, in the fashion suggested by Figure 1. (If you wish to follow along using your right arm, your right hand will be the tag. Your fingerprints are 1; your palm is **j**; your fingernails are **k**; and the back of your hand is **i**. As a tag, your hand will represent the quaternion associated with the part of your hand facing up and toward the same direction you are facing.)

First we orient the quaternion coordinate frame such that the imaginaries form a right-handed coordinate system where **i** points right, **j** up, and **k** toward the viewer. The belt of the quaternion demonstrator is embedded in the $\mathbf{i} - \mathbf{k}$ plane centered along the **k** axis with the fixed end at the origin and the tag at $+\mathbf{k}$. The eyepoint is assumed to be somewhere in the positive $\mathbf{j} - \mathbf{k}$

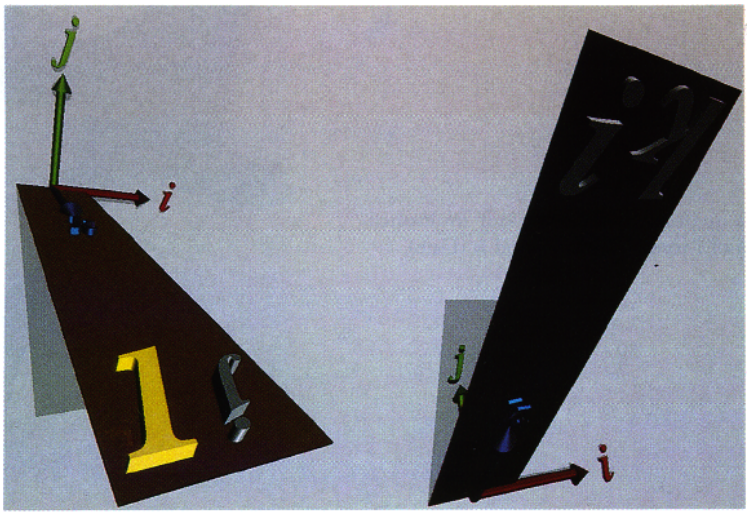


Fig. 1. The quaternion demonstrator in the 1 state (top and bottom views). This figure uses the computer simulation of the quaternion demonstrator, which does not use a tag. Instead it superimposes the 1, **i**, **j**, and **k** labels directly onto the tag end of the belt.

quadrant. (In your arm's coordinate system, the origin is your right shoulder; the **i** axis points toward your left shoulder; the **j** axis points up; and the **k** axis points in the direction you are facing.)

The canonical state of the demonstrator consists of the untwisted belt in this configuration. This state represents the value 1, and is shown in Figure 1. (Extending your arm out in front of you with your palm up puts your arm in the 1 state.)

Rotating the tag by 180 degrees with respect to the **i** axis puts the tag underneath the belt. This state represents the value **i** as can be read on the tag end of the belt (Figure 2 red). Call this rotation a "flip." (Keeping your arm and wrist straight, rotate 180 degrees about your shoulder's axis by dropping your arm to your waist and raise it back up behind you such that your palm is facing down. The back of your hand is facing up and toward the front, so your arm now represents the quaternion **i**.)

Reset the system to the canonical state 1. Rotating the tag by 180 degrees with respect to the **j** axis puts the tag to the right of the fixed end of the belt. This state represents the value **j** as can also be read on the tag end of the belt (Figure 2 green). Call this rotation a "spin." (From the 1 state, rotate your arm horizontally until you touch your chest with your fingertips. Your palm faces up and toward the front, and your arm now represents the quaternion **j**.)

Resetting to 1 again and rotating the tag − 180 degrees with respect to the **k** axis flips the tag over. This state represents the value **k**, as indicated by the upright **k** on the tag end of the belt (Figure 2 blue). Call this rotation a "turn." (From the 1 state, turn your wrist about your arm's axis until your

Fig. 2. The quaternion demonstrator multiplication by each of **i** (red), **j** (green), and **k** (blue).

palm is facing down. Your fingernails face up and to the front, and your arm now represents the quaternion **k**.)

The fact that *positive* **k** is obtained by a negative rotation in the right-handed coordinate system is an artifact of the belt-centered coordinate system. Centering a right-handed coordinate system on the tag with **k** extending along the belt, **j** up, and **i** to the right yields rotations consistent with the right-handed coordinate system. (For the quaternion demonstrator's task of teaching quaternion multiplication the simplicity of the belt-centered coordinate system outweighs the familiarity of the right-handed rotation rules of the tag-centered coordinate system.)

The negative quaternion imaginaries are likewise produced by the opposite flips, twists, and turns, respectively. Although the labels on the tag have no signs, we can tell a positive imaginary from a negative imaginary by the direction of the twists in the belt. (The quaternion $-\mathbf{i}$ is represented from the 1 state by bending your arm at the elbow -180 degrees about your shoulder's axis, with your palm facing down as if to pat yourself on the back. The quaternion $-\mathbf{j}$ is represented by spinning your hand horizontally -180 degrees about the up axis into the position a waiter would use to hold a tray. The quaternion $-\mathbf{k}$ is physiologically impossible to represent in the arm coordinate system.)

In this system, multiplication is represented by the composition of corresponding rotations of the tag. For example, to demonstrate $\mathbf{ij} = \mathbf{k}$, we find that a flip followed by a spin is equivalent to a turn (after a little translation of the tag—moving the tag does not rotate it and does not change the state of the demonstrator). A reverse turn (multiplication by $-\mathbf{k}$) returns the system to its original state. (From the **1** state, flip 180 degrees around your shoulder's axis by dropping your arm to your waist and raising it up again behind you with your palm facing down into the **i** state. Using your shoulder, spin about **j**

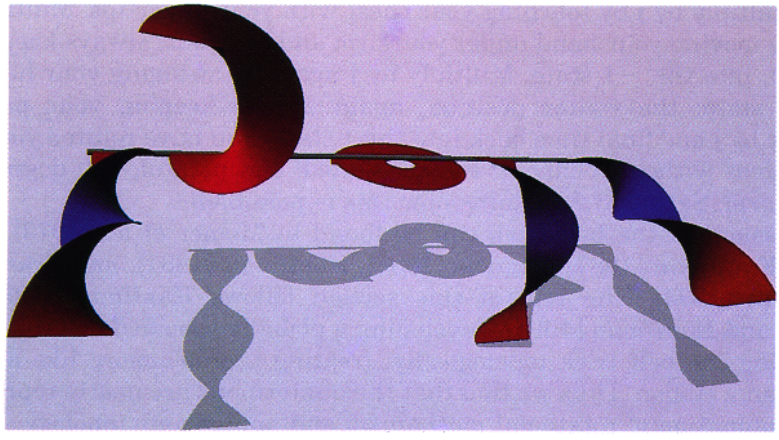


Fig. 3. The belt trick equation: $\mathbf{k}^2 = (-\mathbf{k})^2$ proven visually using a quaternion demonstrator. The ends of the belt remain parallel through the transformation. Each step of the transformation represents the value -1.

180 degrees by swinging your arm out and back to the front with your palm down. Your arm is now in front of you, palm down, representing $\mathbf{k}$.)

By (1), the -1 state is achieved by two flips ($\mathbf{i}^2$), two spins ($\mathbf{j}^2$), two turns ($\mathbf{k}^2$), or a flip-spin-turn ($\mathbf{ijk}$). Each of these operations returns the tag to its original orientation but puts a 360 degree twist in the belt. (From the 1 state, twist your wrist 360 degrees about the arm's axis until your palm faces up again ($\mathbf{k}^2$). Your fingerprints are facing up and in front, but with a full 360 degree twist in your arm, which now represents -1.)

We can also create -1 by two reverse flips ($(-\mathbf{i})^2$) which also returns the tag to its original orientation but puts the opposite (-360 degrees) twist in the belt (more on this in the next section.)

3. THE BELT TRICK

In the quaternion demonstrator, -1 can be represented as $\mathbf{k}^2$ by two turns which return the tag to its original state but cause a 360 degree twist in the belt. Two reverse turns represents -1 also, as $(-\mathbf{k})^2$, but cause a -360 degree twist in the belt. The quaternion demonstrator has (at least) two distinct representations for -1.

These two representations are equivalent, however. Consider the demonstrator in the $\mathbf{k}^2$ state, after two turns. Without rotating the tag, move the tag in a positive 360 degree arc around the fixed end of the belt. This belt trick changes the 360 degree twist in the belt into a -360 degree twist, and proves the Belt Trick Equation shown in Figure 3.

(You can perform a variation of the belt trick called the plate trick, shown live in Sandin et al. [1993], with your arm by representing $\mathbf{j}^4 = 1$. From the 1

state, multiply by **j** by touching your chest with your fingertips. Multiply by **j** again by moving your hand under your arm and back out, always keeping the palm up, into the -1 state. Multiply by **j** again by swinging your hand into the $-\mathbf{j}$ state, the waiter position, again always keeping your palm up. Multiply by **j** one final time back into the 1 state. You have rotated your hand 720 degrees without incurring the associated, and painful, 720 degree twist in your arm. Section 3.2 explains how this is possible.)

Discussions of the belt trick can be found in Misner et al. [1973], Bolker [1973], Kauffman [1987], Francis [1987], Kauffman [1991], and Francis and Kauffman [1994]. The rest of this section follows Kauffman [1991] and Francis and Kauffman [1994], developing a globally specified deformation for simulating the belt trick topologically, treating the belt more like a rubber band than a ribbon. Thus we find that the quaternions are neatly represented by a combination of rotational mechanism and appropriate topology.

3.1 The Belt Trick Deformation

Whereas Figure 3 demonstrates the belt trick using the quaternion demonstrator, where the belt is fixed at one end and free at the tag, the following discussion uses an equivalent but alternate construction. This new system consists of a belt connecting two concentric spheres, and is described quantitatively as follows.

Let

$$S(r) = r\mathbb{S}^2 = \{\mathbf{x} : \|\mathbf{x}\| = r\}$$

specify a sphere of radius r centered at the origin. The *hollow ball*

$$\mathbb{HB} = \bigcup_{r=r_0}^{1} S(r),$$

where $0 < r_0 < 1$ is the radius of the hollowed-out part, will serve to define the space in which the belt performs its trick.

The spheres $S(1), S(r_0)$ are called the outer sphere and the inner sphere, respectively. The outer sphere will represent the fixed end of the belt, and the inner sphere will represent the tag of the quaternion demonstrator. We can use intervals to represent a belt connecting the inner sphere to the outer sphere as

$$\mathscr{B} = ([-\rho, \rho], 0, [0, 1]) \cap \mathbb{HB}$$

where $\rho < r_0$ is half the width of the belt.

The belt-trick can now be illustrated as a global deformation $B_t : \mathbb{R}^3 \to \mathbb{R}^3$ parameterized by time $t \in [0, 1]$. At $t = 0$, $B_t(\mathscr{B})$ deforms the belt $\mathscr{B}$, giving it a 360 degree twist. As $t \to 1$, the belt will continuously deform into a belt with a -360 degree twist without rotating the inner or outer spheres—keeping the ends of the belt fixed.

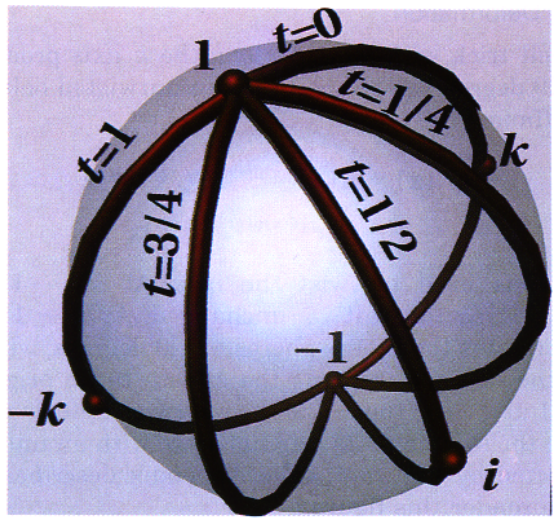


Fig. 4. Unit quaternion paths corresponding to various stages of the belt trick deformation. The sphere $\mathbb{S}^2$ is a slice of the hypersphere $\mathbb{S}^3$, consisting of unit quaternions whose **j** components is zero (similar to the figures at the end of Barr et al. [1992]).

The belt trick deformation shears the belt, rotating each increasingly larger spherical "shell" in $\mathbb{HB}$ by angles of increasingly larger value about an axis that changes over time. The rotation angle function is

$$\theta(\mathbf{x}) = 2\pi \frac{1 - \|\mathbf{x}\|}{1 - r_0} \tag{8}$$

whereas the rotation axis function is

$$\mathbf{u}(t) = e^{\pi t \mathbf{j}}\mathbf{k} = (\sin \pi t, 0, \cos \pi t). \tag{9}$$

The function R, from (7), specifies rotation about an arbitrary axis, and is used to define the belt trick deformation

$$B_t(\mathbf{x}) = R(e^{\frac{1}{2}\theta(\mathbf{x})\mathbf{u}(t)})\mathbf{x}. \tag{10}$$

Consider the unit quaternions used for rotations in the belt trick deformation. These are plotted in Figure 4. At $t = 0$, as $\|\mathbf{x}\|$ ranges from ρ_0 to 1, the unit quaternions form an arc from 1 through $\mathbf{k}$ to -1. As t: $0 \to 1$ this arc rotates about $\mathbb{S}^3$ from one side to the other. At $t = 1/2$ this arc extends from 1 through $\mathbf{i}$ to -1, and at $t = 1$ this arc extends from 1 through $-\mathbf{k}$ to -1. Since the arcs all begin at 1, the orientation of the inner sphere, and end at -1, the orientation of the outer sphere, the inner and outer spheres do not rotate during the belt trick.

3.2 The Unfurling Deformation

Composing the belt trick with a twist along the k axis produces a belt trick that takes the 720 degree twisted belt into an untwisted belt, and is given in Francis and Kauffman [1994] as

$$\begin{aligned} B_{s,t}(\mathbf{x}) &= R(e^{\frac{1}{2}\theta(\mathbf{x})\mathbf{u}(t)})R(e^{\frac{1}{2}s2\pi\mathbf{k}})\mathbf{x} \\ &= R(e^{\frac{1}{2}\theta(\mathbf{x})\mathbf{u}(t)}e^{\frac{1}{2}s2\pi\mathbf{k}})\mathbf{x} \end{aligned} \tag{11}$$

where $s \in [-1, 1]$ is used to twist the belt. At $s = -1$ and $t = 0$, the deformation $B_{s,t}$ leaves the belt $\mathscr{B}$ unchanged. As $s \to 1$ while $t = 0$ the deformation $B_{s,t}$ rotates the inner spheres -720 degrees, which puts a -720 degree twist in the belt and returns the sphere to its original orientation. Then, while $s = 1$ as $t \to 1$, the -720 degree twist unfurls around the inner sphere, returning the system to its original state with an untwisted belt. This process is illustrated by Figure 5. (The Appendix describes the ray-tracing technique used to render this figure.)

The unit quaternions used for rotations in the unfurling deformation are plotted in Figure 6. At $s = 1$, $t = 0$, as $\|\mathbf{x}\|$ ranges from ρ_0 to 1, the unit quaternions form a circle from 1 through $\mathbf{k}$ through -1 through $-\mathbf{k}$ and back to 1. As $t : 0 \to 1$ this circle contracts around $\mathbb{S}^3$, always intersecting 1. At $t = 1/2$ the circle extends from 1 through $\mathbf{i}$ and back to 1, and at $t = 1$ circle degenerates to the point 1. Since the circles all begin and end at 1, as with the belt trick, the inner and outer spheres do not rotate during the unfurling.

4. SIMULATING THE QUATERNION DEMONSTRATOR

The belt trick and unfurling deformations are global deformations. They maintain the belt's volume (for the same reason the twist deformation [Barr 1984] preserves volume) but stretch the length of the belt like a rubber band such that the inner and outer spheres remain centered about the origin. The belt in the quaternion demonstrator maintains a constant length, but the tag is free to move about. This suggests that a local deformation should be used to simulate the belt in the quaternion demonstrator. Furthermore, this local deformation should minimize the twisting of the belt.

First, the orientation of the tag is represented by a quaternion q. Then a geodesic (a great arc on $\mathbb{S}^3$) of unit quaternions interpolates the orientations along the belt from the orientation of the fixed end of the belt, 1, to the orientation of the tag, q. Finally, changes to this geodesic resulting from rotations of the tag must be carefully monitored to prevent drastic changes in the shape of the belt.

4.1 Simulating the Tag

In the physical quaternion demonstrator, the tag is a small rectangle attached to the free end of the belt. In the simulation, we consider the tag to be the edge of the free end of the belt.

Let $\mathbf{v}_F$ represent the orientation of the edge of the belt's fixed end as a vector from one corner to the other. Let $\mathbf{v}_T$ denote the orientation of the tag,

Fig. 5. The unfurling, using a global deformation to show how to remove a 720 degree twist from a ribbon without moving either end.

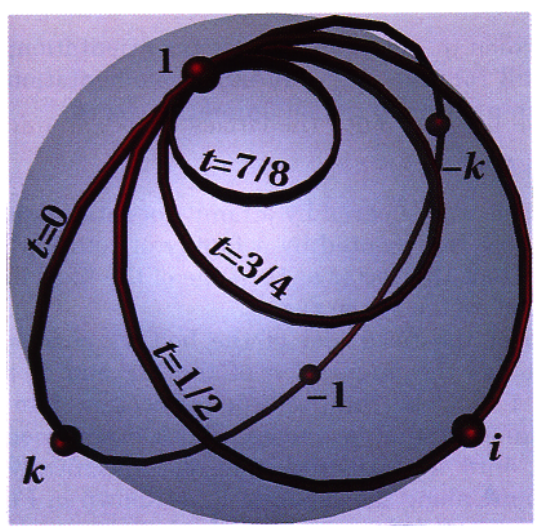


Fig. 6. Unit quaternion paths corresponding to various stages of the unfurling deformation

as the vector connecting the corresponding corners at the edge of the belt's free end. Let $q \in \mathbb{S}^3$ be a unit quaternion denoting the state of the quaternion demonstrator. Then the orientation of the tag with respect to the orientation of the fixed end of the belt is given by

$$\mathbf{v}_T = R(q)\mathbf{v}_F.$$

Multiplication of q by an imaginary rotates the tag 180 degrees discretely. In the simulation, these rotations are performed incrementally and appear

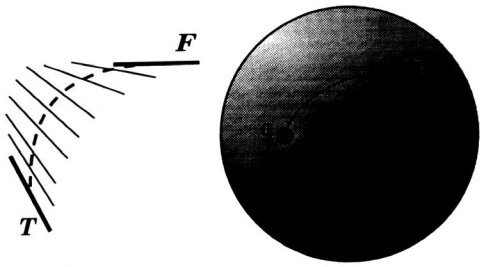


Fig. 7. The belt as the union of rotated line segments (left). The orientation of $\mathbf{v}_F$, the fixed edge of the belt, corresponds to one whereas the orientation of $\mathbf{v}_T$, the tag edge of the belt, corresponds to q. The geodesic on $\mathbb{S}^3$ (right) connects 1 to q, and specifies the orientations of the segments interpolating $\mathbf{v}_F$ to $\mathbf{v}_T$.

continuous. With the physical quaternion demonstrator, multiplication by $\mathbf{i}$ is performed by flipping the tag. In the simulation, multiplication by $\mathbf{i}$ is accomplished by pressing and holding the "i" key and watching the tag slowly flip.

Let q_0 be a unit quaternion denoting the current orientation of the tag. Then incremental multiplication of the tag is simulated by

$$q_1 = q_0 e^{\epsilon \mathbf{u}}, \tag{12}$$

where unit quaternion q_1 specifies the new tag orientation; $\mathbf{u}$ is one of $\mathbf{i}$, $\mathbf{j}$, or $\mathbf{k}$, and ϵ is a small rotation angle. In our implementation, setting $\epsilon = 0.02$ radians resulted in a pleasing ribbon animation speed.

4.2 Simulating the Belt

In Section 3, the belt was sheared by a family of rotating concentric spheres. Here, the belt is best represented by a family of rotating line segments. As before, let $\mathbf{v}_F$ represent the vector at the edge of the fixed end of the belt, and let $\mathbf{v}_T$ represent the vector at the tag.

The orientation of the fixed end of the belt $\mathbf{v}_F$ corresponds to the unit quaternion 1 whereas the orientation of the tag end of the belt $\mathbf{v}_T$ corresponds to q. Let $\Gamma \subset \mathbb{S}^3$ be the geodesic connecting 1 to q. Then the belt consists of the union of line segments whose orientations interpolate the orientation of $\mathbf{v}_F$ into the orientation of $\mathbf{v}_T$, specifically the orientations represented by points along the geodesic Γ, as shown in Figure 7.

We describe the local deformation of the belt by applying the rotations represented by the quaternions along the geodesic Γ to the tangent space of the belt. The deformed belt is then constructed as an initial-value problem by integrating the belt over these deformed tangent vectors.

First, decompose q into exponential form as

$$\begin{aligned} q &= r + \mathbf{v}, \\ \theta &= 2\cos^{-1} r, \\ \mathbf{u} &= \frac{\mathbf{v}}{\|\mathbf{v}\|} = \frac{\mathbf{v}}{\sin \frac{1}{2}\theta}, \\ q &= e^{\frac{1}{2}\theta \mathbf{u}}, \end{aligned}$$

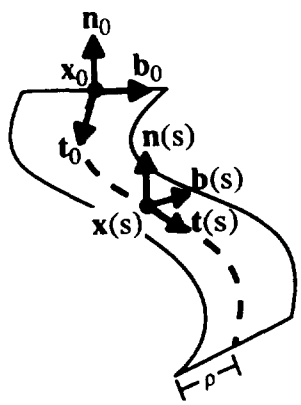


Fig. 8. The Frenet frame along the spine of the belt.

where $\cos^{-1}$ always returns values in the range $(-\pi, \pi]$. This form reveals the rotation q represents. The geodesic Γ extending from 1 to q on $\mathbb{S}^3$ is parameterized by the function $\gamma(s) \in \Gamma$ for $s \in [0, 1]$ as

$$\gamma(s) = e^{\frac{1}{2}s\theta\mathbf{u}}. \tag{13}$$

The unit quaternion function $\gamma(s)$ specifies the orientations the belt twists through on its path from its fixed end to the tag. (Quaternion exponentiation maps the line segment in $\mathbb{R}^3$ connecting the origin to $(1/2)\theta\mathbf{u}$ to the geodesic in $\mathbb{S}^3$ connecting 1 to q.)

Let $\mathbf{x}_0 = \mathbf{0}$ be the position, and $\mathbf{t}_0 = \mathbf{k}$, $\mathbf{b}_0 = \mathbf{i}$, and $\mathbf{n}_0 = \mathbf{j}$ be the Frenet frame (tangent $\mathbf{t}$, binormal $\mathbf{b}$, and normal $\mathbf{n}$) of the center of the fixed end of the belt, as in Figure 8.

The local description of points along the spine of the belt is given by the Frenet frame

$$\begin{aligned}\mathbf{t}(s) &= R(\gamma(s))\mathbf{t}_0,\\ \mathbf{b}(s) &= R(\gamma(s))\mathbf{b}_0,\\ \mathbf{n}(s) &= R(\gamma(s))\mathbf{n}_0.\end{aligned}$$

Integrating the tangent $\mathbf{t}(s)$ produces points along the spine of the belt

$$\mathbf{x}(s) = \mathbf{x}_0 + \int_0^s \mathbf{t}(\sigma)\, d\sigma. \tag{14}$$

The belt is formed as a ruled surface consisting of line segments connecting the vertices

$$\mathbf{x}(s) \pm \rho\mathbf{b}(s),$$

where, as before, ρ is one-half the width of the belt. The twisting of the belt visualizes the interpolation of orientations of a line segment from its fixed end to the tag end.

Since the Frenet frame is just rotated by these functions, the length of the belt remains unchanged under the deformation. (By the way, if the locally specified deformation were not simply a rotation, then the normal vector

transformation rule [Barr 1984] would apply. In such a case, the transformed binormal would be found via the inverse transpose of the Jacobian matrix of $R(\ )$, and the normal would be constructed from the cross product of the tangent vector with the binormal vector.)

4.3 Limiting Belt Velocity

The orientation interpolation geodesic on $\mathbb{S}^3$ is an arc extending from 1 (the north pole) to q. As q passes by -1 (the south pole) the geodesic generated by (13) will move from one side of $\mathbb{S}^3$ to the other (since the range of $\cos^{-1}$ is $(-\pi, \pi])$.

This movement keeps the belt from accumulating unnecessary twists. Since the geodesic connecting 1 to q is the shortest path on $\mathbb{S}^3$, belt tricks occur naturally as the tag is rotated to avoid twists of greater than 2π in the belt. This movement has one disadvantage in that certain small movements of q near the south pole cause the geodesic to swing around quickly to the other side of $\mathbb{S}^3$ resulting in a belt trick that is too fast for the user to follow. In fact, if the tag rotates directly through -1, the geodesic snaps from one side of $\mathbb{S}^3$ to the other, causing an instantaneous belt trick (an instant reverse of the twist in the belt).

For example, turning the tag about the **k** axis from the initial 1 state produces eventually a 2π twist in the ribbon about the **k** axis. Turning the tag slightly causes an instantaneous belt trick, snapping the belt from a 2π twist to a -2π twist. Although both configurations are nearly equivalent, representing nearby quaternion values, their appearance to the user is quite different.

There are two remedies for handling instantaneous belt tricks. The first remedy senses when the tag orientation path crosses -1, or nearly misses it. When it does, this remedy assumes control of the demonstrator from the user and performs an explicit animated belt trick to remove the excess twist in the belt.

The second remedy capitalizes on numerical error to perform belt tricks automatically, as necessary. As (12) rotates the tag incrementally, small numerical errors will accumulate in the quaternion representation of the tag's orientation. In other words, turning the tag about the **k** axis will introduce slight rotations about the **i** and **j** axes as well. By the time the tag's turning has twisted the belt by 2π and more, these perturbations will cause the tag's orientation quaternion q to miss the south pole. The resulting geodesics will quickly swing across $\mathbb{S}^3$ producing a belt trick, although possibly at a very fast rate.

We chose to implement the second remedy in the simulation of the quaternion demonstrator for its elegance and because it never assumes control of the demonstrator away from the user. This elegance may give the impression that the second remeby avoids belt tricks, which are an essential point of the quaternion demonstrator. To the contrary, belt tricks resulting from near misses of the south pole are indiscernable from the belt trick required to simulate the belt continuously through a direct hit. The dependence of this remedy on numerical noise affects its robustness in that a direct hit on the

south pole would lock up the quaternion demonstrator simulation. Such a direct hit is highly unlikely and has never occured in our experience. The only remaining task is to regulate the rate at which the quaternion demonstrator performs automatic belt tricks.

The speed of the automatic belt trick is regulated by controlling the speed at which the geodesic Γ flips around $\mathbb{S}^3$, which, in turn, is controlled from the rate of rotation of the tag by regulating the ϵ in (12). The rest of this section is devoted to deriving the amount of regulation of ϵ necessary to control the speed of the automatic belt tricks when they occur.

Following Misner et al. [1973], we can describe a unit quaternion $r + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ in spherical coordinates with three angles α, ϕ, and θ as

$$\begin{aligned} x &= \sin\alpha \sin\phi \cos\theta, \\ y &= \sin\alpha \sin\phi \sin\theta, \\ z &= \sin\alpha \cos\phi, \\ r &= \cos\alpha. \end{aligned}$$

The inverse is computed as

$$\begin{aligned} \alpha &= \cos^{-1} r, \\ \phi &= \cos^{-1} \frac{z}{\sin\alpha}, \\ \theta &= \cos^{-1} \frac{x}{\sin\alpha \sin\theta} \\ &= \sin^{-1} \frac{y}{\sin\alpha \sin\theta}. \end{aligned}$$

Also from Misner et al. [1973], the differential of geodesic length is given by

$$ds^2 = d\alpha^2 + \sin^2(\alpha)(d\phi^2 + \sin^2(\phi)\, d\theta^2). \tag{15}$$

Geodesics extending from the north pole to q have fixed ϕ, θ, and an α that ranges from 0 at the north pole to a positive value at q.

Specify the original tag orientation q_0 in polar form as $(\alpha_0, \phi_0, \theta_0)$ and likewise with the new tag orientation q_1. The geodesic Γ_0 connecting 1 to q_0 is of the polar form $([0, \alpha_0], \phi_0, \theta_0)$, and the new geodesic Γ_1 is $([0, \alpha_1], \phi_1, \theta_1)$. We are only concerned with geodesics that extend from the north pole to near the south pole, at least where $\alpha_0, \alpha_1 > \pi/2$. We also assume, without loss of generality, that $\phi_0 \le \phi_1$ and $\theta_0 \le \theta_1$.

Let corresponding points on Γ_0 and Γ_1 be points of equal α. By observation, the maximum distance between corresponding points on Γ_0 and Γ_1 occurs at the equator, where $\alpha = 1/2$. Let the distance geodesic Γ_d denote the geodesic between corresponding equatorial points of Γ_0, Γ_1. By regulating the length of Γ_d, we regulate the rate of the belt trick.

Let $\Delta\phi = \phi_1 - \phi_0$ and $\Delta\theta = \theta_1 - \theta_0$. From (15) we can approximate the length of Γ_d with

$$s^2 \approx \Delta\phi^2 + \sin^2(\phi_0)\, \Delta\theta^2 \tag{16}$$

since the change in α on Γ_d is zero and $\sin^2(\pi/2) = 1$.

Let s_{max} be the maximum allowed length of Γ_d—the maximum rate of change between Γ_0 and Γ_1. If the length of Γ_d exceeds the maximum allowed length s_{max} then we must reduce the increment of q. Let

$$\lambda = s_{max}/s \tag{17}$$

be the amount Γ_d needs to be scaled back to meet the maximum allowed length. Then

$$\begin{aligned} s_{max}^2 &= \lambda^2 s^2 \\ &= \lambda^2(\Delta\phi^2 + \sin^2(\phi_0)\Delta\theta^2), \\ &= (\lambda\Delta\phi)^2 + \sin^2(\phi_0)(\lambda\Delta\theta)^2. \end{aligned}$$

Changing the polar values ϕ and θ of q_0 by no more than $\lambda\Delta\phi$ and $\lambda\Delta\theta$ prevents the geodesic from rotating too fast around $\mathbb{S}^3$—prevents the belt from moving too quickly. Hence, the incremental rotation

$$q_1 = q_0 e^{\lambda\epsilon\mathbf{u}}, \tag{18}$$

where λ is given in (17) as the quotient of the parameter s_{max} and the "distance" between successive geodesics s, produces a new tag orientation sufficiently close to the original to limit belt movement properly when necessary. In practice, setting $s_{max} = 0.1$ disciplines the belt into reasonable behavior.

5. CONCLUSION

Using the methods of Section 4, we have constructed a simulated quaternion demonstrator, as described in Section 5.1. The simulated quaternion demonstrator not only demonstrates unit quaternion multiplication, like its physical counterpart, but also illustrates the quaternion interpolation of orientation from one end of the belt to the other. The initial success of this prototype has inspired several ideas for further research in this direction, which are described in Section 5.2.

5.1 Results

Our implementation of the quaternion demonstrator, titled "quatdemo," was developed on an SGI Indigo Elan, and can be obtained via anonymous ftp to the Imaging Research Laboratory at irl.eecs.wsu.edu from the directory /pub/IRL/quatdemo.

It consists of the coordinate axis and a belt. The imaginaries are labeled at the tag end of the belt, in their corresponding orientations. Pressing and holding the "i," "j," or "k" key rotates the tag end of the belt, causing minimal twists in the belt. The current unit quaternion value is represented visually by the configuration of the demonstrator and is verified by a formatted text version of the current unit quaternion value.

Our implementation simulates the spine of the belt discretely with 256 samples, using the Euler method to approximate the integral (14). Euler integration is highly susceptible to accumulated error, but errors in the

position of the tag end of the quaternion demonstrator are of little consequence. The only state where the position of the tag is noticeable is -1, where the tag end of the belt should be coincident with the fixed end of the belt. We are again fortunate in that the numerical noise accumulated in the incremental rotations of the tag, statistically, prevents this state from being represented exactly.

Figure 2 displays the various states of the quaternion demonstrator simulated by the methods discussed in Section 4. As expected, belt tricks occur automatically when necessary to remove excess twists in the belt; the belt never contains more than a full 360 degree twist in any direction. Figure 3 displays an automatically occurring belt trick, which results at the midway point when holding the "k" key down.

After an automatic belt trick, enough error accumulates to cause the spine of the belt to return to a position slightly offset from its original state. In our implementation, pressing the space bar resets the quaternion demonstrator to the 1 state.

5.2 Further Research

The concept of illustrating the track of a rotation through the use of attached belts to objects is basic to the quaternion demonstrator. In this article we have considered the resulting symmetries of a rectangle in three-dimensional space. The same results apply to the symmetry of any object in three-space $\mathbb{R}^3$.

Formally, Let $\mathscr{O} \subset \mathbb{R}^3$ be a subset of $\mathbb{R}^3$ containing the origin. Let $\mathbb{SO}(3)$ denote the set of rotations about the origin of $\mathbb{R}^3$. Let $\mathrm{Symm}(\mathscr{O})$ denote the subgroup of $\mathbb{SO}(3)$ consisting in those rotations $g \in \mathbb{SO}(3)$ for which $g(\mathscr{O}) = \mathscr{O}$ setwise. Now the three-sphere $\mathbb{S}^3$ of unit quaternions covers the $\mathbb{SO}(3)$ doubly via the map

$$\pi : \mathbb{S}^3 \to \mathbb{SO}(3) : \mathbf{v} \to R(q)\mathbf{v} = q\mathbf{v}q^{-1}, \tag{19}$$

where $\mathbf{v}$ is a pure quaternion—hence $\mathbf{v}$ is a vector in $\mathbb{R}^3$. This is an abstract description of our representation of rotations by quaternions.

Now, the set of unit quaternions, $\pi^{-1}(\mathrm{Symm}(\mathscr{O}))$, covering the symmetry group of the object, is a subgroup of $\mathbb{S}^3$, called the *binary group* of $\mathrm{Symm}(\mathscr{O})$. If the object $\mathscr{O}$ is the rectangular tag $\mathscr{T}$, this group is the eight-element quaternion group,

$$\{\pm 1, \pm \mathbf{i}, \pm \mathbf{j}, \pm \mathbf{k}\}. \tag{20}$$

This is the abstract description of what the quaternionic demonstrator has demonstrated.

An extension of the demonstrator (by attaching a belt to another object, $\mathscr{O}$) can illustrate the binary group for the symmetries of any object. These groups, in the case of regular solids such as the tetrahedron, octahedron, or icosahedron, are of great interest both practically and mathematically. This extension of our demonstrator is one of the immediate prospects for further research.

It is also possible to extend our methods to study the structure of rotations of four-space and to the study and illustration of properties of *octonians* [Cayley 1897], which are an eight-dimensional generalization of the quaternions.

APPENDIX

RENDERING

The most straightforward method for rendering a deformed object is to polygonize its surface, apply the deformation to the polygon vertices, then render the resulting polygons. Special-purpose hardware can render polygonal objects in real time, permitting interactive modeling. For example, the local deformation of the belt used in the quaternion demonstrator was rendered in this fashion. Polygonization can be problematic when investigating deformations in that the result of deforming vertices produces a polygon that may be a poor fit when compared to the deformation of the entire polygon, requiring some form of detection and dynamic subdivision.

Alternatively, we can render the deformed object directly as an implicit surface, preserving, at least to pixel precision, the detail of the deformed geometry. Let $f(\mathbf{x})$ be a function implicitly defining the set $A \subset \mathbb{R}^3$ such that

$$f(\mathbf{x}) < 0 \Leftrightarrow \mathbf{x} \in \mathring{A}, \tag{21}$$

$$f(\mathbf{x}) = 0 \Leftrightarrow \mathbf{x} \in \partial A, \tag{22}$$

$$f(\mathbf{x}) > 0 \Leftrightarrow \mathbf{x} \in \mathbb{R}^3 \setminus A, \tag{23}$$

and let D: $\mathbb{R}^3 \to \mathbb{R}^3$ denote the deformation function. Then the deformed set $D(A)$ is implicitly defined by the function $f \circ D^{-1}(\mathbf{x})$.

With few exceptions, ray tracing is the means for direct visualization of implicit surfaces. Some recent ray intersection methods require the Lipschitz constant of the function [Kalra and Barr 1989; Hart 1993]. The Lipschitz constant of a function f: $A \to B$ from metric space (A, d_A) to metric space (B, d_B) is the smallest positive value λ such that

$$d_B(f(x), f(y)) \leq \lambda d_A(x, y) \tag{24}$$

for all $x, y \in A$. The Lipschitz constant bounds the amount a transformation can expand an object. If f: $\mathbb{R} \to \mathbb{R}$, then the Lipschitz constant of f indicates the steepest slope in the graph of f. One can use the Lipschitz-based ray-tracing method to investigate the "unfurling," the global deformation described in Section 3.2.

By observation, the largest dilation caused by $B_{s,t}$ occurs when $s = 1$, $t = 0$. This deformation winds the segment $[r, 1]\mathbf{i}$ twice around the origin. Reducing to $\mathbb{R}^2$, the parametric equation of this curve takes the x axis from $[r, 1]$ into the twice-winding curve as

$$f_x(x) = x \cos 2\theta(x), \tag{25}$$

$$f_y(x) = x \sin 2\theta(x) \tag{26}$$

with derivatives

$$f_x'(x) = \cos 2\theta(x) - 2\theta'(x)\sin 2\theta(x), \tag{27}$$

$$f_y'(x) = \sin 2\theta(x) + 2\theta'(x)\cos 2\theta(x). \tag{28}$$

$$\theta'(x) = -\frac{2\pi}{1 - r_0} \tag{29}$$

Where (29) corresponds to (8). The arc length of this double twist is found using

$$ds^2 = f_x'(x)^2 + f_y'(x)^2, \tag{30}$$

$$= 1 + \frac{16\pi^2}{(1 - r_0)^2}. \tag{31}$$

Since ds^2 reaches its maximum (over the proper domain) when $x = 1$, we have the Lipschitz constant

$$\text{Lip } B_{1,0} = \sqrt{1 + \frac{16\pi^2}{(1 - r_0)^2}}. \tag{32}$$

The deformation $B_{1,0}$ dilates more than any other $B_{s,t}$ for all $s \in [-1, 1]$ and $t \in [0, 1]$. Equation (32) is an upper bound of the Lipschitz constant for $B_{s,t}$, and under a similar argument, for the belt trick deformation B_t. Hence, (32) is a suitable (though not necessarily optimal for all parameters s and t) Lipschitz bound for ray-tracing the results of the belt trick and unfurling deformations.

This Lipschitz constant was used to render the unfurling demonstration in Sandin et al. [1993], excerpted in Figure 5.

ACKNOWLEDGMENTS

We wish to thank the editor and reviewers for very extensive comments and suggestions on this topic.

REFERENCES

BARR, A. H., CURRIN, B., GABRIEL, S., AND HUGHES, J. F. 1992. Smooth interpolation of orientations with angular velocity constraints using quaternions. *Comput. Graph. 26*, 2 (July), 313–320.

BARR, A. H. 1984. Global and local deformations of solid primitives. *Comput. Graph. 18*, 3 (July), 21–30.

BOLKER, E. 1973. The spinor spanner. *Am. Math. Month. 80*, 9 (Nov.), 977–984.

CAYLEY, A. 1897. *The Collected Mathematical Papers of Arthur Cayley*. Cambridge University Press, Cambridge, U.K.

FRANCIS, G. K. 1987. *A Topological Picturebook*. Springer-Verlag, New York.

FRANCIS, G. K. AND KAUFFMAN, L. H. 1994. Air on the Dirac strings. In *Mathematical Legacy of Wilhelm Magnus*. AMS, Providence, R.I.

HAMILTON, W. R. 1866. *Elements of Quaternions*. Longmans Green, London.

HANSON, A. 1992. The rolling ball: Applications of a method for controlling three degrees of freedom using two-dimensional input devices. In *Graphics Gems*, D. Kirk, Ed. Vol. 3. Academic Press, San Diego, Calif., 51–60.

HART, J. C. 1993. Sphere tracing: Simple robust antialiased rendering of distance-based implicit surfaces. Tech. Rep. EECS-93-15, School of EECS, Washington State Univ., Pullman, Wash. Appears in *SIGGRAPH '93*, Course Notes #25 "Modeling, Visualizing and Animating Implicit Surfaces."

KALRA, D. AND BARR, A. H. 1989. Guaranteed ray intersections with implicit surfaces. *Comput. Graph. 23*, 3 (July), 297–306.

KAUFFMAN, L. H. 1991. *Knots and Physics*. World Scientific, Teaneck, N.J.

KAUFFMAN, L. H. 1987. *On Knots*. Princeton University Press, Princeton, N.J.

MISNER, C. W., THORNE, K. S., AND WHEELER, J. A. 1973. *Gravitation*. Freeman, San Francisco, Calif.

SANDIN, D. J., KAUFFMAN, L. H., AND FRANCIS, G. K. 1993. Air on the Dirac strings. *SIGGRAPH Video Rev. 93*. Animation.

SHOEMAKE, K. 1992. ARCBALL: A user interface for specifying three-dimensional orientation using a mouse. In *Proceedings of Graphics Interface '92* (May) 151–156.

SHOEMAKE, K. 1985. Animating rotation with quaternion curves. *Comput. Graph. 19*, 3 (July), 245–254.

SPIVAK, M. 1965. *Calculus on Manifolds*. W. A. Benjamin, New York.

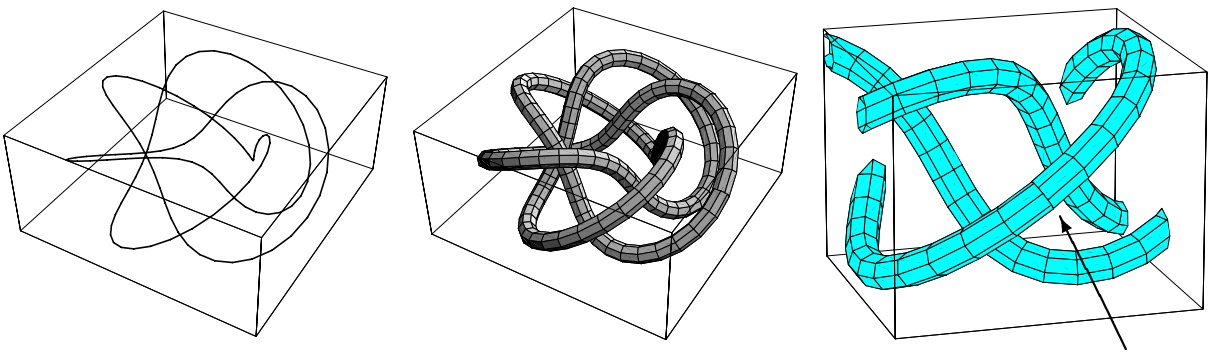
Received October 1993; revised July 1994; accepted July 1994

Editor: John Hughes

Constrained Optimal Framings of Curves and Surfaces using Quaternion Gauss Maps

Andrew J. Hanson *

Computer Science Department
Indiana University
Bloomington, IN 47405 USA



(a) (b) (c)

Figure 1: The (3,5) torus knot, a complex periodic 3D curve. (a) The line drawing is nearly useless as a 3D representation. (b) A tubing based on parallel transporting an initial reference frame produces an informative visualization, but is not periodic. (c) The arrow in this closeup exposes the subtle but crucial non-periodic mismatch between the starting and ending parallel-transport frames; this would invalidate any attempt to *texture* the tube. The methods of this paper provide robust parameterization-invariant principles for resolving such problems.

Abstract

We propose a general paradigm for computing optimal coordinate frame fields that may be exploited to visualize curves and surfaces. Parallel-transport framings, which work well for open curves, generally fail to have desirable properties for cyclic curves and for surfaces. We suggest that minimal quaternion measure provides an appropriate heuristic generalization of parallel transport. Our approach differs from minimal-tangential-acceleration approaches due to the addition of "sliding ring" constraints that fix one frame axis, but allow an axial rotational freedom whose value is varied in the optimization process. Our fundamental tool is the quaternion Gauss map, a generalization to quaternion space of the tangent map for curves and of the Gauss map for surfaces. The quaternion Gauss map takes 3D coordinate frame fields for curves and surfaces into corresponding curves and surfaces constrained to the space of possible orientations in quaternion space. Standard optimization tools provide application-specific means of choosing optimal, e.g., length- or area-minimizing, quaternion frame fields in this constrained space.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques. I.3.8 [Computer Graphics]: Applications.

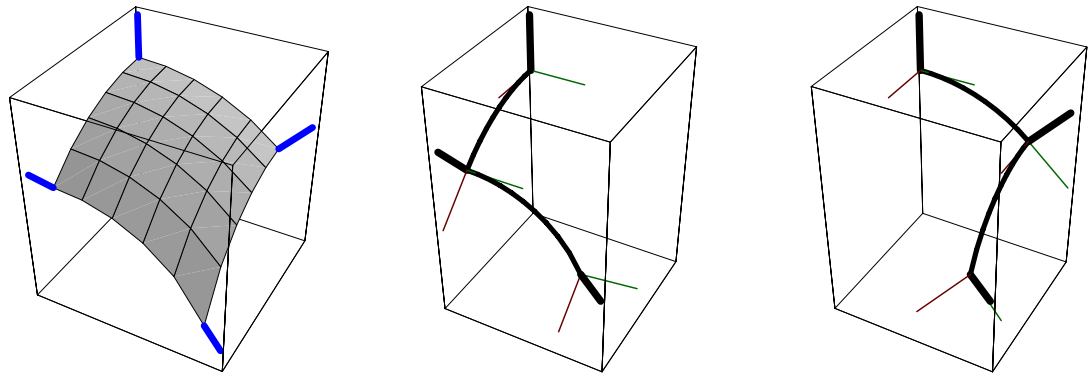
Keywords: Quaternions; Frames; Tubing; Curves; Surfaces

*Email: hanson@cs.indiana.edu

1 Introduction

We propose a general framework for selecting optimal systems of coordinate frames that can be applied to visualizing geometric structures such as curves and surfaces in three-dimensional space. The methods contain "minimal-turning" parallel-transport framings of curves as a special case, are independent of parameterization, and extend naturally to situations where parallel transport is not applicable.

Motivation. Many visualization problems require techniques for effectively displaying the properties of curves and surfaces. The problem of finding appropriate representations can be quite challenging. Representations of space curves based on single lines are often inadequate for graphics purposes; significantly better images result from choosing a "tubing" to display the curve as a graphics object with spatial extent. Vanishing curvature invalidates methods such as the Frenet frame, and alternative approaches to tubing involve heuristics unrelated to parameterization-invariant optimization measures in order to achieve such properties as periodicity. Similar problems occur in the construction of suitable visualizations of complex surfaces and oriented particle systems on surfaces, since the intrinsic orientation properties may be poorly exposed by the original representation. If a surface patch is represented by a rectangular but nonorthogonal mesh, for example, there is no obvious way to choose among alternative local orthonormal frame assignments; if the surface has regions of vanishing curvature, methods based on directions of principal curvatures break down as well.



(a) (b) (c)

Figure 2: (a) A smooth 3D surface patch having a non-orthogonal parameterization, along with its geometrically-fixed normals at the four corners. No unique orthonormal frame is derivable from the parameterization. If we imitate parallel transport for curves to evolve the initial frame at the top corner to choose the frame at the bottom corner, we find that paths (b) and (c) result in incompatible final frames at the bottom corner. This paper addresses the problem of systematically choosing a compatible set of surface frames in situations like this.

While we emphasize curves and surfaces in this paper to provide intuitive examples, there are several parallel problem domains that can be addressed with identical techniques. Among these are extrusion methods and generalized cones in geometric modeling, the imposition of constraints on a camera-frame axis in key-frame animation, and the selection of a 2D array of camera-frame axis choices as a condition on a constrained-navigation environment (see, e.g., Hanson and Wernert [13]).

Figure 1 summarizes the basic class of problems involving curves that will concern us here. The line drawing (a) of a (3,5) torus knot provides no useful information about the 3D structure. Improving the visualization by creating a tubing involves a subtle dilemma that we attempt to expose in the rest of the figure. We cannot use a periodic Frenet frame as a basis for this tubing because inflection points or near-inflection points occur for many nice-looking torus knot parameterizations, and in such cases the Frenet frame is undefined or twists wildly. The parallel-transport tubing shown in (b) is well-behaved but not periodic; by looking carefully at the magnified portion next to the arrow in Figure 1(c), one can see a gross mismatch in the tessellation due to the nonperiodicity. While it would be possible in many applications to ignore this mismatch, it has been the subject of a wide variety of previous papers (see, e.g., [16, 24, 5]), and must obviously be repaired for many other applications such as those requiring textured periodic tubes.

Figure 2 illustrates a corresponding problem for surface patches. While the normals to the four corners of the patch are always well-defined (a), one finds two different frames for the bottom corner depending upon whether one parallel transports the initial frame around the left-hand path (b) or the right-hand path (c). There is no immediately obvious right way to choose a family of frames covering this surface patch.

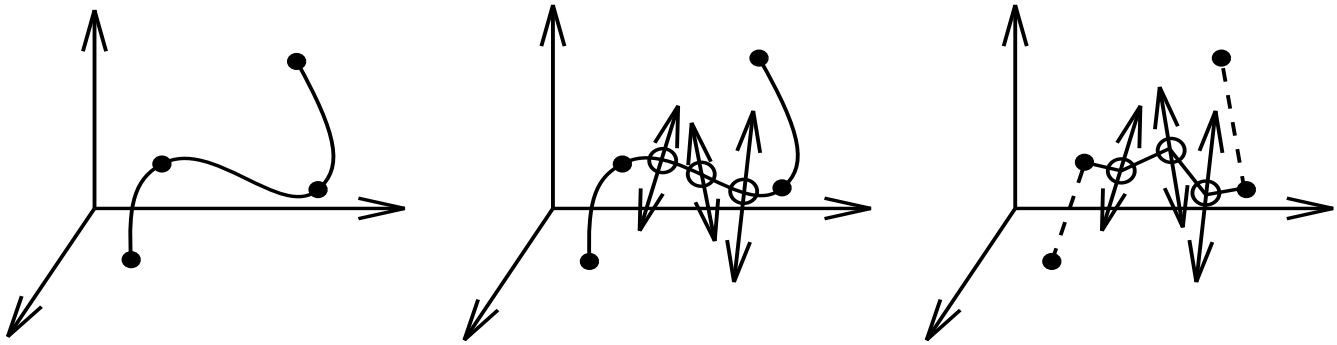
Our goal is to propose a systematic family of optimization methods for resolving problems such as these.

Methodology. We focus on unit quaternion representations of coordinate frames because of the well-known natural structure of unit quaternions as points on the three-sphere S^3, which admits a natural distance measure for defining optimization problems, and supports in addition a variety of regular frame-interpolation methods (see, e.g., [25, 23, 19, 15]). We do not address the related question of optimal freely moving frames treated by the minimal-tangential-acceleration methods (see, e.g., [2, 22, 8]); we are instead concerned with closely-spaced points on curves and surfaces where one direction of the frame is already fixed, and the chosen functional minimization in quaternion space must obey the additional constraint imposed by the fixed family of directions. Additional references of interest, especially regarding the treatment of surfaces, include [14, 20]. Figure 3 provides a visualization of the difference between the general interpolation problem and our constrained problem: a typical spline minimizes the bending energy specified by the chosen anchor points; requiring intermediate points to slide on constrained paths during the minimization modifies the problem. In particular, 3D spline curves need not intersect any of the constraint paths. In addition, we note that we typically have already sampled our curves and surfaces as finely as we need, so that piecewise linear splines are generally sufficient for the applications we discuss.

Our solution to the problem is to transform the intrinsic geometric quantities such as the tangent field of a curve and the normal field of a surface to quaternion space and to construct the quaternion manifold corresponding to the one remaining degree of rotational freedom in the choice of coordinate frame at each point. Paths in this *space of possible frames* correspond to specific choices of the *quaternion Gauss map*, a subspace of the space of possible quaternion frames of the object to be visualized. Mathematically speaking, the space of possible frames is the circular *Hopf fiber* lying above the point in S^2 corresponding to each specific curve tangent or surface normal (see, e.g., [26, 3]).

Parallel Transport and Minimal Measure. Constraining each quaternion point (a frame) to its own circular quaternion path (the axial degree of rotational freedom), we then minimize the quaternion length of the frame assignment for curves and the quaternion area of the frame assignment for surfaces to achieve an optimal frame choice; this choice reduces to the parallel-transport frame for simple cases. Our justification for choosing minimal quaternion length for curves is that there is a unique rotation in the plane of two neighboring tangents that takes each tangent direction to its next neighbor along a curve: this is the geodesic arc connecting the two frames in quaternion space, and is therefore the minimum distance between the quaternion points representing the two frames. The choice of minimal area for surface frames is more heuristic, basically a plausibility argument that the generalization of minimal length is minimal area; no doubt this could be made more rigorous.

By imposing other criteria such as endpoint derivative values and



(a) (b) (c)

Figure 3: (a) The camera frame interpolation problem is analogous to the problem of finding a minimal-bending spline curve through a series of fixed key points. (b) The optimal curve frame assignment problem is analogous to fixing the end points of a curve segment and choosing *in addition* a family of lines along which the intermediate points are constrained to slide during the optimization process; in 3D, the spline path need not pass through the constraint lines. (c) In this paper, our sample points are generally close enough together that we apply the constraints to piecewise linear splines analogous to those shown here.

minimal bending energy (see Barr et al. [2, 22]), the short straight line segments and polygons that result from the simplest minimization could be smoothed to become generalized splines passing through the required constraint rings; since, in practice, our curve and surface samplings are arbitrarily dense, this was not pursued in the current investigation.

For space curves, specifying a frame assignment as a quaternion path leads at once to tubular surfaces that provide a "thickened" representation of the curve that interacts well with texturing, lighting, and rendering models. For surface patches, the approach results in a structure equivalent to that of an anisotropic oriented particle system (also a species of texture) whose pairs of tangent vector fields in the surface produce natural flow fields that characterize the local surface properties and are easy to display.

Background. General questions involving the specification of curve framings have been investigated in many contexts; for a representative selection of approaches, see, e.g., [16, 24, 5]. The quaternion Gauss map is a logical extension of the quaternion frame approach to visualizing space curves introduced by Hanson and Ma [11, 12]. For basic information on orientation spaces and their relationship to quaternions, see, e.g., [1, 21, 19].

Background on the differential geometry of curves and surfaces may be found in sources such as the classical treatise of Eisenhart [7] and in Gray's MATHEMATICA-based text [9], which inspired a number of the illustrations in this paper. The classical Frenet frame is defined and studied in these texts. The frame we refer to as the parallel-transport frame was first described carefully by Bishop [4], and has been commonly used in graphics (see, e.g., [5, 24, 17]). A significant difference between these two methods is that the Frenet frame is locally defined but possibly discontinuous, whereas the parallel-transport frame is continuous but non-local, corresponding to the solution of a differential equation.

2 The Space of Frames

We begin by introducing the key concept of the *space of possible frames*.

Suppose at each sample point $\mathbf{x}(t)$ of a curve, we are given a unit tangent vector, $\hat{\mathbf{T}}(t)$, computed by whatever method one likes (two-point sampling, five-point sampling, analytic, etc.). Then one can immediately write down a one-parameter family describing all possible choices of the normal plane orientation: it is just the set of rotation matrices $R(\theta, \hat{\mathbf{T}}(t))$ (or quaternions $q(\theta, \hat{\mathbf{T}}(t))$) that leave $\hat{\mathbf{T}}(t)$ *fixed*.

For surfaces, the analogous construction follows from determining the unit normal $\hat{\mathbf{N}}(u, v)$ at each point $\mathbf{x}(u, v)$ on the surface patch. The needed family of rotations $R(\theta, \hat{\mathbf{N}}(u, v))$ (or quaternions $q(\theta, \hat{\mathbf{N}}(u, v))$) now leaves $\hat{\mathbf{N}}(u, v)$ fixed and parameterizes the space of possible *tangent* directions completing a frame definition at each point $\mathbf{x}(u, v)$.

We now define $f(\theta, \hat{\mathbf{v}}) = (f_0, f_1, f_2, f_3)$ to be a quaternion describing the family of frames for which the direction $\hat{\mathbf{v}}$ is a preferred fixed axis of the frame, such as the tangent or normal vectors. The orthonormal triad of 3-vectors describing the desired frame is

$$F(\theta, \hat{\mathbf{v}}) = \begin{bmatrix} f_0^2+f_1^2-f_2^2-f_3^2 & 2f_1f_2-2f_0f_3 & 2f_1f_3+2f_0f_2 \\ 2f_1f_2+2f_0f_3 & f_0^2-f_1^2+f_2^2-f_3^2 & 2f_2f_3-2f_0f_1 \\ 2f_1f_3-2f_0f_2 & 2f_2f_3+2f_0f_1 & f_0^2-f_1^2-f_2^2+f_3^2 \end{bmatrix}, \quad (1)$$

where one column, typically the 3rd column, must be $\hat{\mathbf{v}}$.

The standard rotation matrix $R(\theta, \hat{\mathbf{v}})$ leaves $\hat{\mathbf{v}}$ fixed but does not have $\hat{\mathbf{v}}$ as one column of the 3×3 rotation matrix, and so we have more work to do. To compute $f(\theta, \hat{\mathbf{v}})$, we need the following:

- A base reference frame $b(\hat{\mathbf{v}})$ that is guaranteed to have, say, the 3rd column exactly aligned with a chosen vector $\hat{\mathbf{v}}$, which is either the tangent to a curve or the normal to a surface.
- A one-parameter family of rotations that leaves a fixed direction $\hat{\mathbf{v}}$ invariant.

The latter family of rotations is given simply by the standard quaternion

$$q(\theta, \hat{\mathbf{v}}) = (\cos\frac{\theta}{2}, \hat{\mathbf{v}}\sin\frac{\theta}{2}) , \quad (2)$$

for $0 \le \theta < 4\pi$, while the base frame can be chosen as

$$b(\hat{\mathbf{v}}) = q(\arccos(\hat{\mathbf{z}} \cdot \hat{\mathbf{v}}),\ (\hat{\mathbf{z}} \times \hat{\mathbf{v}})/\|\hat{\mathbf{z}} \times \hat{\mathbf{v}}\|) . \quad (3)$$

We refer hereafter to the frame $b(\hat{\mathbf{v}})$ as the *Geodesic Reference Frame* because it tilts the reference vector $\hat{\mathbf{z}}$ along a geodesic arc until it is aligned with $\hat{\mathbf{v}}$; see Figure 4. If $\hat{\mathbf{v}} = \hat{\mathbf{z}}$, there is no problem, since we just take $b(\hat{\mathbf{v}})$ to be the quaternion $(1, \mathbf{0})$; if $\hat{\mathbf{v}} = -\hat{\mathbf{z}}$,

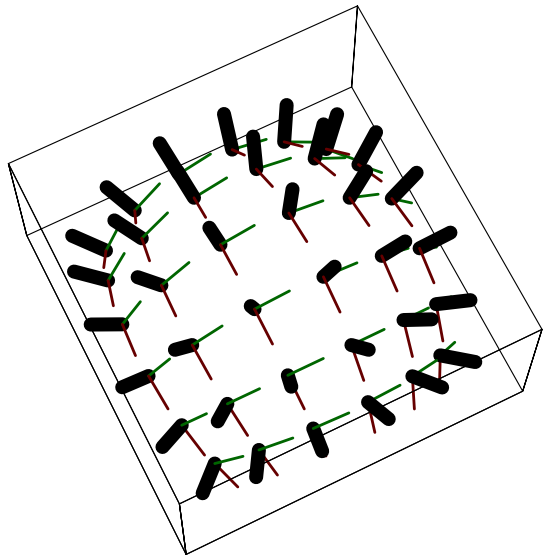


Figure 4: Example of the Geodesic Reference Frame: on the northern hemisphere of a 2-sphere, the Geodesic Reference Frame tilts the $\hat{\mathbf{z}}$ axis of the north pole's identity frame along the shortest arc to align with a specified reference direction.

we may choose any compatible quaternion such as $(0, 1, 0, 0)$. We escape the classic difficulty of being unable to assign a global frame to all of S^2 because we need a parameterization of *all possible* frames, not any one particular global frame. If one wants to use a reference frame that is not the identity frame, one must premultiply $b(\hat{\mathbf{v}})$ on the right by a quaternion rotating from the identity into that reference frame; this is important when constructing a nonstandard Geodesic Reference Frame such as that required to smoothly describe a neighborhood of the southern hemisphere of S^2.

We can thus write the full family of possible quaternion frames keeping $\hat{\mathbf{v}}$ as a fixed element of the frame triad to be the quaternion product

$$f(\theta, \hat{\mathbf{v}}) = q(\theta, \hat{\mathbf{v}}) * b(\hat{\mathbf{v}}) \ , \tag{4}$$

where $*$ denotes quaternion multiplication and all possible frames are described twice since $0 \leq \theta < 4\pi$. To summarize, if we specify a frame axis $\hat{\mathbf{v}}$ to be fixed, then the variable θ in $f(\theta, \hat{\mathbf{v}})$ serves to parameterize a *ring* in quaternion space, each point of which corresponds to a particular 3D frame, and each frame has a diametrically opposite twin.

Surface Patch Example. Figure 5 shows how the frame choice problem of Figure 2 can be visualized in the quaternion space of frames. We choose a quaternion projection that shows only the 3-vector part of the quaternion, dropping q_0. A frame choice is achieved by moving a point around the *sliding ring constraint* defined by Eq. (4) to the desired position. The constraint rings in Figure 5 are the generalizations to quaternion space of the constraint lines symbolized in Figure 3(b). The vertex A admits a family of frames $f(\theta, \hat{\mathbf{z}})$ that is a circle in quaternion space, but projects "edge-on" to a vertical bar in our default projection. The spaces of frames at the other vertices project as ellipses. The outer ring in Figure 5(b) is touched by two paths, corresponding to the clockwise and counterclockwise parallel transport routes in Figure 5(a); the gap between the intercepts in the outer ring corresponds to the inequivalence of the two frames at the bottom vertex of Figure 5(a).

Closed Curve Example. In Figure 6, we show a simple closed curve, the trefoil knot, the quaternion plot of its periodic Frenet frame, and, just to show we can do it, the entire constraint surface in which the Frenet frame and all other possible quaternion framings of the trefoil must lie. In the next section, we show the results of optimizing a continuous family of frames lying within this remarkable surface.

3 Minimal Frames

We have computed a wide selection of examples using the Evolver of K. Brakke [6] as our optimization tool. The Evolver is a public-domain, extensively documented system with a huge range of constraint-solving capabilities, widely used in mathematics and certain engineering problems. It has a very simple interface for handling parametric constraints like our sliding ring constraints, and can also handle a wide variety of energy functionals and boundary specifications. Most of the examples shown here take only a few seconds to stabilize in the Evolver; more complex geometries will of course take longer.

Two enhancements to the Evolver handle the specific issues related to quaternion optimization; the symmetry specification `symmetry_group "central_symmetry"` identifies the quaternion q with $-q$ if desired during the variation to prevent reflected double traversals from varying independently, and the system is able to use the pullback metric on the sphere

$$ds^2 = \sum_{i,j} dx_i \, dx_j \, r^{-4} \left(r^2 \, \delta_{i,j} - x_i \, x_j\right)$$

to compute distances directly on the quaternion three-sphere. Computation using this metric, however, is very slow, and so in practice we have used the Euclidean R^4 chord approximation, which works quite well for closely spaced samples and is much faster. (There are other choices of three-sphere parameterizations and quaternion distance measures that we have not yet attempted that could be more efficient still.) The energy functional that we chose to specify for the Evolver (or that would be implemented in a dedicated system) is thus simply the sum of the Euclidean lengths of each line segment in R^4:

$$d = \sum_{i,j} |x_i - x_j|$$

where $|q| = \sqrt{q \cdot q} = \sqrt{q_0 q_0 + q_1 q_1 + q_2 q_2 + q_3 q_3}$. For surface areas, the Evolver breaks polygons into triangles, computes their areas, and minimizes the total sum as the vertex positions vary.

Our own use of the Evolver required only changing the parameter "`#define BDRYMAX 20`" in `skeleton.h` to the desired (large) value corresponding to the number of desired sliding rings and recompiling. Then, remembering to set "`space_dimension 4`" when working in R^4, one needs in addition a piece of code similar to the following MATHEMATICA fragment to translate Eq. (4) into the boundary constraints for each fixed vector (tangent or normal) and the chosen initial quaternion reference frame:

```
Do[ring = Qprod[makeQfromVec[vlist[[i]],P1],
                qreflist[[i]]]//Chop;
   Write[file," boundary ",i," parame-
ters 1"];
   Write[file, "x1: ", CForm[ ring[[2]]]];
   Write[file, "x2: ", CForm[ ring[[3]]]];
   Write[file, "x3: ", CForm[ ring[[4]]]];
   Write[file, "x4: ", CForm[ ring[[1]]]],
   {i,1,Length[vlist]}]
```

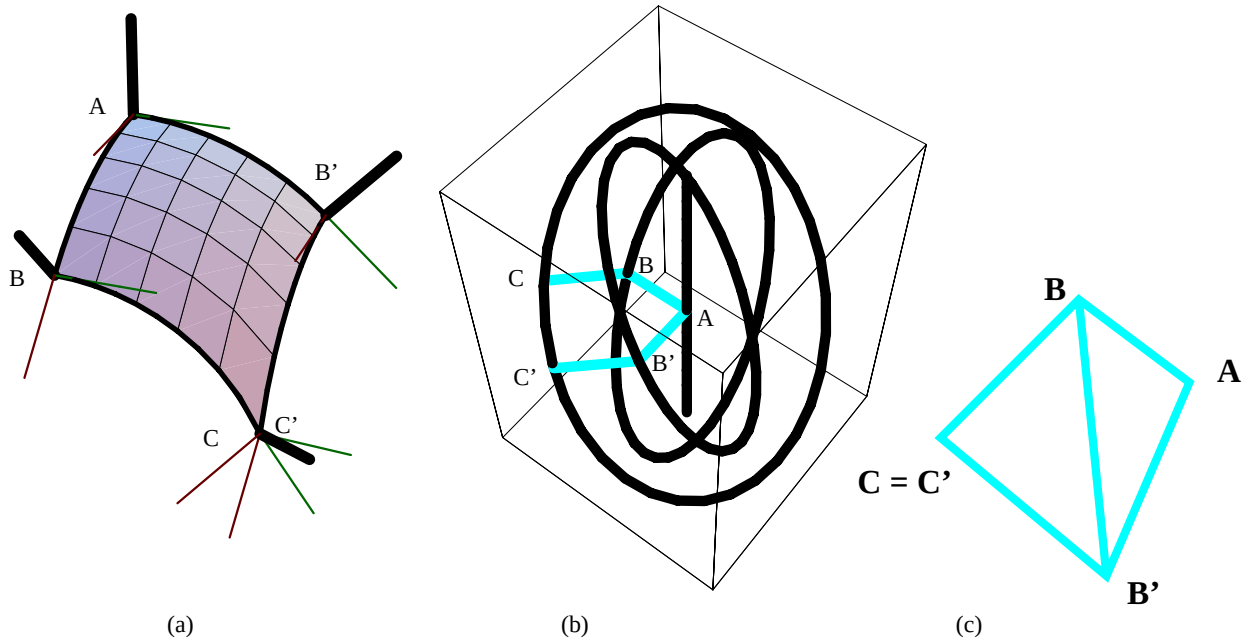


Figure 5: A different viewpoint of the mismatch problem of Figure 2. (a) Choosing different routes to determine the frame at the bottom point results in the incompatible frames shown here in 3D space. (b) The same information is presented here in the quaternion space-of-frames picture. We use throughout a quaternion projection that shows only the 3-vector part of the quaternion, dropping q_0; this is much like projecting away z in a polar projection of the 2-sphere. Each heavy black curve is a ring of possible frame choices that keep fixed the normals in (a); the labels mark the point in quaternion space corresponding to the frames at the corners in (a), so the gap between the labels C and C' represents the frame mismatch in quaternion space on the same constraint ring. (The apparent vertical line is the result of drawing a squashed circle of frames at vertex A in this projection.) (c) The method proposed in this paper to resolve this conflict is to fix one point, say A, divide the polygon $ABCB'$ into triangles, and slide B, C, and B' along the constraint rings until the total triangle areas are minimized, and some compromise with $C = C'$ is reached.

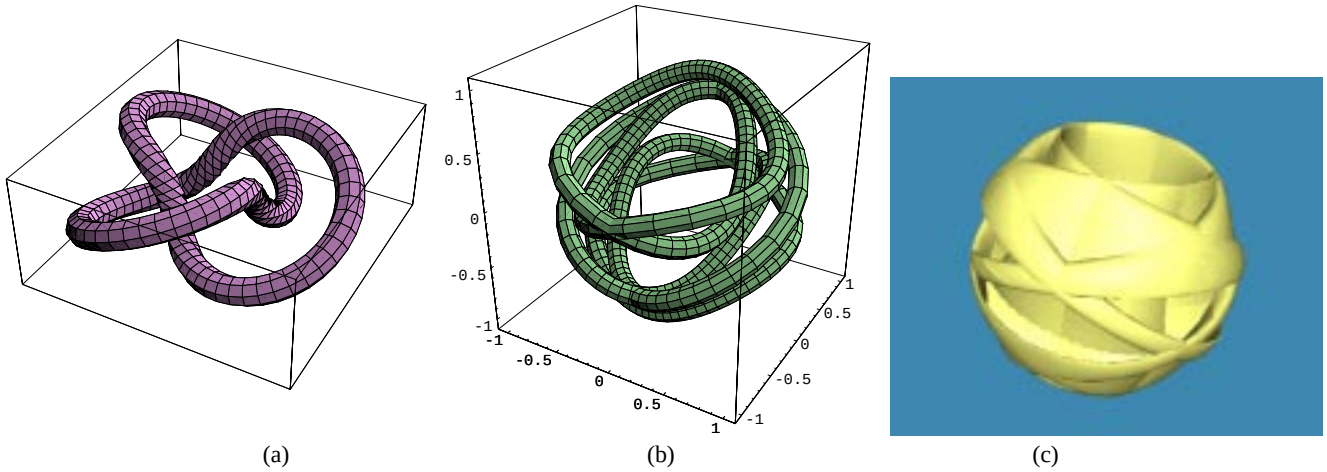


Figure 6: (a) A trefoil torus knot. (b) Its quaternion Frenet frame projected to 3D. For this trefoil knot, the frame does not close on itself in quaternion space unless the curve is traversed twice, corresponding to the double-valued "mirror" image of the rotation space that can occur in the quaternion representation. The longer segments in (b) correspond to the three high-torsion segments observable in (a). (c) The full constraint space for the trefoil knot is a very complex surface swept out by the constraint rings. All quaternions are projected to 3D using only the vector part.

Here `Qprod` and `makeQfromVec` perform the quaternion product and create the quaternion corresponding to Eq. (4) with `P1` replacing θ. Note that, since the Evolver displays only the first three coordinates, we have moved the scalar quaternion to the end; then the Evolver will display our preferred projection automatically.

With these preliminaries, the Evolver can easily be used to minimize the length of the total piecewise linear path among sliding ring constraints for selected curves, and the total area spanned by analogous sliding rings for surfaces. We made no attempt to go beyond piecewise linear curves. One interesting result is that there appear to be families of topologically distinct minima: depending on the conditions imposed, one may find either two disjoint curves (surfaces), one the $q \to (-q)$ image of the other, or a single quaternion curve (surface) that contains its own reflection, such as that in Figure 6(b). The families of frame manifolds containing their own reflected images have minima distinct from the disjoint families.

We now present some simple examples to give a feeling for the process.

Minimal Quaternion Frames for Space Curves. The helix provides a good initial example of the procedure we have formulated. We know that we can always find an initial framing of a curve based on the Geodesic Reference algorithm; however, suppose we wish to impose minimal length in quaternion space on the framing we select, and we do not know whether this frame is optimal with respect to that measure. Then, as illustrated in Figure 7, we can compute the ring constraints on the possible quaternion frames at each sample point and let the Evolver automatically find the optimal framing. The results for several stages of this evolution are shown in the Figure; the final configuration is indistinguishable from the parallel-transport frame, confirming experimentally our theoretical expectation that parallel transport produces the minimal possible twisting.

In Figure 1, we introduced the question of finding an optimal framing of a particular (3,5) torus knot whose almost-optimal parallel-transport framing was not periodic. In Figure 8, we show the solution to this problem achieved by clamping the initial and final quaternion frames to coincide, then letting the Evolver pick the shortest quaternion path for all the other frames.

The types of solutions we find are essentially the same for all reparameterizations of the curve; regardless of the spacing of the sampling, the continuous surface of possible frames is geometrically the same in quaternion space, so paths that are minimal for one sampling should be approximately identical to paths for any reasonable sampling. On the other hand, if we *want* special conditions for certain parameter values, it is easy to fix any number of particular orientations at other points on the curve, just as we fixed the starting points above; derivative values and smoothness constraints leading to generalized splines could be similarly specified (see, e.g., Barr et al. [2, 22]).

Surface Patch Framings. A classic simple example of a surface patch framing problem was presented in the discussion of Figures 2 and 5. This problem can also be handled naturally by the Evolver: we choose an initial quaternion frame for the mesh and minimize the area in quaternion space subject to the constraints that the normals remain unchanged. That is, the frame choices may only slide around constraint rings such as those depicted in Figure 5(b) for the frames at the corners. The results are shown in Figures 10 and 9. As a test, we started one case in a random initial state with a range of 2π in the starting values. All converged to the same optimal final framing. While more complex examples could be given, all the essential features of the method short of dealing rigorously with non-trivial topological manifolds are illustrated by this surface patch example.

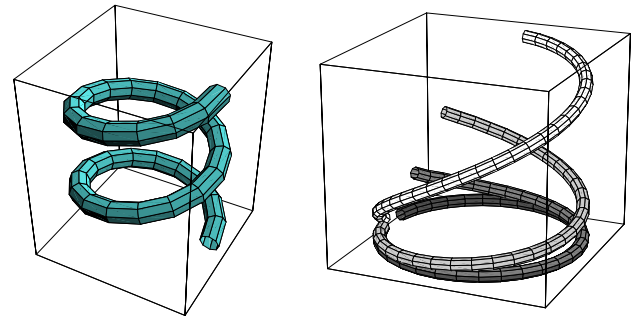


Figure 7: Helix (left) and its evolving quaternion frames (right). Starting from the Geodesic Reference quaternion frame for a single turn of the helix, the very dark gray circle, the Evolver produces these intermediate steps while minimizing the total quaternion curve length subject to the constraints in the space of frames. The final result is the white curve, which is identical to several decimal points with the parallel transport quaternion frame for the same helix; note that the *quaternion length* of the white curve is the shortest, even though in this projection that is not obvious. The numerical energies of the four curves, from dark to light in color, are 3.03, 2.91, 2.82, and 2.66 for the Parallel Transport frame. The individual tubings used to display these curves are in fact created using the parallel transport frame for each individual curve.

Manifolds. For general manifolds, one must treat patches one at a time in any event, since global frames may not exist at all. Although the locally optimal patches cannot be globally joined to one another, we conjecture that some applications might benefit from the next best thing: matching boundary frames of neighboring patches using transitional rotations (see, e.g., [18, 10]). We have carried this out explicitly for simple cases, but omit it here for brevity.

Extensions to Other Domains. We have focussed for expository purposes in this paper on frames with intrinsic natural constraints imposed by the tangents to curves and normals to surfaces. However, the method extends almost trivially to applications involving externally specified constraints on frames. Geometric construction algorithms based on extrusions reduce to the tubing problem. For ordinary camera control interpolation, one could constrain any direction of the camera frame to be fixed by calculating its appropriate constraint ring in the quaternion Gauss map, and then extend a method like that of Barr et al. [2, 22]) to smoothly compute intermediate frames subject to the constraints. For more general constrained navigation methods like those described by Hanson and Wernert [13]), the camera vertical direction could be fixed at chosen points over the entire constraint manifold, and the remaining frame parameters determined by optimization within the manifold of ring constraints, possibly subject to fixing entire key-frames at selected locations or boundaries.

4 Conclusion

We have introduced a general framework derived from the quaternion Gauss map for studying and selecting appropriate families of coordinate frames for curves and surface patches in 3D space. Minimizing length for quaternion curve maps and area for surfaces is proposed as the appropriate generalization of parallel transport for the selection of optimal frame fields. These smooth frames can be used to generate tubular surfaces based on the space curves, thus allowing their effective display on polygon-based graphics en-

gines with texturing. The analogous results for surface patches allow the selection of optimal local coordinate systems that may be adapted for display purposes and related applications such as texturing based on oriented particle systems. Our principal new tool is the space of all possible frames, a manifold of constraints immersed in the space of quaternion frames. By defining energies and boundary conditions in this space one can produce a rich variety of application-adapted criteria for specifying optimal families of frames. Work remaining to be done in the future includes applying the method to other domains such as geometric modeling and viewpoint interpolation, studying more carefully the topologically distinct minimal quaternion area solutions found for certain surface framings, and studying more challenging problems in the surface domain, e.g., topological tori with various numerical bumps and deformations are known to admit global frames, but little is known about how to compute good ones, and this method is a logical candidate.

Acknowledgments

The author 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 Ken Brakke for help with the Evolver. This research was made possible in part by NSF infrastructure grant CDA 93-03189.

References

[1] S. L. Altmann. *Rotations, Quaternions, and Double Groups*. Oxford University Press, 1986.

[2] Alan H. Barr, Bena Currin, Steven Gabriel, and John F. Hughes. Smooth interpolation of orientations with angular velocity constraints using quaternions. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 313–320, July 1992.

[3] M. Berger. *Geometry I, II*. Springer Verlag, Berlin, 1987.

[4] Richard L. Bishop. There is more than one way to frame a curve. *Amer. Math. Monthly*, 82(3):246–251, March 1975.

[5] Jules Bloomenthal. Calculation of reference frames along a space curve. In Andrew Glassner, editor, *Graphics Gems*, pages 567–571. Academic Press, Cambridge, MA, 1990.

[6] Kenneth A. Brakke. The surface evolver. *Experimental Mathematics*, 1(2):141–165, 1992. The "Evolver" system, manual, and sample data files are available by anonymous ftp from geom.umn.edu, The Geometry Center, Minneapolis MN.

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

[8] S. Gabriel and James T. Kajiya. Spline interpolation in curved space. In *State of the Art Image Synthesis*, 1985. Siggraph '85 Course notes.

[9] Alfred Gray. *Modern Differential Geometry of Curves and Surfaces with Mathematica*. CRC Press, Inc., Boca Raton, FL, second edition, 1998.

[10] Cindy M. Grimm and John F. Hughes. Modeling surfaces with arbitrary topology using manifolds. In *Computer Graphics Proceedings, Annual Conference Series*, pages 359–368, 1995. Proceedings of SIGGRAPH '95.

[11] A. J. Hanson and H. Ma. Quaternion frame approach to streamline visualization. *IEEE Trans. on Visualiz. and Comp. Graphics*, 1(2):164–174, June 1995.

[12] A. J. Hanson and Hui Ma. Visualizing flow with quaternion frames. In *Proceedings of Visualization '94*, pages 108–115. IEEE Computer Society Press, 1994.

[13] A. J. Hanson and E. Wernert. Constrained 3d navigation with 2d controllers. In *Proceedings of Visualization '97*, pages 175–182. IEEE Computer Society Press, 1997.

[14] J. T. Kajiya. Anisotropic reflection models. In *Computer Graphics*, volume 19, pages 15–21, 1985. Proceedings of SIGGRAPH '85.

[15] Myoung-Jun Kim, Myung-Soo Kim, and Sung Yong 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.

[16] F.. Klock. Two moving coordinate frames for sweeping along a 3d trajectory. *Computer Aided Geometric Design*, 3, 1986.

[17] N. L. Max. Computer representation of molecular surfaces. *IEEE Computer Graphics and Applications*, 3(5):21–29, Aug 1983.

[18] J. Milnor. *Topology from the Differentiable Viewpoint*. The University Press of Virginia, Charlottesville, 1965.

[19] 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.

[20] John C. Platt and Alan H. Barr. Constraint methods for flexible models. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 279–288, August 1988.

[21] D. Pletincks. Quaternion calculus as a basic tool in computer graphics. *The Visual Computer*, 5(1):2–13, 1989.

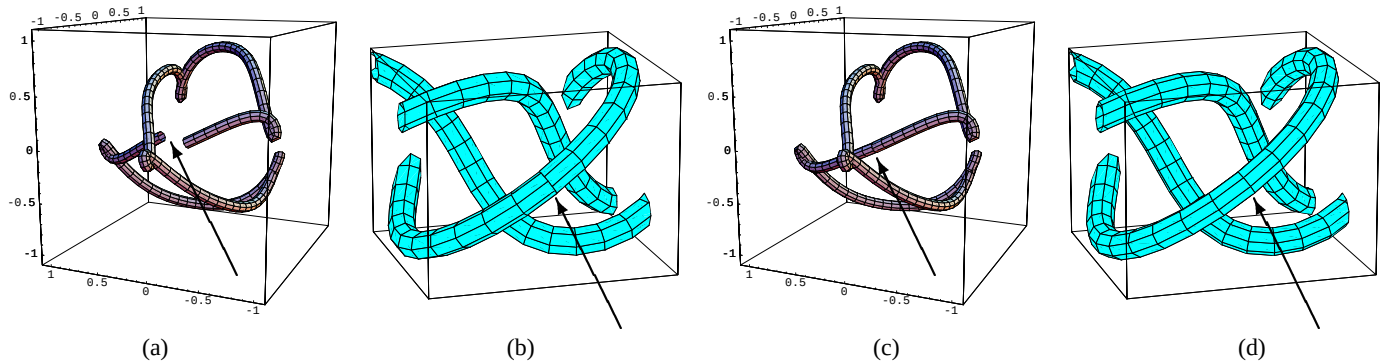
[22] Ravi Ramamoorthi and Alan H. Barr. Fast construction of accurate quaternion splines. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 287–292. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

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

[24] Uri Shani and Dana H. Ballard. Splines as embeddings for generalized cylinders. *Computer Vision, Graphics, and Image Processing*, 27:129–156, 1984.

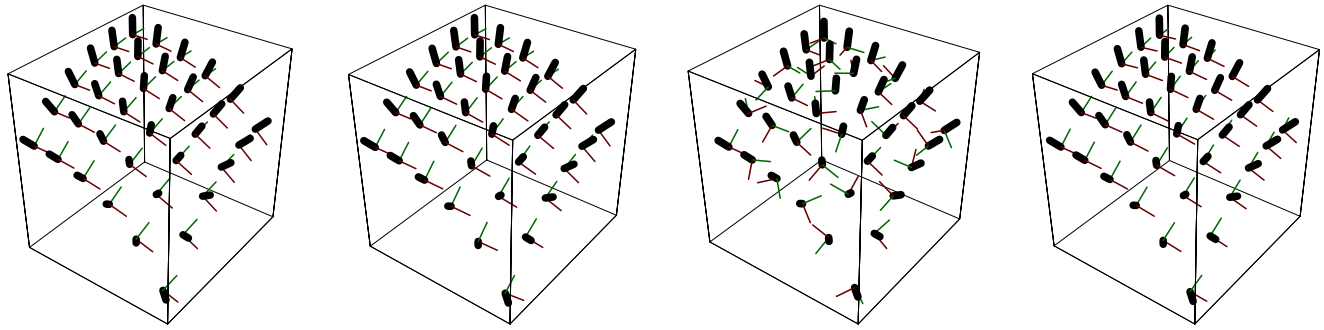
[25] K. Shoemake. Animating rotation with quaternion curves. In *Computer Graphics*, volume 19, pages 245–254, 1985. Proceedings of SIGGRAPH 1985.

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



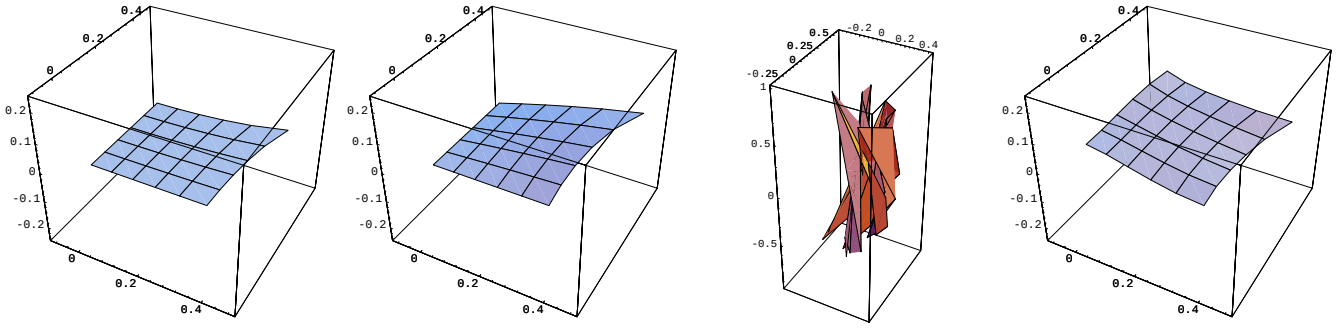
(a) (b) (c) (d)

Figure 8: Optimization of the non-periodic parallel transport frame of the (3,5) torus knot introduced in Figure 1 to produce a nearby periodic framing. (a) The original quaternion parallel transport frame used to produce the tubing in Figure 1(b,c). (b) The frame mismatch, repeated for completeness. (c) The result of fixing the final frame to coincide with the initial frame, leaving the other frames free to move on the constraint rings, and minimizing the resulting total length in quaternion space. The length of the original curve was 13.777 and that of the final was 13.700, not a large difference, but noticeable enough in the tube and the quaternion space plot. (d) Closeup of the corresponding framing of the knot in ordinary 3D space, showing that the mismatch problem has been successfully resolved. This tube can *now* be textured, since the frames match exactly.



(a) (b) (c) (d)

Figure 9: Study of *possible* and *optimal* reference frames on a surface patch; the corresponding quaternion fields are given in Figure 10. (a) The Geodesic Reference frames for the small patch of Figure 2. (b) Two-step parallel transport frames. (c) Random frames. (d) The unique frame configuration resulting from minimizing area in quaternion space with the upper corner fixed.



(a) (b) (c) (d)

Figure 10: Quaternion areas corresponding to the frame assignments in Figure 9. (a) The initial Geodesic Reference quaternions for the small patch shown in Figure 2. (b) Initial quaternions from parallel transporting the vertex frame down one edge, and then across line by line. (c) A random starting configuration with the single same fixed corner point as (a) and (b) and a range of $-\pi$ to $+\pi$ relative to the Geodesic Reference frame. (d) The result of minimization of the quaternion area is the same for all starting points. The relative areas are: 0.147, 0.154, 0.296, and 0.141, respectively. Thus the Geodesic Reference is very close to optimal, but is distinct.

TECHNICAL REPORT NO. 518

Quaternion Gauss Maps
and
Optimal Framings of Curves and Surfaces

by

Andrew J. Hanson

October 1998

COMPUTER SCIENCE DEPARTMENT
INDIANA UNIVERSITY
Bloomington, Indiana 47405-4101

Quaternion Gauss Maps and Optimal Framings of Curves and Surfaces

Andrew J. Hanson
Department of Computer Science
Lindley Hall 215
Indiana University
Bloomington, IN 47405 USA

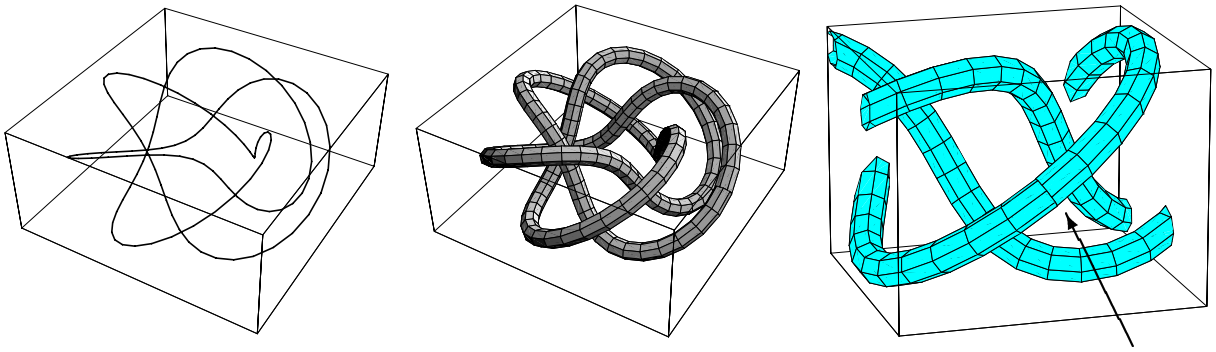
October 1998

Abstract

We propose a general paradigm for generating optimal coordinate frame fields that may be exploited to annotate and display curves and surfaces. Parallel-transport framings, which work well for open curves, generally fail to have desirable properties for cyclic curves and for surfaces. We suggest that minimal quaternion measure provides an appropriate generalization of parallel transport. Our fundamental tool is the "quaternion Gauss map," a generalization to quaternion space of the tangent map for curves and of the Gauss map for surfaces. The quaternion Gauss map takes 3D coordinate frame fields for curves and surfaces into corresponding curves and surfaces constrained to the space of possible orientations in quaternion space. Standard optimization tools provide application-specific means of choosing optimal, e.g., length- or area-minimizing, quaternion frame fields in this constrained space. We observe that some structures may have distinct classes of minimal quaternion framings, e.g, one disconnected from its quaternion reflection, and another that continuously includes its own quaternion reflection. We suggest an effective method for visualizing the geometry of quaternion maps that is used throughout. Quaternion derivations of the general moving-frame equations for both curves and surfaces are given; these equations are the quaternion analogs of the Frenet and Weingarten equations, respectively. We present examples of results of the suggested optimization procedures and the corresponding tubings of space curves and sets of frames for surfaces and surface patches.

1 Introduction

We propose a general framework for selecting optimal systems of coordinate frames that can be applied to the study of geometric structures such as curves and surfaces in three-dimensional space. The methods contain "minimal-turning" parallel-transport framings of curves as a special case,



(a) (b) (c)

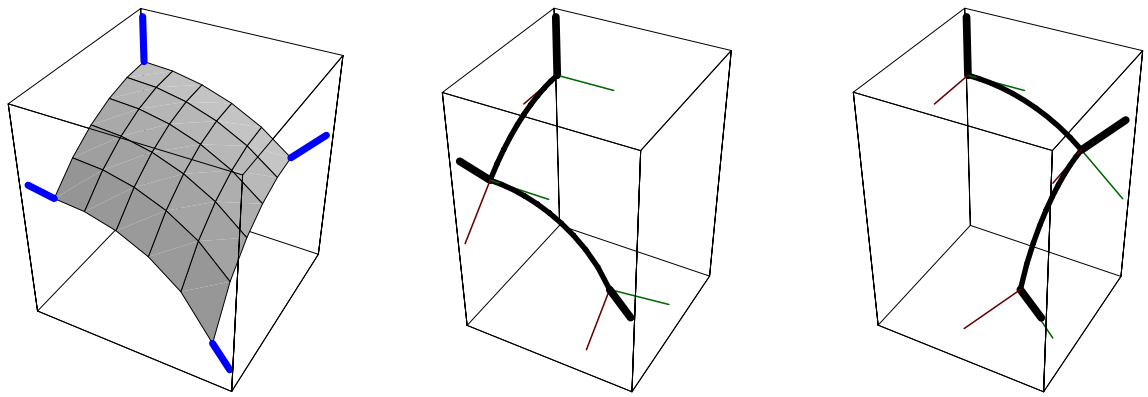
Figure 1: The (3,5) torus knot, a complex periodic 3D curve. (a) The line drawing is nearly useless as a 3D representation. (b) A tubing based on parallel transporting an initial reference frame produces an informative visualization, but is not periodic. (c) The arrow in this closeup exposes the subtle but crucial non-periodic mismatch between the starting and ending parallel-transport frames; this would invalidate any attempt to *texture* the tube. The methods of this paper provide robust parameterization-invariant principles for resolving such problems.

and extend naturally to situations where parallel-transport is not applicable. This article presents additional details of the IEEE Visualization '98 paper by the author [16].

Motivation. Many graphics problems require techniques for effectively displaying the properties of curves and surfaces. The problem of finding appropriate representations can be quite challenging. Representations of space curves based on single lines are often inadequate for graphics purposes; significantly better images result from choosing a "tubing" to display the curve as a graphics object with spatial extent. Vanishing curvature invalidates methods such as the Frenet frame, and alternative approaches such as those based on parallel transport involve arbitrary heuristics to achieve such properties as periodicity. Similar problems occur in the construction of suitable visualizations of complex surfaces and oriented particle systems on surfaces. If a surface patch is represented by a rectangular but nonorthogonal mesh, for example, there is no obvious local orthonormal frame assignment; if the surface has regions of vanishing curvature, methods based on directions of principal curvatures break down as well.

While we emphasize curves and surfaces in this paper to provide intuitive examples, there are several parallel problem domains that can be addressed with identical techniques. Among these are extrusion methods and generalized cones in geometric modeling, the imposition of constraints on a camera-frame axis in key-frame animation, and the selection of a 2D array of camera-frame axis choices as a condition on a constrained-navigation environment (see, e.g., Hanson and Wernert [20]).

Figure 1 summarizes the basic class of problems involving curves that will concern us here. The line drawing (a) of a (3,5) torus knot provides no useful information about the 3D structure.



(a) (b) (c)

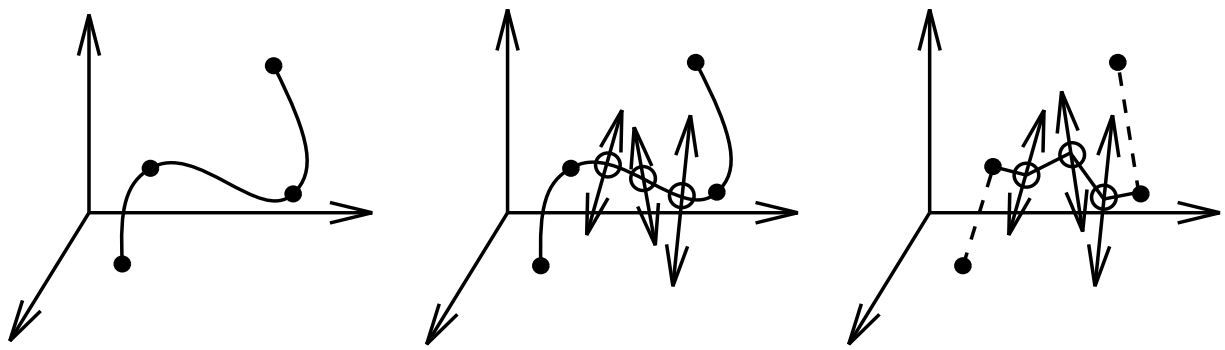
Figure 2: (a) A smooth 3D surface patch having a non-orthogonal parameterization, along with its geometrically-fixed normals at the four corners. No unique orthonormal frame is derivable from the parameterization. If we imitate parallel transport for curves to evolve the initial frame at the top corner to choose the frame at the bottom corner, we find that paths (b) and (c) result in incompatible final frames at the bottom corner. This paper addresses the problem of systematically choosing a compatible set of surface frames in situations like this.

Improving the visualization by creating a tubing involves a subtle dilemma that we attempt to expose in the rest of the figure. We cannot use a periodic Frenet frame as a basis for this tubing because inflection points or near-inflection points occur for many nice-looking torus knot parameterizations, and in such cases the Frenet frame is undefined or twists wildly. The parallel-transport tubing shown in (b) is well-behaved but not periodic; by looking carefully at the magnified portion next to the arrow in Figure 1(c), one can see a gross mismatch in the tessellation due to the non-periodicity which would, for example, preclude the assignment of a consistent texture. While it would be possible in many applications to ignore this mismatch, it has been the subject of a wide variety of previous papers (see, e.g., [24, 36, 5]), and must obviously be repaired for many other applications such as those requiring textured periodic tubes.

Figure 2 illustrates a corresponding problem for surface patches. While the normals to the four corners of the patch are always well-defined (a), one finds two different frames for the bottom corner depending upon whether one parallel transports the initial frame around the left-hand path (b) or the right-hand path (c). There is no immediately obvious right way to choose a family of frames covering this surface patch.

Our goal is to propose a systematic family of optimization methods for resolving problems such as these.

Methodology. We focus on unit quaternion representations of coordinate frames because of the well-known natural structure of unit quaternions as points on the three-sphere S^3, which admits a natural distance measure for defining optimization problems, and supports in addition a variety of regular frame-interpolation methods (see, e.g., [37, 35, 31, 23]). We do not address the related



(a) (b) (c)

Figure 3: (a) The camera frame interpolation problem is analogous to the problem of finding a minimal-bending spline curve through a series of fixed key points. (b) The optimal curve frame assignment problem is analogous to fixing the end points of a curve segment and choosing *in addition* a family of lines along which the intermediate points are constrained to slide during the optimization process; in 3D, the spline path need not pass through the constraint lines. (c) In this paper, our sample points are generally close enough together that we apply the constraints to piecewise linear splines analogous to those shown here.

question of optimal freely moving frames treated by the minimal-tangential-acceleration methods (see, e.g., [2, 34, 11]); we are instead concerned with closely-spaced points on curves and surfaces where one direction of the frame is already fixed, and the chosen functional minimization in quaternion space must obey the additional constraint imposed by the fixed family of directions. Additional references of interest, especially regarding the treatment of surfaces, include [22, 32]. Figure 3 provides a visualization of the difference between the general interpolation problem and our constrained problem: a typical spline minimizes the bending energy specified by the chosen anchor points; requiring intermediate points to slide on constrained paths during the minimization modifies the problem. In particular, 3D spline curves need not intersect any of the constraint paths. In addition, we note that we typically have already sampled our curves and surfaces as finely as we need, so that piecewise linear splines are generally sufficient for the applications we discuss.

Our solution to the problem is to transform the intrinsic geometric quantities such as the tangent field of a curve and the normal field of a surface to quaternion space and to construct the quaternion manifold corresponding to the one remaining degree of rotational freedom in the choice of coordinate frame at each point. Curves and surfaces in these *spaces of possible frames* correspond to specific choices of the *quaternion Gauss map*, a subspace of the space of possible quaternion frames of the object to be visualized. Mathematically speaking, the space of possible frames is the circular *Hopf fiber* lying above the point in S^2 corresponding to each specific curve tangent or surface normal (see, e.g., [39, 3]).

For space curves, specifying a frame assignment as a quaternion path leads at once to tubular surfaces that provide a “thickened” representation of the curve that interacts well with lighting and rendering models. For surface patches, the approach results in a structure equivalent to that of

an anisotropic oriented particle system whose pairs of tangent vector fields in the surface produce natural flow fields that characterize the local surface properties and are easy to display. We will see that certain complex features of surfaces that are well-known in manifold theory arise naturally and can be clearly visualized using the quaternion Gauss map.

In the course of the discussion, we introduce a useful method of visualizing the geometry of the space of quaternions in which quaternion Gauss maps and the spaces of possible quaternion frames are represented. We show how to compute the required subspaces of frames in practice, and how to express this information in a form that can be used to optimize an energy measure, thereby leading to optimal frame choices. We also outline in the appendix a treatment of the curve and surface frame differential equations expressed directly in quaternion coordinates using the quaternion Lie algebra; these methods expose essential fundamental features of the quaternion frame methodology that are analogous to the Frenet and Weingarten equations.

Parallel Transport and Minimal Measure. Constraining each quaternion point (a frame) to its own circular quaternion path (the axial degree of rotational freedom), we then minimize the quaternion length of the frame assignment for curves and the quaternion area of the frame assignment for surfaces to achieve an optimal frame choice; this choice reduces to the parallel-transport frame for simple cases. Our justification for choosing minimal quaternion length for curves is that there is a unique rotation in the plane of two neighboring tangents that takes each tangent direction to its next neighbor along a curve: this is the geodesic arc connecting the two frames in quaternion space, and is therefore the minimum distance between the quaternion points representing the two frames. The choice of minimal area for surface frames is more heuristic, basically a plausibility argument that the generalization of minimal length is minimal area; no doubt this could be made more rigorous.

By imposing other criteria such as endpoint derivative values and minimal bending energy (see Barr et al. [2, 34]), the short straight line segments and polygons that result from the simplest minimization could be smoothed to become generalized splines passing through the required constraint rings; since, in practice, our curve and surface samplings are arbitrarily dense, this was not pursued in the current investigation.

For space curves, specifying a frame assignment as a quaternion path leads at once to tubular surfaces that provide a "thickened" representation of the curve that interacts well with texturing, lighting, and rendering models. For surface patches, the approach results in a structure equivalent to that of an anisotropic oriented particle system (also a species of texture) whose pairs of tangent vector fields in the surface produce natural flow fields that characterize the local surface properties and are easy to display.

Background. General questions involving the specification of curve framings have been investigated in many contexts; for a representative selection of approaches, see, e.g., [24, 36, 5, 28]. The quaternion Gauss map is a logical extension of the quaternion frame approach to visualizing space curves introduced by Hanson and Ma [19, 18]. The formulation of the quaternion form of the differential equations for frame evolution was introduced as early as the 1890's by Tait [41].

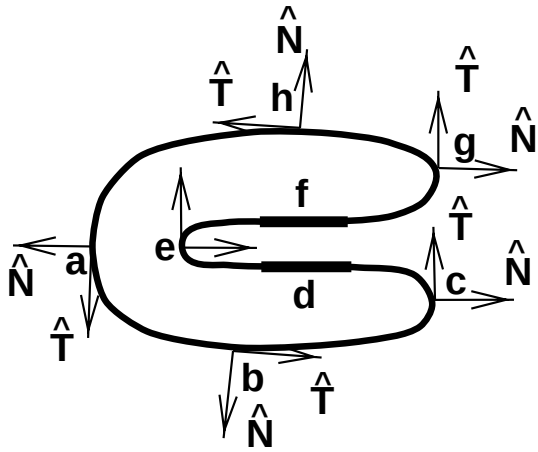


Figure 4: A smooth 2D curve with its normal and tangent frame fields. The segments d and f are intended to be straight.

For basic information on orientation spaces and their relationship to quaternions, see, e.g., [1, 33, 25]. Our own conventions are summarized in the appendix. The task of visualizing quaternions is also important, and we will describe our own approach below; for an interesting alternative, see Hart, Francis, and Kauffman [21]. Additional background on the differential geometry of curves and surfaces may be found in sources such as the classical treatise of Eisenhart [8] and in Gray's MATHEMATICA-based text [12], which inspired a number of the illustrations in this paper.

2 The Differential Geometry of Coordinate Frames

Our first goal is to define moving coordinate frames that are attached to curves and surfaces in 3D space. We will assume that our curves and surfaces are defined in practice by a discrete set of sample points connected by straight line segments, so that numerical derivatives can be defined at each point if analytic derivatives are not available. We begin with a pedagogical presentation of the properties of 2D curves, and then extend the surprisingly rich concepts that arise to 3D curves and surfaces.

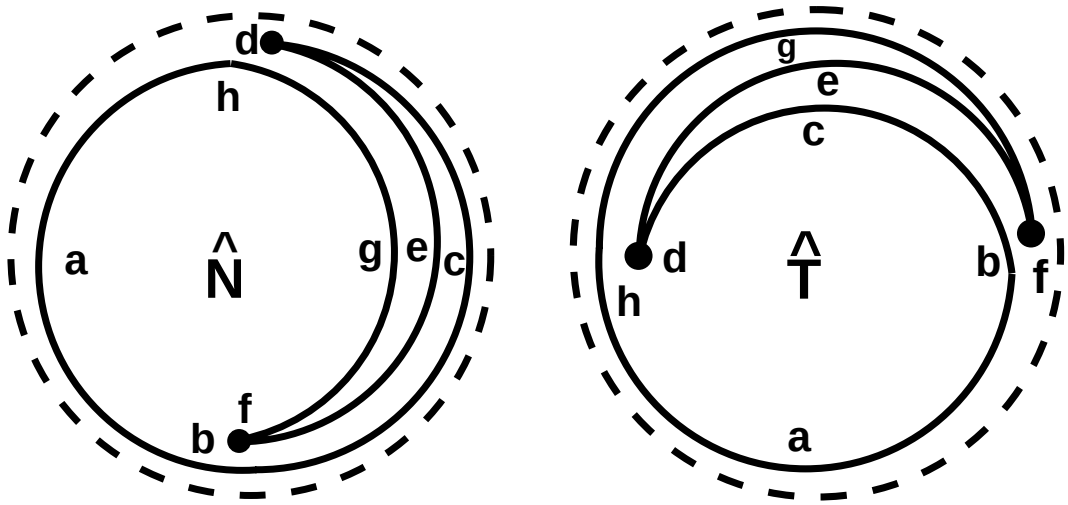
2.1 Orientation Maps of 2D Curves

Suppose we have a smooth, arbitrarily differentiable 2D curve $\mathbf{x}(t) = x(t)\hat{\mathbf{x}} + y(t)\hat{\mathbf{y}}$. The curve itself generates a continuous set of changing tangents and normals of the form

$$\mathbf{T}(t) = d\mathbf{x}(t)/dt = x'\hat{\mathbf{x}} + y'\hat{\mathbf{y}} \quad (1)$$

$$\mathbf{N}(t) = y'\hat{\mathbf{x}} - x'\hat{\mathbf{y}} \ . \quad (2)$$

We choose this relative orientation convention so that in any dimension the tangent vector is expressible as the positive-signed cross-product of the normal(s); see [15] for further details. Unit



(a) (b)

Figure 5: 2D Gauss map sketches of (a) the normal directions and (b) the tangent directions corresponding to the U-shaped curve in Figure 4. All these vectors lie on the unit circle in 2D. The straight line segments along d and f in Figure 4 correspond to single points in both maps.

length vectors will hereafter be distinguished with the conventional notation $\hat{\mathbf{v}} = \mathbf{v}/\|\mathbf{v}\|$, so the normalized tangent and normal directions are denoted by $\hat{\mathbf{T}}$ and $\hat{\mathbf{N}}$.

In Figure 4, we show an example of a 2D curve with its tangent and normal fields. The *normalized* tangent and normal fields have only one degree of freedom, which we denote by the angle $\theta(t)$; the column vectors $\hat{\mathbf{N}}$ and $\hat{\mathbf{T}}$ then represent a moving orthonormal coordinate frame that may be expressed in the form

$$\begin{bmatrix} \hat{\mathbf{N}} & \hat{\mathbf{T}} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} . \tag{3}$$

We may derive a 2D version of the frame equations by differentiating the frame to get

$$\hat{\mathbf{N}}'(t) = +v\kappa\hat{\mathbf{T}} \tag{4}$$

$$\hat{\mathbf{T}}'(t) = -v\kappa\hat{\mathbf{N}} , \tag{5}$$

where $\kappa(t)$ is the curvature and $v(t)$ is the "velocity" relative to the infinitesimal measure of curve length $ds^2 = \mathbf{dx}(t) \cdot \mathbf{dx}(t)$, that is, $d\theta(t)/dt = (ds/dt)(d\theta(s)/ds) = v(t)\kappa(t)$.

Note: we will find sign choices to be a subtle exercise throughout this paper. In Figure 4, the fact that the normal $\hat{\mathbf{N}}$ is chosen to point to the *outside* of a curve encircling an enclosed area in the right-hand sense makes the system inequivalent to the Frenet frame of the corresponding 3D curve, which would have $\hat{\mathbf{N}}$ pointing *inwards* everywhere except around the point e, and would be undefined along the straight segments d and f.

2D Tangent Map and Gauss Map. A 2D version of the Gauss map [8, 12] used in the classical differential geometry of surfaces follows when we discard the original curve in Figure 4 and restrict our view to show *only* the path of the normalized normals, as in Figure 5(a), or the normalized tangents, as in Figure 5(b); both vector fields take values only in the unit circle. We note that any sufficiently small open neighborhood of the curve has unique tangent and normal directions, up to the possibility of a shared limit point for straight segments such as d and f in Figure 4; over the whole curve, however, particular neighborhoods of directions may be repeated many times, resulting in an overlapping, non-unique 2D map, as indicated schematically in Figure 5. We will accept this as a feature, not necessarily a deficiency, of the construction.

2D "Quaternions." In the appendix, we present the details of a derivation of a quaternion-like approach to the representation of 2D frames that may be informative to some readers. A brief summary begins by noting that the normal and tangent vectors can be parameterized by a quadratic form in the two variables a and b as

$$\begin{bmatrix} \hat{\mathbf{N}} & \hat{\mathbf{T}} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} a^2 - b^2 & -2ab \\ 2ab & a^2 - b^2 \end{bmatrix}, \tag{6}$$

where imposing the constraint $a^2 + b^2 = 1$ guarantees orthonormality of the frame.

By taking derivatives and extracting common factors, we find that the single matrix equation

$$\begin{bmatrix} a' \\ b' \end{bmatrix} = \frac{1}{2} v(t) \begin{bmatrix} 0 & -\kappa \\ +\kappa & 0 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \end{bmatrix} \tag{7}$$

in the two variables with one constraint contains *both* the frame equations $\hat{\mathbf{T}}' = -v\kappa\hat{\mathbf{N}}$ and $\hat{\mathbf{N}}' = +v\kappa\hat{\mathbf{T}}$. The $\hat{\mathbf{N}}'$ and $\hat{\mathbf{T}}'$ equations are superficially a more complex set of two vector equations in four variables with three constraints. Equation (7) is effectively the *square root* of the frame equations. Rotations may be realized as complex multiplication in $(a + ib)$, and the pair $(a,\, b) = (\cos(\theta/2),\, \sin(\theta/2))$ parameterizes any rotation. Since $(a, b) \sim (-a, -b)$, the variables give a double covering of the space of rotations if we take the angular range from $0 \to 4\pi$ instead of 2π. These are precisely the properties we expect of quaternion representations of rotations.

2.2 3D Space Curves

We now move on to three-dimensional space curves. The fundamental difference in 3D is that, while the tangent direction is still determinable directly from the space curve, there is an additional degree of rotational freedom in the normal plane, the portion of the frame perpendicular to the tangent vector. This is indicated schematically in Figure 6.

Tangent Map. The tangent direction of a 3D curve at each point is given simply by taking the algebraic or numerical derivative of the curve at each sample point and normalizing the result. Each tangent direction thus has two degrees of freedom and lies on the surface of the two-sphere S^2. The curve resulting from joining the ends of neighboring tangents is the *tangent map* of the curve. As

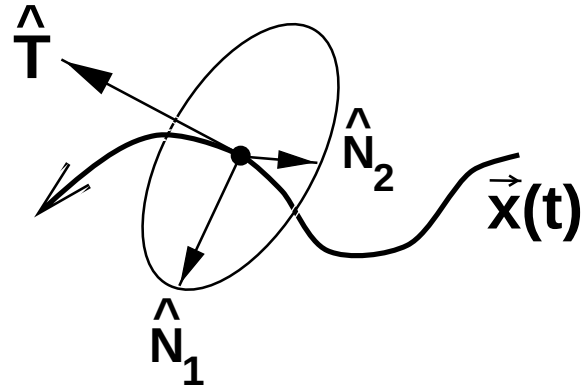


Figure 6: General form of a moving frame for a 3D curve $\mathbf{x}(t)$, with the tangent direction $\hat{\mathbf{T}}$ determined directly from the curve derivative, and the exact orientation of the basis $(\hat{\mathbf{N}}_1,\ \hat{\mathbf{N}}_2)$ for the normal plane determined only up to an axial rotation about $\hat{\mathbf{T}}$.

in the 2D case treated above, the tangent map of a 3D curve is not necessarily single-valued except in local neighborhoods, and may have limit points (e.g., if there are straight segments). In Figure 7(a,b), we show examples of two classic 3D curves, one a closed knot, the (2,3) trefoil knot lying on the surface of a torus, and the other the open helix:

$$\begin{aligned}
\mathbf{x}_{\text{torus}}(p,q)(a,b,c)(t) &= (a + b\cos(qt))\cos(pt)\,\hat{\mathbf{x}} + (a + b\cos(qt))\sin(pt)\,\hat{\mathbf{y}} + c\sin(qt)\,\hat{\mathbf{z}} \\
\mathbf{x}_{\text{helix}}(a,b,c)(t) &= a\cos(t)\,\hat{\mathbf{x}} + b\sin(t)\,\hat{\mathbf{y}} + ct\,\hat{\mathbf{z}}\ .
\end{aligned}$$

Differentiating these curves yields the tangent maps in Figure 7(c).

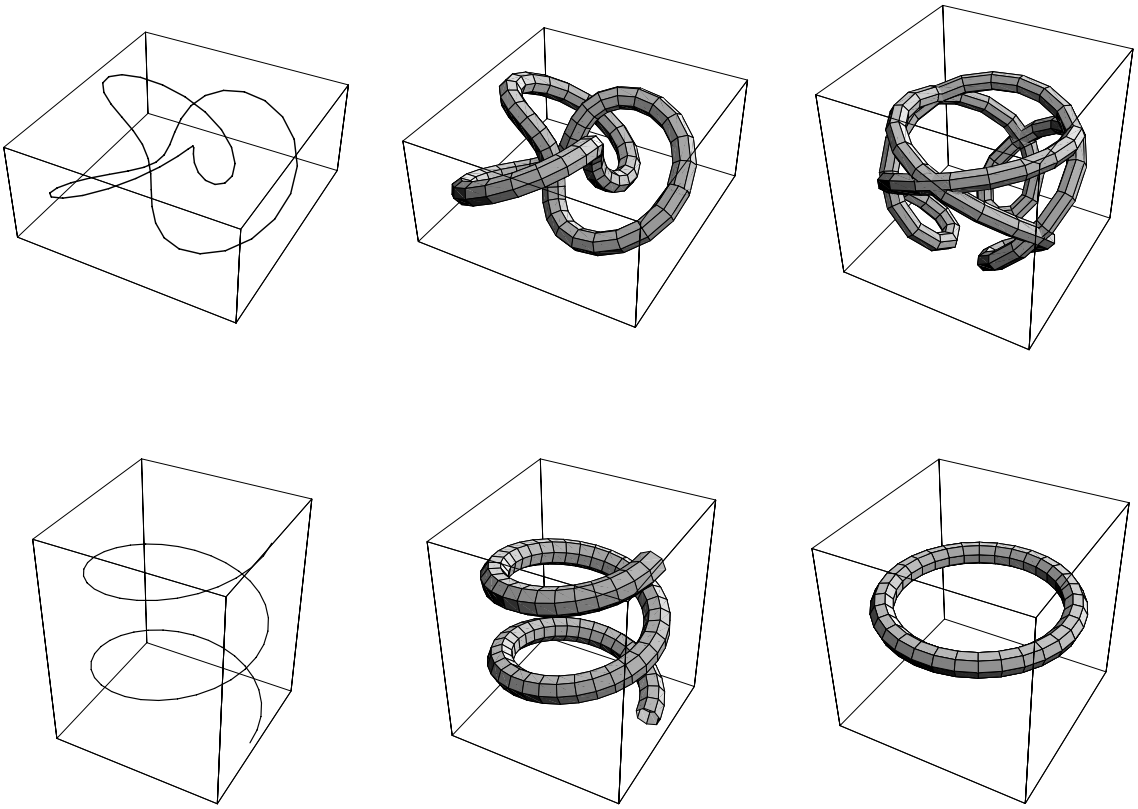
General Form of Curve Framings in 3D. The evolution properties of all possible frames for a 3D curve $\mathbf{x}(t)$ can be written in a unified framework. The basic idea is to consider an arbitrary frame to be represented in the form of columns of a 3×3 orthonormal rotation matrix,

$$\text{Curve Frame} = \left[\hat{\mathbf{N}}_1\,\hat{\mathbf{N}}_2\,\hat{\mathbf{T}}\right]\ . \tag{8}$$

Here $\hat{\mathbf{T}}(t) = \mathbf{x}'(t)/\|\mathbf{x}'(t)\|$ is the normalized tangent vector determined directly by the curve geometry, and which is thus unalterable; $(\hat{\mathbf{N}}_1(t),\ \hat{\mathbf{N}}_2(t))$ is a pair of orthonormal vectors spanning the plane perpendicular to the tangent vector at each point of the curve. Since $\|\hat{\mathbf{T}}\|^2 = \|\hat{\mathbf{N}}_1\|^2 = \|\hat{\mathbf{N}}_2\|^2 = 1$ and all other inner products vanish by definition, any change in a basis vector must be orthogonal to itself and thereby expressible in terms of the other two basis vectors. Thus the most general possible form for the frame evolution equations is

$$\begin{bmatrix} \hat{\mathbf{N}}_1'(t) \\ \hat{\mathbf{N}}_2'(t) \\ \hat{\mathbf{T}}'(t) \end{bmatrix} = v(t) \begin{bmatrix} 0 & +k_z(t) & -k_y(t) \\ -k_z(t) & 0 & +k_x(t) \\ +k_y(t) & -k_x(t) & 0 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{N}}_1(t) \\ \hat{\mathbf{N}}_2(t) \\ \hat{\mathbf{T}}(t) \end{bmatrix}\ , \tag{9}$$

where $v(t) = \|\mathbf{x}'(t)\|$ is the velocity of the curve if we are not using a unit speed parameterization.



(a) (b) (c)

Figure 7: Tangent maps. (a) The (2,3) torus knot and the helix as 3D line drawings. (b) Illustrating an application of tubing to make the 3D curves more interpretable. (c) The corresponding normalized tangent maps determined directly from the curve geometry. These are curves on the two-sphere, and have also been tubed to improve visibility.

The particular choice of notation and signs for the curvatures $\mathbf{k}$ in Eq. (9) is compellingly motivated by the quaternion Lie algebra treatment in the appendix, and its natural properties are also exposed using the Darboux form of the equations,

$$\begin{aligned} \hat{\mathbf{N}}_1' &= v(t)\,\mathbf{F} \times \hat{\mathbf{N}}_1 \\ \hat{\mathbf{N}}_2' &= v(t)\,\mathbf{F} \times \hat{\mathbf{N}}_2 \\ \hat{\mathbf{T}}' &= v(t)\,\mathbf{F} \times \hat{\mathbf{T}}\ , \end{aligned} \tag{10}$$

where $\mathbf{F}$ generalizes the Darboux vector field (see, e.g., Gray [12], p. 205):

$$\mathbf{F} = k_x \hat{\mathbf{N}}_1 + k_y \hat{\mathbf{N}}_2 + k_z \hat{\mathbf{T}}\ . \tag{11}$$

The square magnitude of the total "force" acting on the frame is $\|\mathbf{F}\|^{\mathbf{2}} = k_x^2 + k_y^2 + k_z^2$, and we will see below that this is a minimum for the parallel-transport frame.

The arbitrariness of the basis $(\hat{\mathbf{N}}_1(t),\, \hat{\mathbf{N}}_2(t))$ for the plane perpendicular to $\hat{\mathbf{T}}(t)$ can be exploited as desired to eliminate any one of the (k_x, k_y, k_z) (see, e.g., [4]). For example, if

$$\begin{aligned} \hat{\mathbf{M}}_1 &= \hat{\mathbf{N}}_1 \cos\theta - \hat{\mathbf{N}}_2 \sin\theta \\ \hat{\mathbf{M}}_2 &= \hat{\mathbf{N}}_1 \sin\theta + \hat{\mathbf{N}}_2 \cos\theta\ , \end{aligned} \tag{12}$$

differentiating and substituting Eq. (9) yields

$$\hat{\mathbf{M}}_1' = \hat{\mathbf{M}}_2 (k_z - \theta') - \hat{\mathbf{T}}(k_x \sin\theta + k_y \cos\theta) \tag{13}$$

$$\hat{\mathbf{M}}_2' = -\hat{\mathbf{M}}_1 (k_z - \theta') + \hat{\mathbf{T}}(k_x \cos\theta - k_y \sin\theta)\ . \tag{14}$$

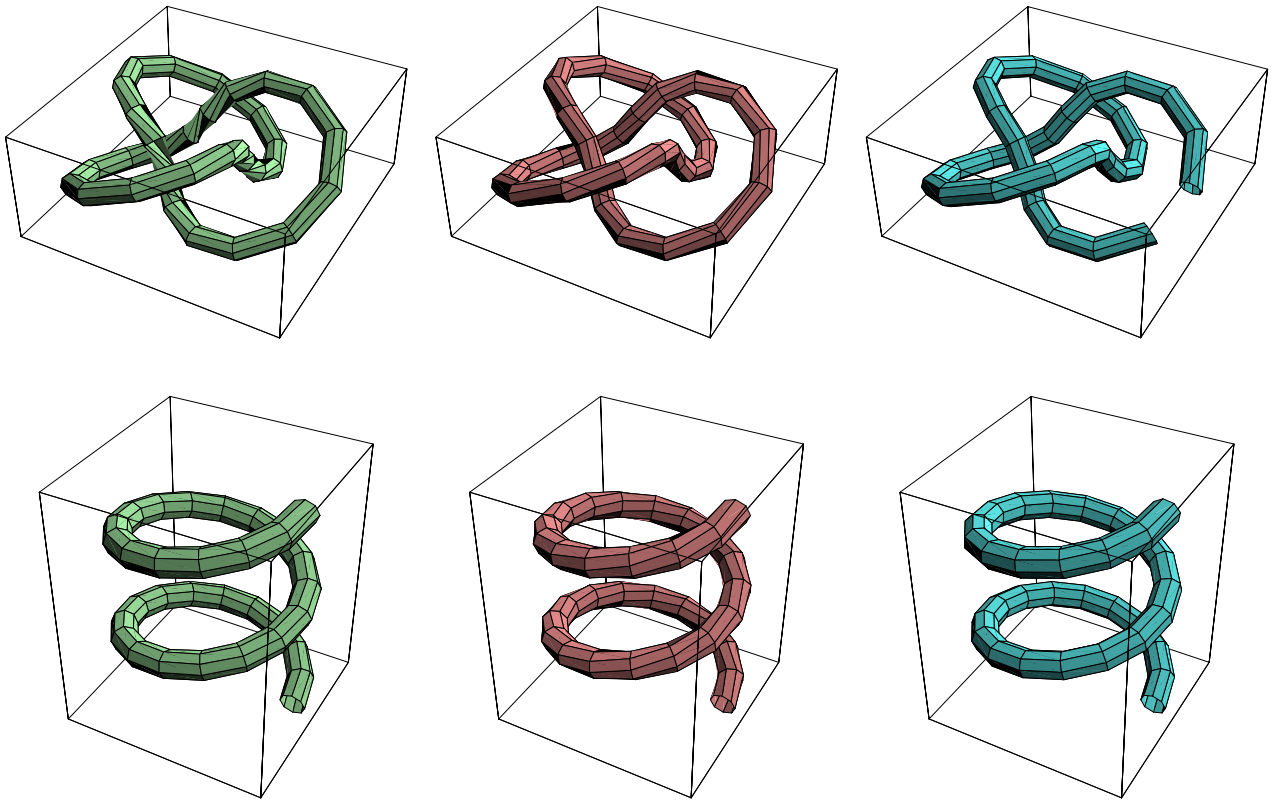
Thus the angle $\theta(t)$ may be chosen to cancel the angular velocity k_z in the $(\hat{\mathbf{N}}_1(t),\, \hat{\mathbf{N}}_2(t))$ plane. The same argument holds for any other pair. Attempting to eliminate additional components produces new mixing, leaving at least two independent components in the evolution matrix.

Tubing. For completeness, we note that to generate a ribbon or tube such as those used to display curves throughout this paper, one simply sweeps the chosen set of frames through each curve point $\mathbf{p}(t)$ to produce a connected tube,

$$\mathbf{x}(t, \theta) = \mathbf{p}(t) + \cos\theta\, \hat{\mathbf{N}}_1(t) + \sin\theta\, \hat{\mathbf{N}}_2(t)\ .$$

The resulting structure is sampled in t and over one full 2π period in θ to produce a tessellated tube. Arbitrary functions of $(t,\, \theta)$ can be introduced instead of the cosine and sine to produce ribbons and general linear structures.

Classical Frames. We now note a variety of approaches to assigning frames to an entire 3D space curve, each with its own peculiar advantages. Figure 8 compares the tubings of the (2,3) trefoil knot and the helix for each of the three frames described below.



(a) (b) (c)

Figure 8: Curve framings for the (2,3) torus knot and the helix based on (a) Frenet frame, (b) Geodesic Reference frame (minimal tilt from North pole), and (c) Parallel Transport frame, which is not periodic like the other frames.

- **Frenet-Serret Frame.** This classical frame is determined by local conditions at each point of the curve, but is undefined whenever the curvature vanishes (e.g., when the curve straightens out or has an inflection point). For the Frenet frame, $k_x = 0$, k_y is the inverse radius of curvature, i.e., the curvature $\kappa(t)$, and $k_z(t)$ is the torsion $\tau(t)$, which mixes the two normal vectors in their local plane. This choice produces the usual equations

$$\begin{bmatrix} \hat{\mathbf{T}}'(t) \\ \hat{\mathbf{N}}'(t) \\ \hat{\mathbf{B}}'(t) \end{bmatrix} = v(t) \begin{bmatrix} 0 & \kappa(t) & 0 \\ -\kappa(t) & 0 & \tau(t) \\ 0 & -\tau(t) & 0 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{T}}(t) \\ \hat{\mathbf{N}}(t) \\ \hat{\mathbf{B}}(t) \end{bmatrix} . \tag{15}$$

  Note that the squared Darboux vector is thus $\|\mathbf{F}\|^{\mathbf{2}} = \kappa^2 + \tau^2 \geq \kappa^2$.

  If $\mathbf{x}(t)$ is any thrice-differentiable space curve, we can identify the triad of normalized Frenet frame vectors directly with the local derivatives of the curve,

$$\begin{aligned} \hat{\mathbf{T}}(t) &= \frac{\mathbf{x}'(t)}{\|\mathbf{x}'(t)\|} \\ \hat{\mathbf{N}}_1 = \hat{\mathbf{N}}(t) &= \hat{\mathbf{B}}(t) \times \hat{\mathbf{T}}(t) \\ \hat{\mathbf{N}}_2 = \hat{\mathbf{B}}(t) &= \frac{\mathbf{x}'(t) \times \mathbf{x}''(t)}{\|\mathbf{x}'(t) \times \mathbf{x}''(t)\|} , \end{aligned} \tag{16}$$

  with $\kappa = \|\mathbf{x}'(t) \times \mathbf{x}''(t)\| / \|\mathbf{x}'(t)\|^3$, $\tau = \mathbf{x}'(t) \times \mathbf{x}''(t) \cdot \mathbf{x}'''(t) / \|\mathbf{x}'(t) \times \mathbf{x}''(t)\|^2$. For further details, see [8, 12].

- **Parallel-Transport Frame.** This frame is equivalent to a heuristic approach that has been frequently used in graphics applications (see, e.g., [24, 36, 5, 28]). A careful mathematical treatment by Bishop [4] presents its differential properties in a form that can be easily compared with the standard features of the Frenet frame. The parallel transport frame is distinguished by the fact that it uses the smallest possible rotation at each curve sample to align the current tangent vector with the next tangent vector. The current orientation of the plane normal to the tangent vector depends on the history of the curve, starting with an arbitrary initial frame, and so one is essentially integrating a differential equation for the frame change around the curve. The frame depends on the initial conditions, and cannot be determined locally on the curve like the Frenet frame. The algorithm with the best limiting properties [27] for computing this frame involves determining the normal direction $\hat{\mathbf{N}} = \mathbf{T}_i \times \mathbf{T}_{i+1} / \|\mathbf{T}_i \times \mathbf{T}_{i+1}\|$ to the plane of two successive tangents to the curve, finding the angle $\theta = \arccos(\hat{\mathbf{T}}_i \cdot \hat{\mathbf{T}}_{i+1})$, and rotating the current frame to the next frame using the 3×3 matrix $R(\theta, \hat{\mathbf{N}})$ or its corresponding quaternion (see appendix)

$$q(\theta,\, \hat{\mathbf{N}}) = q(\arccos(\hat{\mathbf{T}}_i \cdot \hat{\mathbf{T}}_{i+1}), \mathbf{T}_i \times \mathbf{T}_{i+1} / \|\mathbf{T}_i \times \mathbf{T}_{i+1}\|) . \tag{17}$$

  If the successive tangents are collinear, one leaves the frame unchanged; if the tangents are anti-collinear, a result can be returned, but it is not uniquely determined.

To identify the parallel transport frame with Eq. (9), we set $k_y \Rightarrow k_1$, $-k_x \Rightarrow k_2$, and $k_z = 0$ to avoid unnecessary mixing between the normal components (effectively the definition of parallel transport); this choice produces Bishop's frame equations,

$$\begin{bmatrix} \hat{\mathbf{N}}_1'(t) \\ \hat{\mathbf{N}}_2'(t) \\ \hat{\mathbf{T}}'(t) \end{bmatrix} = v(t) \begin{bmatrix} 0 & 0 & -k_1(t) \\ 0 & 0 & -k_2(t) \\ k_1(t) & k_2(t) & 0 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{N}}_1(t) \\ \hat{\mathbf{N}}_2(t) \\ \hat{\mathbf{T}}(t) \end{bmatrix} . \tag{18}$$

Since $\|\hat{\mathbf{T}}'\|^2 = (k_1)^2 + (k_2)^2$ is an invariant independent of the choice of the normal frame, Bishop identifies the curvature, orientation, and angular velocity

$$\begin{aligned} \kappa(t) &= \left((k_1)^2 + (k_2)^2\right)^{1/2} \\ \theta(t) &= \arctan\left(\frac{k_2}{k_1}\right) \\ \omega(t) &= \frac{d\theta(t)}{dt} . \end{aligned}$$

k_1 and k_2 thus correspond to a Cartesian coordinate system for the "curvature polar coordinates" (κ, θ) with $\theta = \theta_0 + \int \omega(t)\, dt$; $\omega(t)$ is effectively the classical torsion $\tau(t)$ appearing in the Frenet equations. Note that the squared Darboux vector $\|\mathbf{F}\|^{\mathbf{2}} = \|\hat{\mathbf{T}}'\|^2 = k_1^2 + k_2^2 = \kappa^2$ is now a frame invariant. It is missing the torsion component present for the Frenet frame, and thus assumes its minimal value.

- **Geodesic Reference Frame.** In this paper, we will often need a frame that is guaranteed to have a particular axis in one direction, but we will not care about the remaining axes because they will be considered as a space of possibilities. A convenient frame with these properties can always be constructed starting from the assumption that there exists a canonical reference frame in which, say, the $\hat{\mathbf{z}}$ axis corresponds to the preferred direction. Thus if $\hat{\mathbf{v}}$ is the desired direction of the new axis, we can simply tilt the reference axis $\hat{\mathbf{z}}$ into $\hat{\mathbf{v}}$ along a minimal, geodesic curve using an ordinary rotation $R(\theta, \hat{\mathbf{n}})$ or its corresponding quaternion (see appendix):

$$q(\theta, \hat{\mathbf{n}}) = q(\arccos(\hat{\mathbf{z}} \cdot \hat{\mathbf{v}}),\, \hat{\mathbf{z}} \times \hat{\mathbf{v}} / \|\hat{\mathbf{z}} \times \hat{\mathbf{v}}\|) . \tag{19}$$

  Clearly any reference frame, including frames related to the viewing parameters of a moving observer, could be used instead of $\hat{\mathbf{z}}$. This frame has the drawback that it is ambiguous whenever $\hat{\mathbf{v}} = -\hat{\mathbf{z}}$; sequences of frames passing through this point will not necessarily be smoothly varying since only a single instance of a one-parameter family of frames can be returned automatically by a context-free algorithm. Luckily, this is of no consequence for our application. As we will discuss later in the quaternion framework, this property is directly related to the absence of a global vector field on the two-sphere.

- **General Frames.** We will henceforth work with the general framework for coordinate frames of arbitrary generality, rather than choosing conventional frames or hybrids of the

frames described so far (see, e.g., Klock [24]). While the classical frames have many fundamentally appealing mathematical properties, we are not in fact restricted to use any one of them. Keeping the tangent vector field intact, we may modify the angle of rotation about the tangent vector at will to produce an application-dependent frame assignment. An example of such an application is a closed curve with inflection points: the Frenet frame is periodic but not globally defined, the parallel transport frame will not be periodic in general, and the Geodesic Reference frame will be periodic but may have discontinuities for antipodal orientations. Thus, to get a satisfactory smooth global frame, we need something close to a parallel transport frame but with a periodic boundary condition; an example of an ad hoc solution is to take the Parallel Transport frame and impose periodicity by adding to each vertex's axial rotation a fraction of the angular deficit of the parallel transport frame after one circuit. But this is highly heuristic and depends strongly on the chosen parameterization. In the following sections, we introduce a more comprehensive approach.

2.3 3D Surfaces

If we are given a surface patch $\mathbf{x}(u, v)$ with some set of non-degenerate coordinates (u, v), we may determine the normals at each point by computing

$$\mathbf{N}(u, v) = \mathbf{x}_u \times \mathbf{x}_v \ , \tag{20}$$

where $\mathbf{x}_u = \partial\mathbf{x}/\partial u$ and $\mathbf{x}_v = \partial\mathbf{x}/\partial v$. For surfaces defined numerically in terms of vertices and triangles, we would choose a standard procedure such as averaging the normals of the faces surrounding each vertex to determine the vertex normal. Alternatively, if we have an implicit surface described by the level-set function $f(\mathbf{x}) = 0$, the normals may be computed directly from the gradient at any point $\mathbf{x}$ satisfying the level set equation:

$$\mathbf{N}(\mathbf{x}) = \nabla f(\mathbf{x}) \ .$$

The normalized normal is defined as usual by $\hat{\mathbf{N}} = \mathbf{N}/\|\mathbf{N}\|$.

For 3D curves, the geometry of the curve determined the tangent vector $\hat{\mathbf{T}}$ and left a pair of normal vectors $(\hat{\mathbf{N}}_1, \hat{\mathbf{N}}_2)$ with one extra degree of freedom to be determined in the total frame $[\hat{\mathbf{N}}_1\, \hat{\mathbf{N}}_2\, \hat{\mathbf{T}}]$. The analogous observation for surfaces is that the geometry fixes the *normal* at each surface point, leaving a pair of *tangent* vectors $(\hat{\mathbf{T}}_1, \hat{\mathbf{T}}_2)$ with one extra degree of freedom to be determined in the total surface frame,

$$\text{Surface Frame} = \left[\hat{\mathbf{T}}_1\, \hat{\mathbf{T}}_2\, \hat{\mathbf{N}}\right] \ . \tag{21}$$

When a (u, v) surface parameterization is available, the surface partial derivatives $\mathbf{x}_u$ and $\mathbf{x}_v$ can in principle be used to assign a frame $\left[\hat{\mathbf{T}}_1\, \hat{\mathbf{T}}_2\, \hat{\mathbf{N}}\right]$ (using Gram-Schmidt if $\mathbf{x}_u \cdot \mathbf{x}_v \neq 0$), but there is no reason to believe that this frame has any special properties in general. In practice, it is extremely convenient to define a rectangular mesh on the surface patch, and a grid parameterized by (u, v) typically serves this purpose.

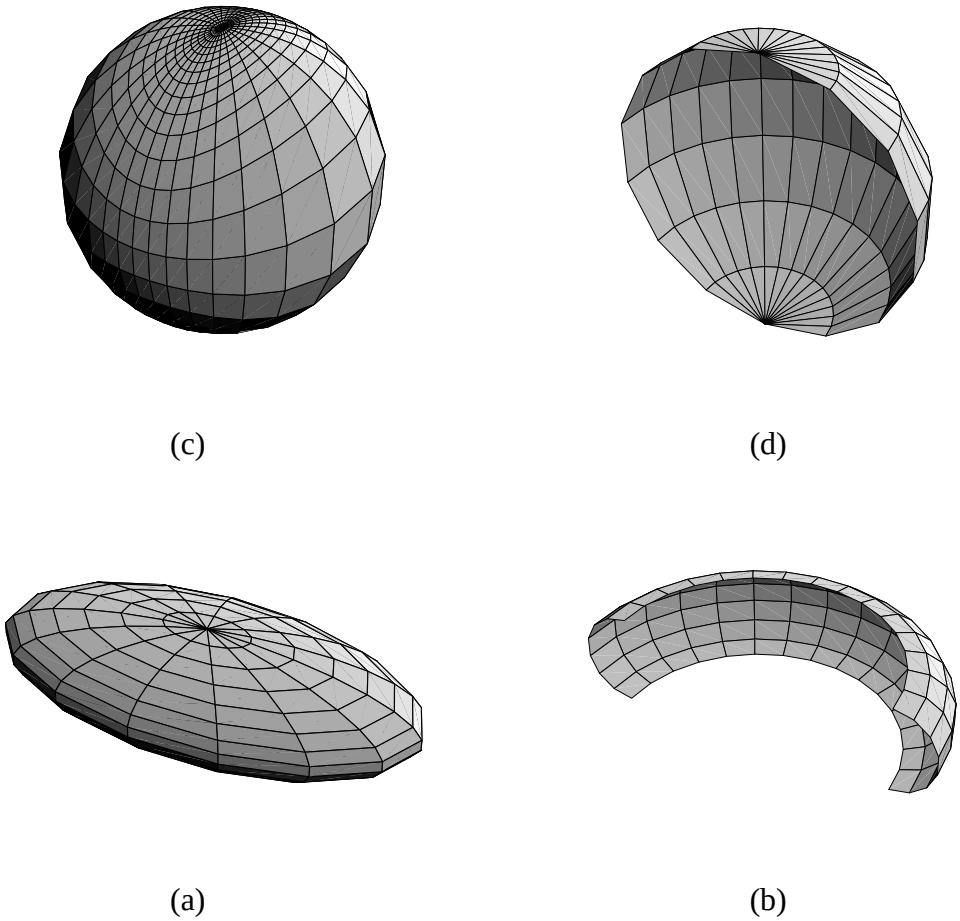


Figure 9: Classical Gauss maps of surfaces. (a) An ellipsoid and (b) a portion of a torus. (c,d) The corresponding standard Gauss maps of the normal vectors onto the sphere. Patches with coincident normals (e.g., for the full torus) would overlap in this representation.

Classical Gauss Map. The surface analog of the tangent map of a curve is the Gauss map, which takes a selection of points on the surface, typically connected by a mesh of some sort, and associates to each point its normalized surface normal. The Gauss map is then the plot of each of these normals in the coordinate system of a unit sphere S^2 in R^3. The Gauss map is guaranteed to be unique in some sufficiently small open set of each point of a regular surface, but may be arbitrarily multiple valued for the entire surface; note also that many nearby surface points can be mapped to a single point in the Gauss map, e.g., for certain types of planar curves in the surface or a planar area patch.

In Figure 9, we show a coordinate mesh on an elliptical surface and its single-valued Gauss map, as well as a quarter of a torus and its Gauss map; the Gauss map of the entire torus would cover the sphere twice, and there are two entire circles on the torus that correspond to single points, the North and South poles, in the Gauss map.

Surface Frame Evolution. The equations for the evolution of a surface frame follow the same basic structure as those of a space curve, except the derivatives are now directional, with two linearly independent degrees of freedom corresponding to the tangent basis $(\hat{\mathbf{T}}_1,\ \hat{\mathbf{T}}_2)$ in the surface. Typically (see [8, 12]), one assumes a not-necessarily-orthogonal parameterization (u, v) that permits one to express the tangent space in terms of the partial derivatives $(\mathbf{x}_u,\ \mathbf{x}_v)$, giving the normals $\hat{\mathbf{N}}(u, v)$ of Eq. (20). Then one can express the local curvatures in terms of any linearly independent pair of vector fields $(\mathbf{U},\ \mathbf{V})$ as

$$D_{\mathbf{U}}\hat{\mathbf{N}} \times D_{\mathbf{V}}\hat{\mathbf{N}} = K\ (\mathbf{U} \times \mathbf{V}) \tag{22}$$

$$D_{\mathbf{U}}\hat{\mathbf{N}} \times \mathbf{V} + \mathbf{U} \times D_{\mathbf{V}}\hat{\mathbf{N}} = 2H\ (\mathbf{U} \times \mathbf{V})\ . \tag{23}$$

With $\mathbf{U} = \mathbf{x}_u \cdot \nabla$ and $\mathbf{V} = \mathbf{x}_v \cdot \nabla$, we get the classical expressions. As Gray succinctly notes, since all the derivatives of $\hat{\mathbf{N}}$ are perpendicular to $\hat{\mathbf{N}}$, the whole apparatus amounts to constructing the tangent map of the Gauss map.

If we try to build the geometry of surfaces from a parametric representation, then each directional derivative has a a vector equation of the form of Eq. (9). Thus we may write equations of the general form

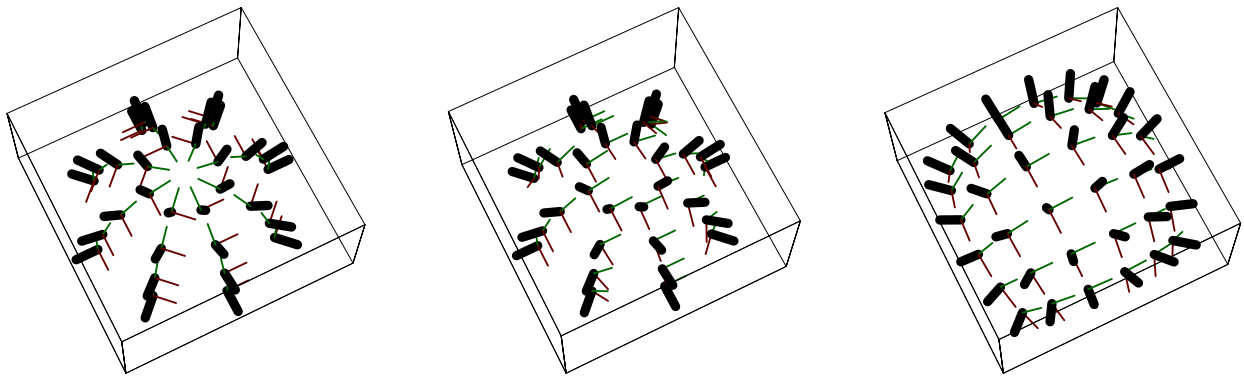
$$\frac{\partial}{\partial u}\begin{bmatrix}\hat{\mathbf{T}}_1(u,v)\\ \hat{\mathbf{T}}_2(u,v)\\ \hat{\mathbf{N}}(u,v)\end{bmatrix} = \begin{bmatrix}0 & +a_z(u,v) & -a_y(u,v)\\ -a_z(u,v) & 0 & +a_x(u,v)\\ +a_y(u,v) & -a_x(u,v) & 0\end{bmatrix}\begin{bmatrix}\hat{\mathbf{T}}_1(u,v)\\ \hat{\mathbf{T}}_2(u,v)\\ \hat{\mathbf{N}}(u,v)\end{bmatrix} \tag{24}$$

and

$$\frac{\partial}{\partial v}\begin{bmatrix}\hat{\mathbf{T}}_1(u,v)\\ \hat{\mathbf{T}}_2(u,v)\\ \hat{\mathbf{N}}(u,v)\end{bmatrix} = \begin{bmatrix}0 & +b_z(u,v) & -b_y(u,v)\\ -b_z(u,v) & 0 & +b_x(u,v)\\ +b_y(u,v) & -b_x(u,v) & 0\end{bmatrix}\begin{bmatrix}\hat{\mathbf{T}}_1(u,v)\\ \hat{\mathbf{T}}_2(u,v)\\ \hat{\mathbf{N}}(u,v)\end{bmatrix}\ . \tag{25}$$

The last lines of each of Eqs. (24) and (25) are typically combined in textbook treatments to give

$$\begin{bmatrix}\frac{\partial\hat{\mathbf{N}}(u,v)}{\partial u}\\ \frac{\partial\hat{\mathbf{N}}(u,v)}{\partial v}\end{bmatrix} = [\mathcal{K}]\begin{bmatrix}\hat{\mathbf{T}}_1(u,v)\\ \hat{\mathbf{T}}_2(u,v)\end{bmatrix}\ . \tag{26}$$



(a) (b) (c)

Figure 10: Examples of frame choices for the upper portion of an ordinary sphere. (a) Frames derived from standard polar coordinates on sphere. (b) Geodesic Reference frame for the sphere; each frame is as close as possible to the canonical coordinate axes at the North pole. (c) Frames derived from projective coordinates on the sphere, which turn out to be the same frame field as the Geodesic Reference frame.

where the matrix $[\mathcal{K}]$ has eigenvalues that are the principal curvatures k_1 and k_2, and thus

$$K = \det[\mathcal{K}] = k_1 k_2 \tag{27}$$

is the Gaussian curvature and

$$H = \frac{1}{2}\,\mathrm{tr}\,[\mathcal{K}] = \frac{1}{2}(k_1 + k_2) \tag{28}$$

is the mean curvature.

Examples of Surface Framings. If we are given a description of a surface, we can compute normals and choices of the corresponding frames by various means. In Figure 10, we illustrate three of these for the sphere. The first is derived from the standard orthonormal polar coordinate system, and the second is the extension to surfaces of the Geodesic Reference frame, which assigns the frame closest to a standard reference axis at the North Pole. The third is a frame based on polar projective coordinates for the sphere,

$$\begin{aligned} x(u,v) &= \frac{2u}{1+u^2+v^2} \\ y(u,v) &= \frac{2v}{1+u^2+v^2} \\ z(u,v) &= \frac{1-u^2-v^2}{1+u^2+v^2}\,, \end{aligned} \tag{29}$$

which map the real plane into the unit sphere with $x^2 + y^2 + z^2 = 1$ except for the point at infinity corresponding to the South pole. In fact, the polar projective coordinates generate the

same assignments as the Geodesic Reference frame does, so, except for the difference in locations of the grid sampling, these are the same framings.

Note: Do not be confused by alternate *samplings* of the same *framings*; if a parameterization $\mathbf{x}(u,v)$ gives a frame with $\mathbf{T}_1 = \partial\mathbf{x}(u,v)/\partial u$ and $\mathbf{T}_2 = \partial\mathbf{x}(u,v)/\partial v$, we can change to a polar sampled *mesh*, $(r = (u^2+v^2)^{1/2},\ \theta = \arctan(v,u))$, yet still retain the same frames at the same points $\mathbf{x}(r,\theta) = \mathbf{x}(u = r\cos\theta,\ v = r\sin\theta)$.

3 Quaternion Frames

In Section 2.1, we discussed the nature of 2D frames and noted a means of re-expressing the four equations with three constraints of the conventional frame system more efficiently; we showed a transformation into an equivalent set of two equations involving a single pair of variables obeying a unit length constraint and whose rotation transformation properties were realized by complex multiplication. Quaternions accomplish exactly this same transformation for 3D rotations: they permit the nine coupled frame equations with six orthonormality constraints in 3D to be succinctly summarized in terms of four quaternion equations with the single constraint of unit length. Detailed derivations along with other basic properties of quaternions are provided for reference in the appendix. A brief summary is given below.

Quaternion Frame Equations. Our task is now to rephrase the general properties of curve and surface frames in quaternion language so that, for example, we have a sensible space in which to consider optimizing frame assignments.

We begin with the standard definition for the correspondence between 3×3 matrices $R^i{}_j$ and quaternions q:

$$R_q(\mathbf{V})^i = \sum_j R^i{}_j V^j = q * (0, V^i) * q^{-1}\ . \tag{30}$$

Henceforth, we will use the notation "$*$" to distinguish quaternion multiplication, and will use "$\cdot$" when necessary to denote ordinary Euclidean inner products. Next, we express each orthonormal frame component as a column of $R^i{}_j$ by using an arbitrary quaternion to rotate each of the three Cartesian reference axes to a new, arbitrary, orientation:

$$\begin{aligned} \hat{\mathbf{N}}_1 \text{ or } \hat{\mathbf{T}}_1 &= q * (0, \hat{\mathbf{x}}) * q^{-1} \\ \hat{\mathbf{N}}_2 \text{ or } \hat{\mathbf{T}}_2 &= q * (0, \hat{\mathbf{y}}) * q^{-1} \\ \hat{\mathbf{T}} \text{ or } \hat{\mathbf{N}} &= q * (0, \hat{\mathbf{z}}) * q^{-1}\ . \end{aligned} \tag{31}$$

(Technically speaking, in the above equation $\hat{\mathbf{T}}$ really means the quaternion $(0,\ \hat{\mathbf{T}})$ with only a vector part, etc.) All this can be transformed into the following explicit representation of the frame vectors as columns of a matrix of quaternion quadratic forms:

$$\begin{bmatrix} [\hat{\mathbf{N}}_1] & [\hat{\mathbf{N}}_2] & [\hat{\mathbf{T}}] \end{bmatrix} =$$
$$\begin{bmatrix} [\hat{\mathbf{T}}_1] & [\hat{\mathbf{T}}_2] & [\hat{\mathbf{N}}] \end{bmatrix} =$$

$$\begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2q_1q_2-2q_0q_3 & 2q_1q_3+2q_0q_2 \\ 2q_1q_2+2q_0q_3 & q_0^2-q_1^2+q_2^2-q_3^2 & 2q_2q_3-2q_0q_1 \\ 2q_1q_3-2q_0q_2 & 2q_2q_3+2q_0q_1 & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix} . \tag{32}$$

Note: MATHEMATICA users should remind themselves that matrices are stored as lists of rows in MATHEMATICA, so one must *transpose* a standard matrix to easily retrieve column vectors from Eq. (32) and avoid mysterious sign errors.

Taking differentials of Eq. (31), we generate expressions of the form

$$dq = q * (q^{-1} * dq) = q * \frac{1}{2}(0, \mathbf{k}) \tag{33}$$

$$\begin{aligned} dq^{-1} &= (dq^{-1} * q) * q^{-1} \\ &= -(q^{-1} * dq) * q^{-1} \\ &= -\frac{1}{2}(0, \mathbf{k}) * q^{-1} \end{aligned} \tag{34}$$

where

$$\mathbf{k} = 2(q_0\,\mathbf{dq} - \mathbf{q}\,dq_0 - \mathbf{q} \times \mathbf{dq}) .$$

Substituting these expressions into the the calculation for the first column, we immediately find the expected commutators of quaternion multiplication:

$$\begin{aligned} d\hat{\mathbf{N}}_1 &= dq * (0, \hat{\mathbf{x}}) * q^{-1} + q * (0, \hat{\mathbf{x}}) * dq^{-1} \\ &= \frac{1}{2}\, q * ((0, \mathbf{k}) * (0, \hat{\mathbf{x}}) - (0, \hat{\mathbf{x}}) * (0, \mathbf{k})) * q^{-1} \\ &= q * (0, \mathbf{k} \times \hat{\mathbf{x}}) * q^{-1} . \end{aligned}$$

The rest of the columns are computed similarly, and a straightforward expansion of the components of the cross products proves the correspondence between Eq. (33) and Eq. (9).

To relate the derivative to a specific curve coordinate system, for example, we would introduce the curve velocity normalization $v(t) = \|\mathbf{x}'(t)\|$ and write

$$q' = v(t)\frac{1}{2}\, q * (0, \mathbf{k}) . \tag{35}$$

One of our favorite ways of rewriting this equation follows directly from the full form for the quaternion multiplication rule given in the appendix; since this multiplication can be written as an orthogonal matrix multiplication on the 4D quaternion space, we could equally well write

$$\begin{bmatrix} q_0' \\ q_1' \\ q_2' \\ q_3' \end{bmatrix} = v(t)\frac{1}{2} \begin{bmatrix} 0 & -k_x & -k_y & -k_z \\ +k_x & 0 & +k_z & -k_y \\ +k_y & -k_z & 0 & +k_x \\ +k_z & +k_y & -k_x & 0 \end{bmatrix} \cdot \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} . \tag{36}$$

This is the 3D analog of Eq. (7).

At this point, there are many other directions we could carry this basic structure, but we will not pursue the general theory of quaternion differential geometry further here. We will conclude with a short summary of the quaternion treatment of the classical surface equations. Starting from Eq. (33), we are led immediately to the quaternion analogs of Eqs. (24) and (25):

$$q_u \equiv \partial q/\partial u = \frac{1}{2}\, q * (0, \mathbf{a}) \tag{37}$$

$$q_v \equiv \partial q/\partial v = \frac{1}{2}\, q * (0, \mathbf{b})\ . \tag{38}$$

But how shall we express the curvatures in a way similar to the classical formula in Eq. (26)? An elegant form follows by pursuing the quaternion analog of the vector field equations given in Eqs. (22,23). We write

$$\begin{aligned} q_u * q_v^{-1} &= -\frac{1}{4}\, q * (0, \mathbf{a}) * (0, \mathbf{b}) * q^{-1} \\ &= -\frac{1}{4}\, q * (-\mathbf{a}\cdot\mathbf{b},\ \mathbf{a}\times\mathbf{b}) * q^{-1} \\ &= -\frac{1}{4}\, \left[-\mathbf{a}\cdot\mathbf{b}\,\hat{\mathbf{I}} + (\mathbf{a}\times\mathbf{b})_x\,\hat{\mathbf{T}}_1 + (\mathbf{a}\times\mathbf{b})_y\,\hat{\mathbf{T}}_2 + (\mathbf{a}\times\mathbf{b})_z\,\hat{\mathbf{N}}\right]\ , \end{aligned} \tag{39}$$

where we use the quaternion forms in Eq. (31) with the addition of the quaternion identity element $\hat{\mathbf{I}} = (1,\, \mathbf{0}) = q*(1,\, \mathbf{0})*q^{-1}$ for the frame vectors. We see that the projection to the normal direction gives precisely the determinant $(\mathbf{a}\times\mathbf{b})_z = K$ identified in Eq. (26) as the scalar curvature. The mean curvature follows from an expression similar to Eq. (23),

$$\begin{aligned} q * (0, \hat{\mathbf{x}}) * q_u^{-1} + q * (0, \hat{\mathbf{y}}) * q_v^{-1} &= -\frac{1}{2} q * (-\hat{\mathbf{x}}\cdot\mathbf{a} - \hat{\mathbf{y}}\cdot\mathbf{b},\ \hat{\mathbf{x}}\times\mathbf{a} + \hat{\mathbf{y}}\times\mathbf{b}) * q^{-1} \\ &= -\frac{1}{2}\left[-(a_x + b_y)\hat{\mathbf{I}} + b_z\hat{\mathbf{T}}_1 - a_z\hat{\mathbf{T}}_2 + (a_y - b_x)\hat{\mathbf{N}}\right] \end{aligned} \tag{40}$$

where again the coefficient of the normal, $(a_y - b_x) = \mathrm{tr}\,[\mathcal{K}] = 2H$, is the desired expression. Similar equations can be phrased directly in the 4D quaternion manifold using the forms of Eq. (36).

3.1 Visualizing Quaternion Frames

Seeing the parameters of a single quaternion. Any (unit) quaternion is a point on S^3 and therefore is described by three parameters incorporated in the standard parameterization

$$q(\theta, \hat{\mathbf{n}}) = (\cos\frac{\theta}{2}, \hat{\mathbf{n}}\sin\frac{\theta}{2})\ , \tag{41}$$

where $0 \le \theta < 4\pi$, and the eigenvector of the rotation matrix (unchanged by the rotation), is a point on the two-sphere S^2 representable as $\hat{\mathbf{n}} = (\cos\alpha\cos\beta,\ \sin\alpha\cos\beta, \sin\beta)$ with $0 \le \alpha < 2\pi$ and $-\pi/2 \le \beta \le \pi/2$.

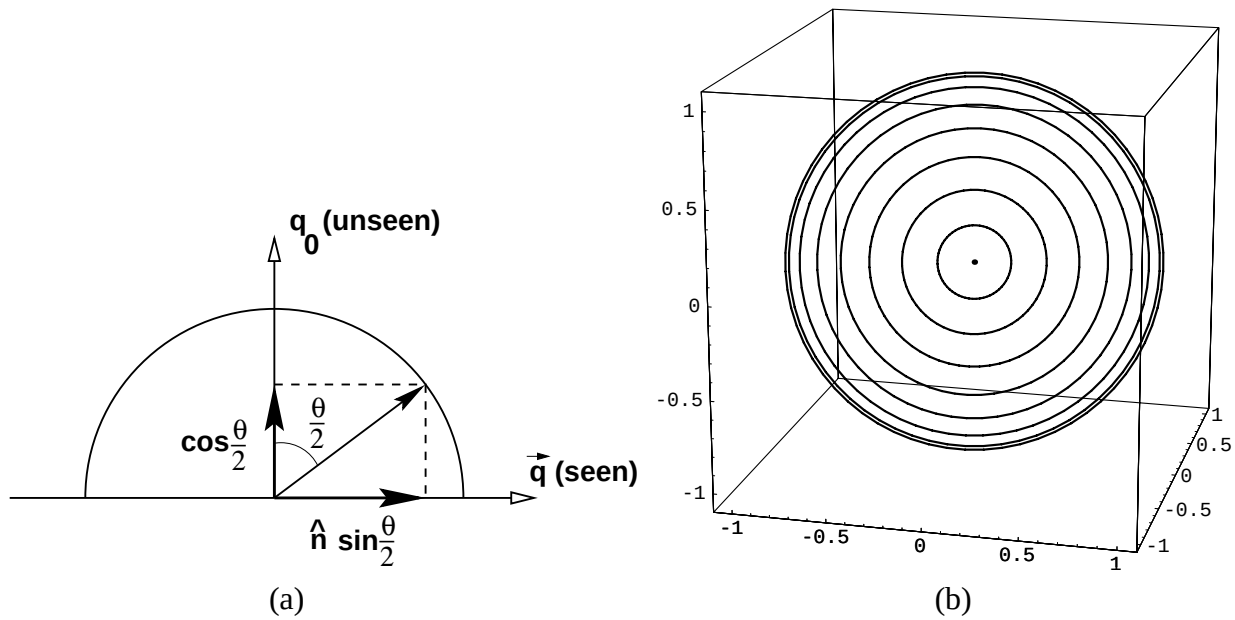


Figure 11: Illustration of how the q_0 part of a quaternion is "known" if we have a 3D image of the vector part $\mathbf{q} = \hat{\mathbf{n}} \sin \frac{\theta}{2}$ of the quaternion. (b) Schematic representation of the concentric-sphere uniform distance scales needed to form a mental model of the metric distances in quaternion space between two points in the parallel 3D projection. Distances are roughly Euclidean near the origin ($\mathbf{q} \approx 0$ in (a)) and equal-length lines appear increasingly compressed as the radius approaches unity.

An informative visualization of quaternions can be constructed by examining their properties carefully. If we simply make a 3D display of the vector part of the quaternion, $\hat{\mathbf{n}} \sin \frac{\theta}{2}$, we see that the scalar element of the quaternion is redundant, since, for each θ,

$$q_0 = \cos \frac{\theta}{2} = \pm \left(1 - \left| \hat{\mathbf{n}} \sin \frac{\theta}{2} \right|^2 \right)^{1/2} . \tag{42}$$

That is, q_0 is just the implicitly known height of the 4D unit vector in the unseen projection direction, as illustrated in Figure 11(a). In Figure 11(b), we schematize the mental model of metric distance required to complete the interpretation of the visualization. If we imagine dividing the arc of the semi-circle in Figure 11(a) into equal angular segments, the arc lengths are all the same distance apart in spherical coordinates. Projected onto the $\mathbf{q}$ plane, however, the projected spacing is non-uniformly scaled by a a factor of $\sin \theta$. Thus to keep our vision of distance consistent, we imagine the space to be like 3D graph paper with concentric spheres drawn at equal distances in the special scale space; such 3D graph paper would look like Figure 11(b). Distances are essentially Euclidean near the 3D origin, for small 3D radii, and are magnified as the radius approaches unity to make the marked spheres equidistant in conceptual space.

If we assume the positive root is always taken for q_0, then we effectively restrict ourselves to a single hemisphere of S^3 and eliminate the two-fold redundancy in the correspondence between

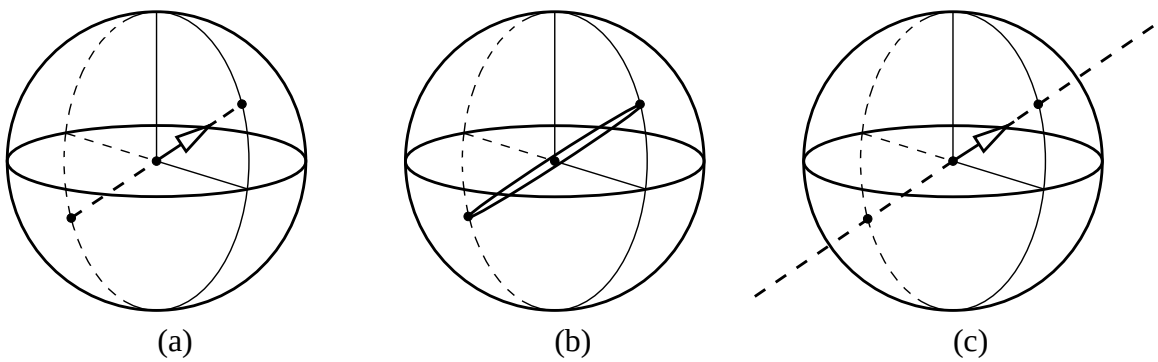


Figure 12: (a) This image represents the 3D vector part of the quaternion $q = (\cos\frac{\theta}{2}, \hat{\mathbf{t}}\sin\frac{\theta}{2})$ representing a single instance of the one-parameter family of possible rotations leaving invariant the tangent vector $\hat{\mathbf{t}}$ at one point of a space curve. (b) A representation of the entire one-parameter space of possible frames having the same tangent vector $\hat{\mathbf{t}}$. The vector part of the quaternion must lie on the diameter of the two-sphere in 3D depicted here. We depict the diameter as a very skinny ellipse, because it is in fact a degenerate projection of a circle in 4D, which could be exposed as shown by making a small 4D rotation before projecting to 3D. (c) A polar projection of the same object removes the doubling by projecting the circle to a line through infinity in R^3.

quaternions and the rotation group. Alternatively, despite the fact that quaternions with both signs of q_0 map to the same point in this projection, we can indicate the simultaneous presence of both hemispheres using graphical cues; one possible method is to use saturated colors in the "front" hemisphere, and faded colors (suggesting distance) for objects in the "back" hemisphere.

Hemispheres in S^3**.** To clarify the terminology, we note that a projected hemisphere for S^2 is a filled disk (a "two-ball") in the plane, and the full surface of the sphere consists of two such disks joined at the outer circular boundary curve; for S^3, we use the word hemisphere to indicate a filled solid two-sphere (technically a "three-ball"), and imagine the full volume of the three-sphere to consist of *two* such spherical solids joined on the skin (a two-sphere) of the surface enclosing both.

The family of possible values of Eq. (41) projects to a double-valued line (actually an "edge-on" projection of a circle) which is a directed diameter of the unit two-sphere, in the direction of $\hat{\mathbf{n}}$; in a polar projection, this circle becomes a line to infinity through the origin. These representations of a unit quaternion as a vector from the origin to a point inside the solid two-sphere (the three-ball) are illustrated schematically in Figure 12.

Any particular 3D rotation is represented twice, since the quaternion circle is parameterized by $0 \leq \theta < 4\pi$. A simple parallel projection thus produces two solid balls on top of each other in the 3D projection, one the analog of the "North pole disk" of a two-sphere parallel projected from 3D to a screen, the other the analog of the "South pole disk" of a two-sphere. The analog of a polar projection, which for a two-sphere sends the North pole to infinity of R^2, flattens the three-sphere out to fill R^3, as shown in Figure 12(c), and eliminates the double-valued properties of the parallel projection.

CURVE LENGTHS	(2,3) Torus Knot	Helix
Frenet Frame	14.3168	6.18501
Geodesic Reference Frame	14.6468	7.82897
Parallel Transport Frame	10.1865	6.06301

Table 1: Relative lengths (in radians) of the quaternion frame maps for various frame choices describing the (2,3) torus knot and the helix. The Parallel Transport frame is the shortest possible frame map.

3.2 Quaternion Frames for Curves

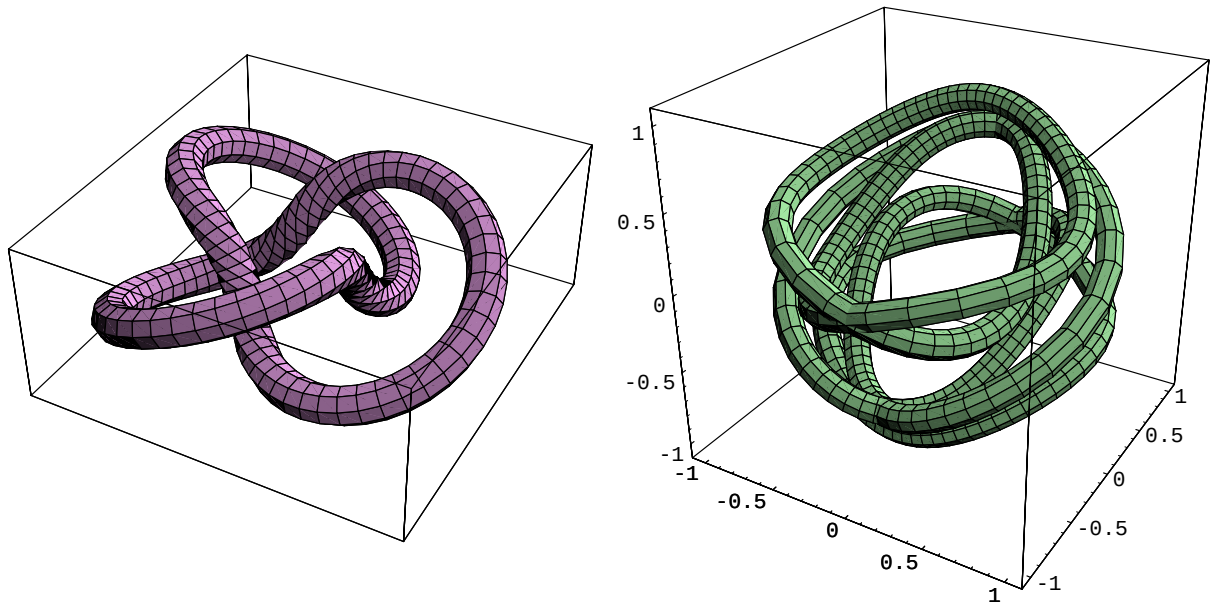
We now can produce quaternion frames for space curves directly by several techniques.

Quaternions from Local 3D Frames. In the case of the Frenet frame, we have no choice but to consider each frame as totally independent of the others. Each is locally computable, and there is in principle no relation between them, since the curvature could vanish at any point. In this case, we compute the frames directly from Eq. (16), thus deriving a 3×3 orthogonal matrix $R(t)$ at each point of the curve. We then apply standard inversion algorithms [37, 35, 31] to obtain the corresponding quaternion up to a sign. Finally, we apply a simple operator that checks the local continuity of the corresponding frames. If two quaternion vectors representing neighboring frames have a dot product near negative one, we change the sign of one to keep it near its neighbors. If two neighbors are excessively far apart in terms of the 4D angle between them, and are not simply near-negatives of one another, then the Frenet frame probably is poorly defined and should be tagged as such until continuity resumes. Figure 13 shows the Frenet frame tubing of a torus knot and the corresponding trajectory of these frames in the vector subspace of quaternion space.

Note: Forcing close quaternion Frenet frames on closed curves such as torus knots results in a very interesting phenomenon. Depending on the parameters of the curve, the path in quaternion space may close after a single traversal of the curve, or it may require two or more traversals, as in the case shown in Figure 13. We have checked this feature on a wide range of torus knots, and found that there are generally "jumps" between needing different numbers of circuits at those parameter values that imply inflection points (zero curvature) in the curve.

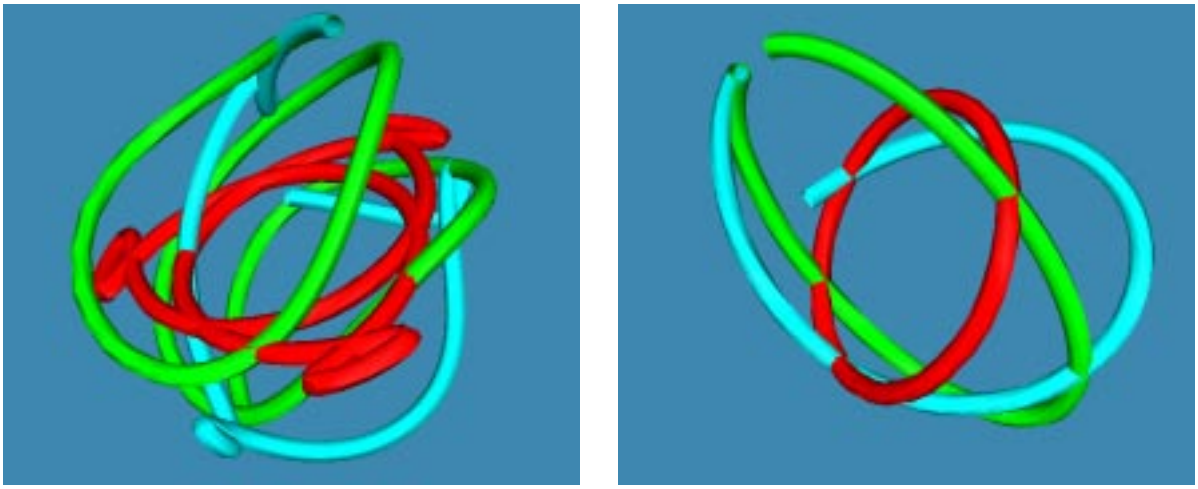
Direct Quaternion Frames. The Geodesic Reference frame and the Parallel Transport frame, in contrast to the Frenet frame, can be defined directly in terms of quaternions if desired, as indicated in Section 2.2; all that is needed is an initial quaternion reference frame, and then the geometry of the curve specifies enough at each point to express the needed rotation in quaternion form.

Comparison of Tubings and Quaternion Frames. Previously, in Figure 8, we compared the tubings for the (2,3) torus knot and for the helix based on the Frenet, Geodesic Reference, and Parallel Transport frames. The corresponding quaternion paths are illustrated together in Figure 14. The Parallel Transport frame shown uses the initial Frenet frame as a starting point; we could



(a) (b)

Figure 13: (a) A trefoil torus knot. (b) Its quaternion Frenet frame projected to 3D. For this trefoil knot, the frame does not close on itself in quaternion space unless the curve is traversed twice, corresponding to the double-valued “mirror” image of the rotation space that can occur in the quaternion representation. Observe the longer segments in (b): these correspond to the three high-torsion segments observable in (a).



(a) (b)

Figure 14: (a) Quaternion frames in “standard” 3D vector visualization projection for the (2,3) torus knot: Red—Geodesic Reference: this is planar by construction, since all 3D points must lie in the plane perpendicular to the reference axis; the 3D origin is at the centroid of the red curve. Green—Frenet: the Frenet frame is actually cyclic, but to see this easily for this 2,3 torus knot, the mirror image of the current frame must be added, giving effectively a double traversal of the curve as shown in Figure 13. Cyan—Parallel Transport: the PT frame must be given a starting value, which here is seen at the top center of the image to coincide with the (green) Frenet frame. The PT frame is not cyclic, but is the shortest path, with three very noticeable tight loops. (b) The same selection of quaternion frames for the helix. Again, the red Geodesic Reference curve is planar (and cycles back on itself twice for this helix); the green Frenet frame takes a longer path that will return to its original orientation, and the cyan Parallel Transport frame, seen starting at the same orientation as the Frenet, will not ordinarily return to the same orientation, but will have the shortest 4D path length. (The hidden double circuit of the Geodesic Reference frame for this helix in fact makes it longer.)

use any starting quaternion with the correct tangent vector. The relative path lengths of the curves in Figure 14 are summarized in Table 1.

We note the following properties:

- **Frenet.** Periodic for periodic non-singular curves, has a tendency to twist a bit too much (where the torsion is high), leaving long jumps between neighboring samples in quaternion space; undefined at inflection points and zero curvature segments.

- **Geodesic Reference.** Also guaranteed to be periodic for periodic curves, but has the odd property that it always lies in a plane perpendicular to the reference axis in our preferred 3D quaternion projection. Ambiguous and therefore potentially not smooth for frames opposing the reference frame direction.

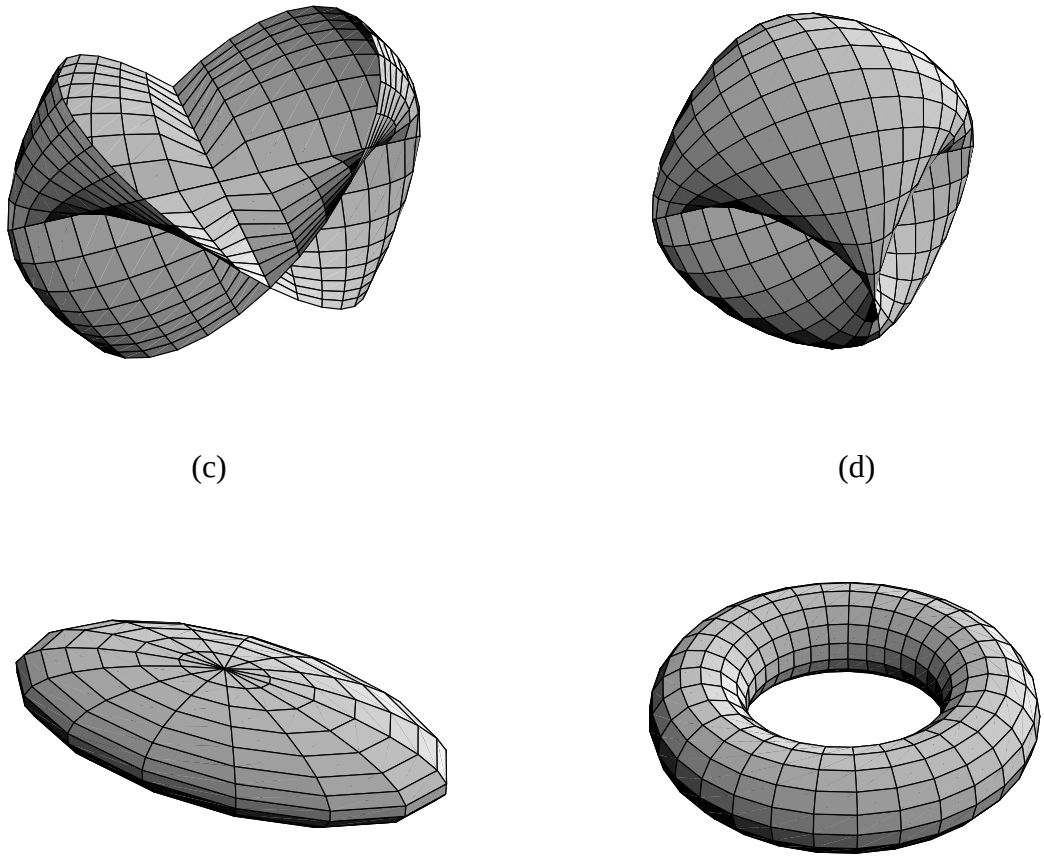
- **Parallel Transport.** This is the quaternion frame with minimal 4D length, though it may be difficult to see this feature immediately in our standard projection. It is not in general a periodic path. Different choices of starting frame produce curves of identical length differing by rigid (possibly reflecting) 4D motions (see Eq. (56)).

3.3 Quaternion Gauss Map for Surfaces

The quaternion Gauss map extends the Gauss map to include a representation of the entire coordinate frame at each surface point, introducing a number of new issues. In particular there is a useful, but mathematically suspect, approach that we might call an "engineering" approach to the quaternion Gauss map that lets us quickly get informative visualizations for those special cases where we are given a locally orthogonal parameterization of the surface except perhaps for isolated singularities of the coordinate system.

For these cases, we may construct the precise quaternion analog of the Gauss map by lifting the surface's coordinate mesh into the space of quaternions at each value of the orthonormal coordinatization (u, v) of the surface or surface patch. The correspondence of this map to the Gauss map is *not* directly visible, since (see Eq. (32)) the normal directions of the Gauss map are nontrivial quadratic forms constructed from all the quaternion components; however, a projection to a subspace of the quaternion space based on the bilinear action of quaternions on pure vectors may be constructed by imitating the projection of the Hopf fibration of S^3 (see, e.g., Shoemake [39, 3]). In Figure 15, we show two such cases, an ellipsoid with orthonormal polar coordinates singular at the poles and a torus with global, nonsingular, coordinates, using our now-standard projections of the quaternion Gauss map to 3D. In each of these cases, a single circuit of the surface generates only one-half of the quaternion surface shown; the symmetric quaternion figure results from traversing the surface twice to adjoin the reflected image of the single-circuit quaternion surface. That is, each point on the 3D surfaces appears twice, once at q, and once at $-q$, in these periodic quaternion Gauss maps.

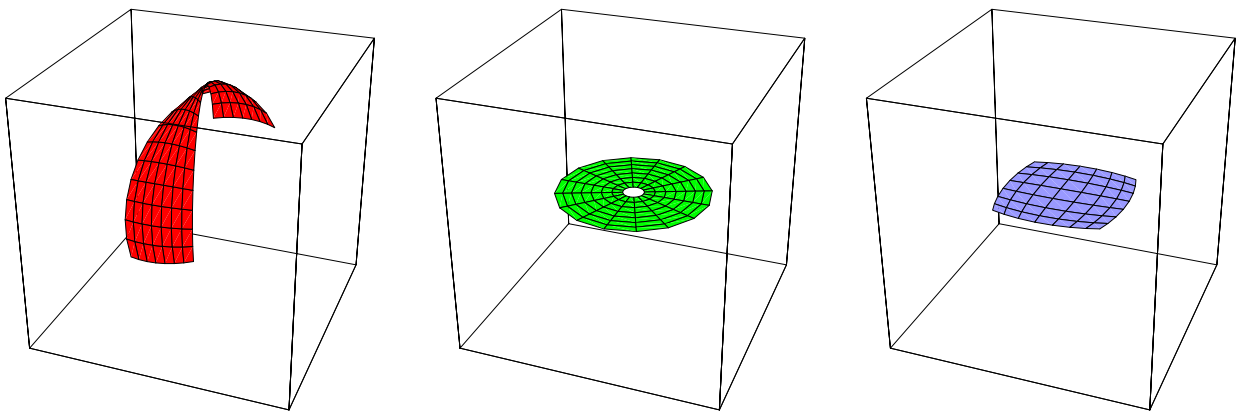
We see that the singular coordinate system typically used for the ellipsoid is topologically a cylinder; the circles corresponding to the singularities of the coordinate system (circles of normal directions) at the North and South poles correspond to *boundaries* of the quaternion Gauss map.



(c) (d)

(a) (b)

Figure 15: Examples of quaternion Gauss maps for surfaces. (a) The ellipsoid and (b) the torus. (c,d) The corresponding Quaternion Gauss maps, projected from the three-sphere in 4D. The equatorial direction has been traversed twice in order to get a closed path in the map; the singular poles in the ellipsoid coordinate system correspond to the edges or boundaries of the quaternion-space ribbon.



(a) (b) (c)

Figure 16: Examples of quaternion Gauss maps for the frame choices for the upper portion of an ordinary sphere given originally in Figure 10. (a) Frames derived from standard polar coordinates on sphere. (b) Geodesic reference frame for the sphere; each frame is as close as possible to the canonical coordinate axes at the North pole. (c) Frames derived from projective coordinates on the sphere.

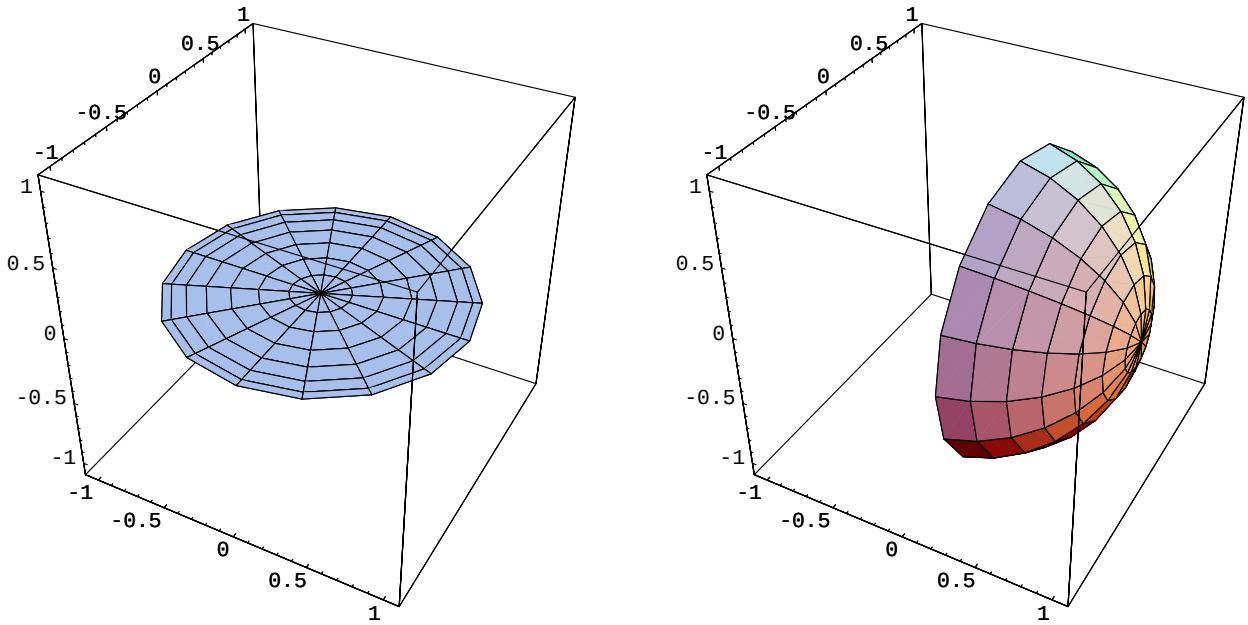
PATCH AREAS	Hemispherical patch
Polar Coordinates	2.1546
Geodesic Reference Frame	1.9548

Table 2: Areas (in steradians) of the quaternion frame maps for the polar coordinate and Geodesic Reference frame choices on the hemispherical patches of Figure 16.

The torus, which has the extremely unusual feature that it possesses a global regular coordinate system, has a (reflection doubled) quaternion Gauss map which is another, four-dimensional, torus embedded in the quaternion S^3 space.

Quaternion Maps of Alternative Sphere Frames. In Figure 10, we showed three alternate sets of frames for the upper half of an ordinary sphere. The assigned coordinate systems may be converted directly into quaternion frames and coerced into consistency in the usual manner. In Figure 16, we show the results. The Geodesic Reference frames and the projective coordinates are in fact the same space of frames computed in different ways: both are planes perpendicular to the $\hat{\mathbf{z}}$ axis. The coordinate systems used to compute the quaternion Gauss maps in parts (a) and (b) of the figure are commensurate, so we may compare the areas, computed using solid angle on the three-sphere in units of steradians; the results are shown in Table 2.

Covering the Sphere and the Geodesic Reference Frame South Pole Singularity. The Geodesic Reference frame for a surface patch has the peculiarity that it has an ambiguity whenever the vector to be assigned is exactly opposite the reference frame. As we show in Figure 17, the tilting from the



(a) (b)

Figure 17: The Geodesic reference frame tilts to an ambiguous result as the tilt angle approaches π, the inverted direction of the chosen reference frame. We see two different 3D projections of the quaternion surface, (a) giving the vector coordinates (q_1, q_2, q_3), and (b) the coordinates (q_0, q_1, q_2). The center is the North pole, the middle ring is the equator, and outer circle is in fact the space of possible frames at the South pole of the sphere: there is no unique way to tilt the North pole to the orientation of the South pole, as there is a full circle of arbitrariness in the choice.

reference frame in quaternion space (easily seen in ordinary 3D space as well) eventually reaches a quaternion circle representing the ambiguous orientation of the frame with reference direction along the $-\hat{\mathbf{z}}$ axis. This phenomenon is a practical consequence of the fact that the two-sphere does not admit a global vector field: according to classical manifold theory (see, e.g., Milnor or Grimm and Hughes [29, 13]), one needs at least two separate patches, one for the North pole and one for the South pole, to place a complete set of coordinates (or equivalently, for our problem, a set of frames) on a sphere.

The more mathematical approach requires that interesting surfaces be defined as a collection of patches [13, 7], and the spaces of frames for each patch must be matched up and sewn together by assigning a transition function along the boundaries. There are a variety of ways one can approach the problem of taking a manifold and associating fiber bundles with it; the most relevant fiber bundle for the context of the current problem is the *space of moving frames* of the space R^3 in which the surface is embedded [7, 40]. We in fact move as usual from the space of frames to the space of associated quaternions. Then at each point $\mathbf{x}$ of a patch we have frames that are matrix-valued functions from the patch into the group SU(2) of quaternion frames (which we treat as the topological space S^3). We can express the relationship between the frames q and q' of two neighboring patches U and U', represented as quaternions, via quaternion multiplication by a transition function t:

$$q' = t * q \ .$$

We may in fact explicitly construct the transition functions between the two patches as quaternion maps, giving a quaternion version of one of the classical procedures of manifold theory. In Figure 18(a), we show the the projective coordinates on the sphere that produced the set of coordinate frames in Figure 10(c), which are essentially equivalent to those in (b) sampled at polar coordinate values. Using the polar coordinate sampling, so that we can easily identify the equator, we show in Figure 18(b) the quaternion maps corresponding to the coordinate frames derived from this orthonormal coordinate system covering the North pole (disk in center) and the South pole (smashed side view of a hemisphere in the 3D projection with its $q \rightarrow (-q)$ partner). These coordinate systems agree at exactly *one* point on the equator, which is (almost) evident from the figure; note that we have chosen to display the coordinate systems only up to the equator, unlike the patches of Figure 17, which cover the entire sphere except for one pole.

In order to establish a mapping covering the complete sphere, we must write down an explicit correspondence between the quaternion frames for each patch at each shared point on the equator. In Figure 19(a), we show the geodesic arcs on S^3 symbolizing the transition rotation

$$t(\theta) = q_{\text{south}}(\theta) * q^{-1}_{\text{north}}(\theta)$$

at each point on the equatorial circle parameterized by θ. Note carefully the order of quaternion multiplication; with our conventions a different order will not work. The arcs themselves are actually segments of the space of possible frames, since the simplest rotation between two frames with the same normal (at the same point on the equator) is a geodesic rotation about that normal. Figure 19(b) and (c) shows the transition functions $q(\theta)$ sampled at regular intervals in θ and referred to the origin $(1, 0, 0, 0)$ in quaternion space. Each quaternion point at the end of an arc

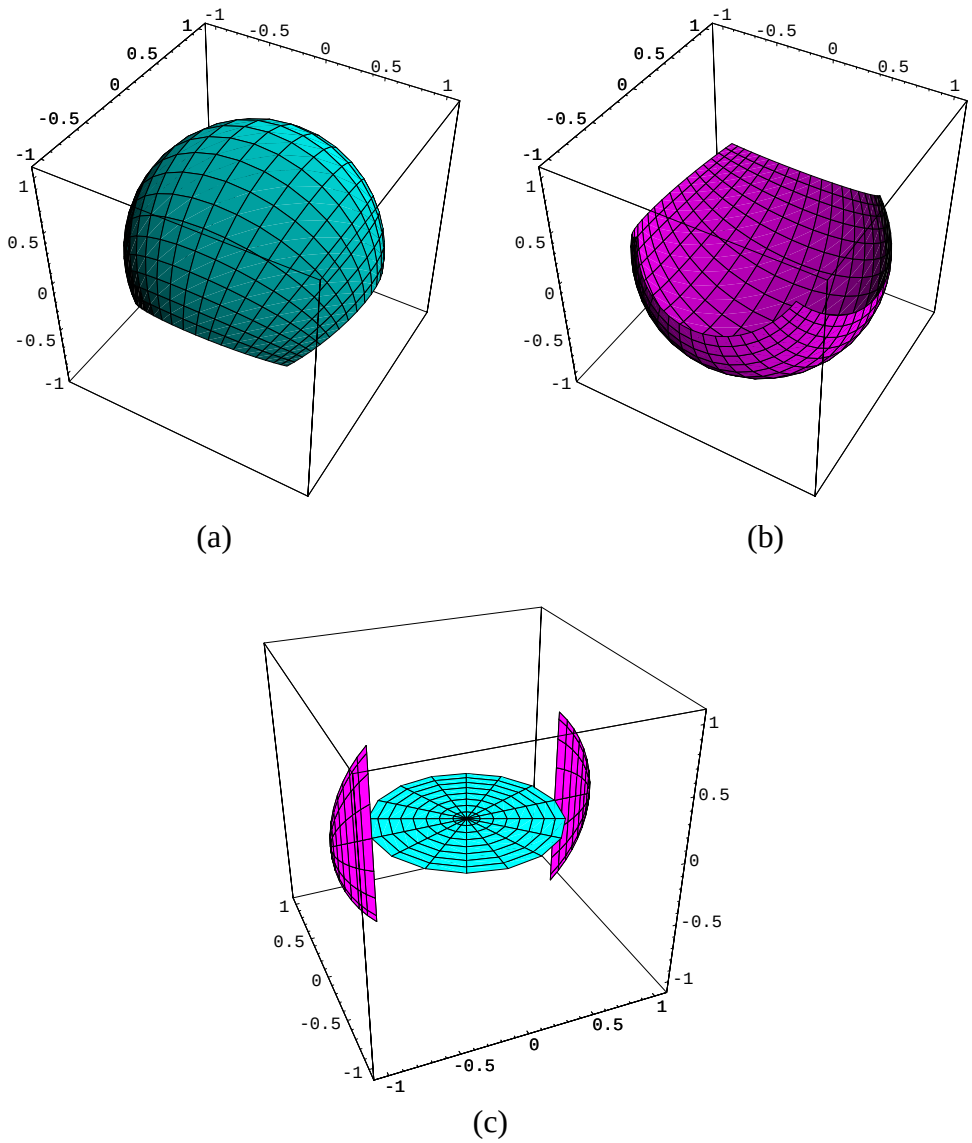
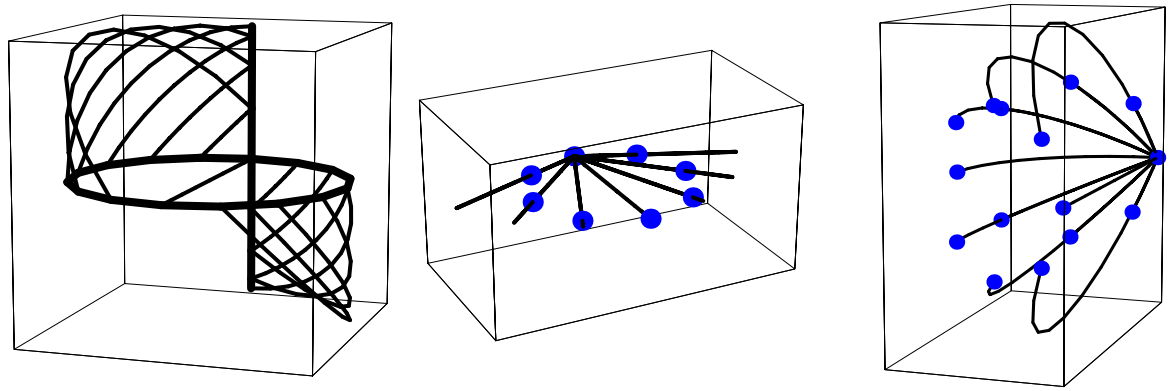


Figure 18: (a) The North pole projective coordinatization of the sphere; (b) a similar regular patch for the South pole. Because of the "no-hair" theorem, no single regular patch can cover the entire sphere. (c) The quaternion mappings of the systems of frames given by the North and South pole coordinate patches, sampled in polar coordinates. The $q \to (-q)$ reflected images are included, though the North pole's images both have the same projection and are thus indistinguishable here. The maps in (c) extend only to the equator, unlike the patches given in Figure 17.

represents a rotation to be applied to a point on the North pole patch equator to obtain the coordinate frame at the corresponding point on the South pole patch equator. One point is in fact the identity, and there is some degeneracy due to reflection symmetry across the equator.



(a) (b) (c)

Figure 19: (a) The transition functions from the North pole frame to the South pole frame as arcs in the three-sphere. These arcs are pieces of the space of possible frames with a given normal on the equatorial point. (b) A representation of the transition functions as arcs from the origin in rotation space (the pole $(1, 0, 0, 0)$ in quaternion space) common to all the arcs here. The ends of the arcs thus represent the actual rotation needed to match the coordinate systems at each point on the equator. (c) A different projection from 4D to 3D, showing more details of the structure of the transition function arcs, which have a two-fold degeneracy in the standard projection (b).

4 The Space of Frames

We at last ready to introduce the key concept of the *space of possible frames*.

Suppose at each sample point $\mathbf{x}(t)$ of a curve, we are given a unit tangent vector, $\hat{\mathbf{T}}(t)$, computed by whatever method one likes (two-point sampling, five-point sampling, analytic, etc.). Then one can immediately write down a one-parameter family describing all possible choices of the normal plane orientation: it is just the set of rotation matrices $R(\theta, \hat{\mathbf{T}}(t))$ (or quaternions $q(\theta, \hat{\mathbf{T}}(t))$) that leave $\hat{\mathbf{T}}(t)$ *fixed*.

For surfaces, the analogous construction follows from determining the unit normal $\hat{\mathbf{N}}(u, v)$ at each point $\mathbf{x}(u, v)$ on the surface patch. The needed family of rotations $R(\theta, \hat{\mathbf{N}}(u, v))$ (or quaternions $q(\theta, \hat{\mathbf{N}}(u, v))$) now leaves $\hat{\mathbf{N}}(u, v)$ fixed and parameterizes the space of possible *tangent* directions completing a frame definition at each point $\mathbf{x}(u, v)$.

However, there is one slight complication: the family of frames $R(\theta, \hat{\mathbf{v}})$ leaving $\hat{\mathbf{v}}$ fixed does not have $\hat{\mathbf{v}}$ as one column of the 3×3 rotation matrix, and so does not actually describe the desired family of frames. Therefore we proceed as follows:

We define $f(\theta, \hat{\mathbf{v}}) = (f_0, f_1, f_2, f_3)$ to be a quaternion describing the family of frames for which the direction $\hat{\mathbf{v}}$ is a preferred fixed axis of the frame, such as the tangent or normal vectors. The orthonormal triad of 3-vectors describing the desired frame is

$$F(\theta, \hat{\mathbf{v}}) = \begin{bmatrix} f_0^2 + f_1^2 - f_2^2 - f_3^2 & 2f_1 f_2 - 2f_0 f_3 & 2f_1 f_3 + 2f_0 f_2 \\ 2f_1 f_2 + 2f_0 f_3 & f_0^2 - f_1^2 + f_2^2 - f_3^2 & 2f_2 f_3 - 2f_0 f_1 \\ 2f_1 f_3 - 2f_0 f_2 & 2f_2 f_3 + 2f_0 f_1 & f_0^2 - f_1^2 - f_2^2 + f_3^2 \end{bmatrix}, \tag{43}$$

where one column, typically the 3rd column, must be $\hat{\mathbf{v}}$.

The standard rotation matrix $R(\theta, \hat{\mathbf{v}})$ leaves $\hat{\mathbf{v}}$ fixed but does not have $\hat{\mathbf{v}}$ as one column of the 3×3 rotation matrix, and so we have more work to do. To compute $f(\theta, \hat{\mathbf{v}})$, we need the following:

- A base reference frame $b(\hat{\mathbf{v}})$ that is guaranteed to have, say, the 3rd column exactly aligned with a chosen vector $\hat{\mathbf{v}}$, which is either the tangent to a curve or the normal to a surface.
- A one-parameter family of rotations that leaves a fixed direction $\hat{\mathbf{v}}$ invariant.

The latter family of rotations is given simply by the standard quaternion

$$q(\theta, \hat{\mathbf{v}}) = (\cos\frac{\theta}{2}, \hat{\mathbf{v}} \sin\frac{\theta}{2}) \ , \tag{44}$$

for $0 \leq \theta < 4\pi$, while the base frame can be chosen as

$$b(\hat{\mathbf{v}}) = q(\arccos(\hat{\mathbf{z}} \cdot \hat{\mathbf{v}}), \ (\hat{\mathbf{z}} \times \hat{\mathbf{v}})/\|\hat{\mathbf{z}} \times \hat{\mathbf{v}}\|) \ . \tag{45}$$

We refer hereafter to the frame $b(\hat{\mathbf{v}})$ as the *Geodesic Reference Frame* because it tilts the reference vector $\hat{\mathbf{z}}$ along a geodesic arc until it is aligned with $\hat{\mathbf{v}}$; see Figure 20. If $\hat{\mathbf{v}} = \hat{\mathbf{z}}$, there is no problem, since we just take $b(\hat{\mathbf{v}})$ to be the quaternion $(1, \mathbf{0})$; if $\hat{\mathbf{v}} = -\hat{\mathbf{z}}$, we may choose any

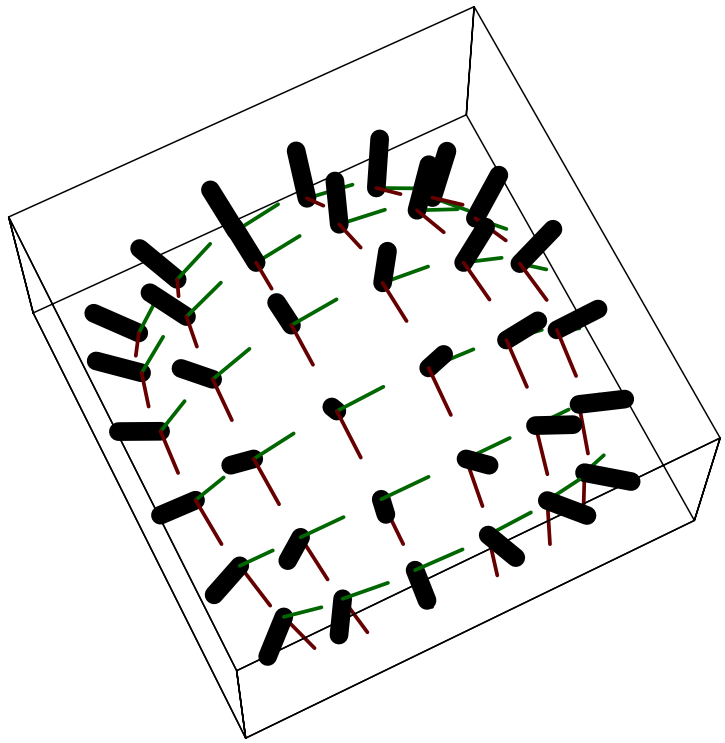


Figure 20: Example of the Geodesic Reference Frame: on the northern hemisphere of a 2-sphere, the Geodesic Reference Frame tilts the $\hat{\mathbf{z}}$ axis of the north pole's identity frame along the shortest arc to align with a specified reference direction.

compatible quaternion such as $(0, 1, 0, 0)$. We escape the classic difficulty of being unable to assign a global frame to all of S^2 because we need a parameterization of *all possible* frames, not any one particular global frame. If one wants to use a reference frame that is not the identity frame, one must premultiply $b(\hat{\mathbf{v}})$ on the right by a quaternion rotating from the identity into that reference frame; this is important when constructing a nonstandard Geodesic Reference Frame such as that required to smoothly describe a neighborhood of the southern hemisphere of S^2.

We can thus write the full family of possible quaternion frames keeping $\hat{\mathbf{v}}$ as a fixed element of the frame triad to be the quaternion product

$$f(\theta, \hat{\mathbf{v}}) = q(\theta, \hat{\mathbf{v}}) * b(\hat{\mathbf{v}}) \ , \tag{46}$$

where $*$ denotes quaternion multiplication and all possible frames are described twice since $0 \leq \theta < 4\pi$. To summarize, if we specify a frame axis $\hat{\mathbf{v}}$ to be fixed, then the variable θ in $f(\theta, \hat{\mathbf{v}})$ serves to parameterize a *ring* in quaternion space, each point of which corresponds to a particular 3D frame, and each frame has a diametrically opposite twin.

We argue that, since optimization will typically be done in the full quaternion space, the fact that two opposite-sign quaternions map to the same physical three-space rotation is not a detriment; in fact, it potentially permits an additional stability in the variational process since rotations by $+\pi$

and $-\pi$ are not close to each other in quaternion space as they are in ordinary rotation matrices. In principle, any quaternion Gauss map can be replaced by its exact negative, and the variational process could converge from an ambiguous starting point to either one; the frames would be the same. In our standard projection, the two reflection-equivalent maps are inversions of one another about the 3D origin; their unseen opposite q_0 values can of course cause an additional large separation of the maps in 4D space.

4.1 Full Space of Curve Frames

We can now construct the space of frames step by step using the method above. In Figure 21, we illustrate various views of the construction of the space of frames for the trefoil knot, beginning with a few tangent vectors and the quaternion basis frames corresponding to quaternions that tilt the reference axis into this tangent direction. The circular curve of quaternions representing the space of normal frames is drawn for each tangent; each basis frame touches this curve once. Then the family of these circular curves sweeps out a cylindrical two-manifold, the full space of invariant frames for a 3D curve.

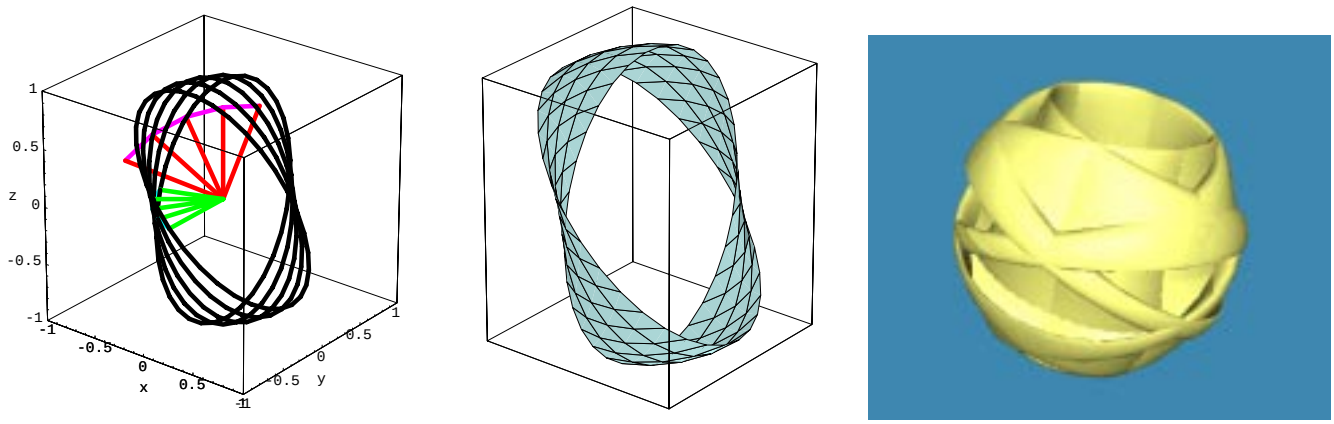
This space has several nontrivial properties. One is that, given one circular ring of frames, a neighboring ring that is a parallel-transported version of the first ring is a so-called "Clifford parallel" of the first ring: the distance from any point on one ring to the nearest point on the second ring is the same. This is nontrivial to visualize and is a feature of the 4D space we are working in. Another property is that the intervals between rings in the quaternion space directly indicate the curvature. This comes about because the magnitude of $\hat{\mathbf{T}}'$ is related to the parallel transport transition between any two sample points, given by Eq. (17); since the parallel transport frames are legal frames, and since the starting frame is arbitrary, each full ring is a parallel transport of its predecessor, with the angular distance of the transition rotation providing a measure of the curvature relative to the sampling interval.

4.2 Full Space of Surface Maps

The full space of frames for a surface patch is even more complex to visualize, since it is a "hyper-cylindrical" 3-manifold, formed by the direct product of patches of surface area with the rings of possible frames through each surface point.

As a very simple case of a surface, consider the patch introduced at the beginning of the paper in Figure 2(a). The coordinate system used does not provide a unique tangent frame, and so one cannot immediately determine a logical frame choice.

In Figure 22, we show spaces of possible frames for the four corners as four rings of quaternion values compatible with the normals at the patch corners. Parallel transporting the initial frame along the two different routes in Figure 2(b,c) produces incompatible frames at the final corner; we represent this situation in Figure 22 by drawing the routes in quaternion space between the initial frame (the degenerate circle appearing as a central vertical line) and the final frame; the mismatch between the two final frames is illustrated by the fact that the two paths meet at different points on the final ring specifying the frame freedom for the bottom corner's frame.



(a) (b) (c)

Figure 21: (a) The first several pieces of the construction of the invariant quaternion space for the frames of the trefoil knot. The red fan of vectors shows the first several elements of the tangent map, represented as vectors from the origin to the surface of the two-sphere and connected by a line. Each green vector points from the origin to the Geodesic Reference element of the quaternion space $q(\arccos(\hat{\mathbf{t}} \cdot \hat{\mathbf{z}}),\ \hat{\mathbf{t}} \times \hat{\mathbf{z}} / \|\hat{\mathbf{t}} \times \hat{\mathbf{z}}\|)$ guaranteed to produce a frame with the tangent $\hat{\mathbf{t}}$. The black curves are the first several elements of the one-parameter space of quaternions representing *all possible* quaternion frames with the tangent $\hat{\mathbf{t}}$. (b) This piece of the space of possible frames represented as a continuous surface, where a circle on the surface corresponds to the space of frames for one point on the curve. (c) The rest of the full constraint space for the trefoil knot. All quaternions are projected to 3D using only the vector part.

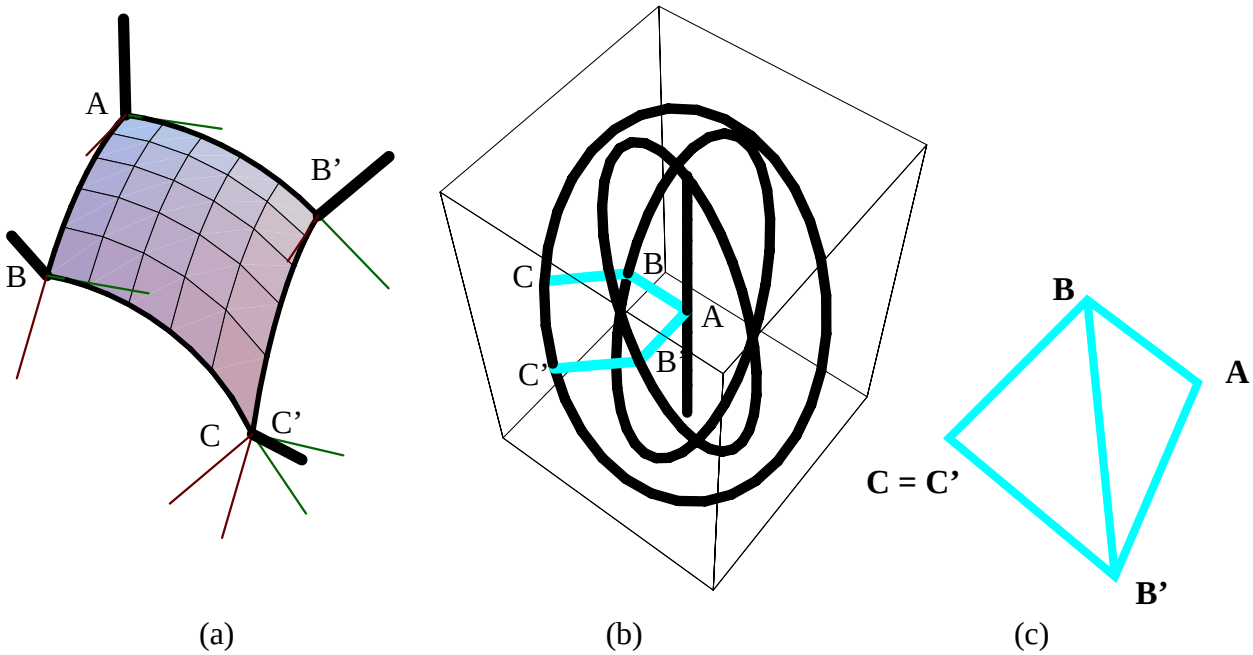
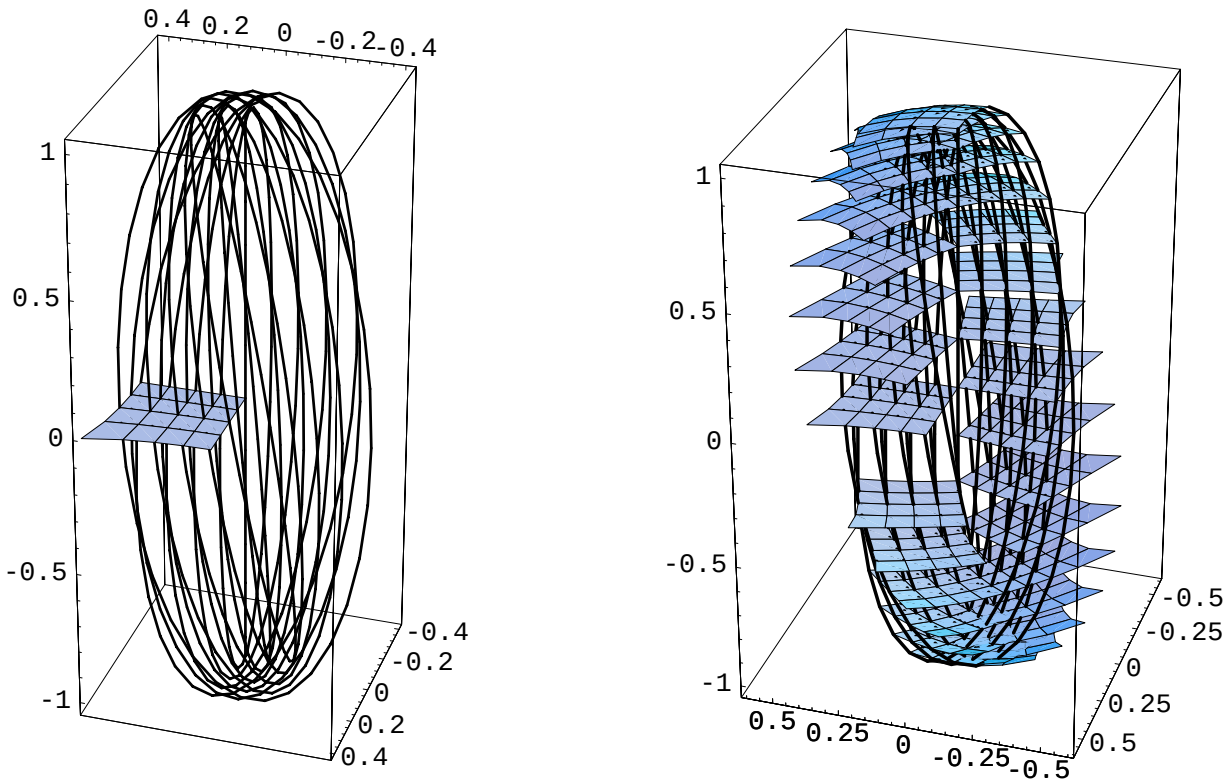


Figure 22: A different viewpoint of the mismatch problem of Figure 2. (a) Choosing different routes to determine the frame at the bottom point results in the incompatible frames shown here in 3D space. (b) The same information is presented here in the quaternion space-of-frames picture. We use throughout a quaternion projection that shows only the 3-vector part of the quaternion, dropping q_0; this is much like projecting away z in a polar projection of the 2-sphere. Each heavy black curve is a ring of possible frame choices that keep fixed the normals in (a); the labels mark the point in quaternion space corresponding to the frames at the corners in (a), so the gap between the labels C and C' represents the frame mismatch in quaternion space on the same constraint ring. (The apparent vertical line is the result of drawing a squashed circle of frames at vertex A in this projection.) (c) The method proposed in this paper to resolve this conflict is to fix one point, say A, divide the polygon $ABCB'$ into triangles, and slide B, C, and B' along the constraint rings until the total triangle areas are minimized, and some compromise with $C = C'$ is reached.



(a) (b)

Figure 23: (a) A more complete picture of the space of frames for this surface patch; the surface shown is a sparse quaternion frame choice for the surface, and we show a subset of the rings of constraints. Each ring passes through one quaternion point on the frame map, the point specifying the current frame choice. Variations must keep each vertex on its ring. (b) An equivalent set of frames is formed by applying a rotation to the entire set of frames. All points follow their own ring of constraints to keep the same normal, These pictures represent the *three-manifold* in quaternion space swept out by the possible variations.

Sliding and Overall Rotational Freedom. In Figure 23(a), We go one step further, and first show how the quaternion Gauss map of an entire patch is situated relative to the ring space; keeping one corner fixed and sliding the rest of the frames around the circular rings takes us to distinct families of frames, which obviously have different areas in the quaternion space. Finally, in Figure 23(b), we keep the fundamental space of frames the same, but exercise the freedom to choose the single parameter describing the basis for the overall orientation; rotating the basis sweeps out both the three-manifold describing the space of frames for this patch, and the family of equivalent frames differing by an insignificant orientation change in the basis vector.

In order to resolve the frame choice ambiguity, one needs a systematic approach; we propose in the next section to accomplish this by optimizing appropriate quantities, e.g., by minimizing the area of the quaternion Gauss map in quaternion space.

We remark that the general features of the surface curvature can in principle be noted from the space of possible frames in a similar manner to that for curves. The family of curves through any point spanning the surfaces tangent space at that point possesses a family of rings parallel to the space of frames at the point, allowing estimates of the rates of change in different directions; the principal curvatures then correspond to the maxima and minima.

5 Choosing Paths in Quaternion Space

We have now seen that the space of possible frames at any point of a curve or surface thus takes the form of a great circle on the unit three-sphere representing the unit quaternions in 4D Euclidean space. While diametrically opposite points on this circle represent the same frame in 3D space, the two-fold redundancy can actually be an advantage, since it helps avoid certain types of wrap-around problems encountered when trying to find paths in the space. Our task then is to select a set of values of the parameter on each of these great circles.

The advantage of looking at this entire problem in the space of quaternions is that one can clearly compare the intrinsic properties of the various choices by examining such properties as length and smoothness in the three-sphere. We note the following issues:

- **Frame-frame distance.** Suppose we are given two neighboring tangents, $\hat{\mathbf{t}}_1$ and $\hat{\mathbf{t}}_2$, and two corresponding candidate frame choices parameterized by θ_1 and θ_2. What is the "distance" in frame space between these? The simplest way to see how we should define the distance is by observing that, by Euler's fundamental theorem, there is a single rotation matrix $R(\theta, \hat{\mathbf{n}})$ That takes one frame to the other; if $R_1(\theta_1, \hat{\mathbf{t}}_1)$ and $R_2(\theta_2, \hat{\mathbf{t}}_2)$ are the two frames, then one can write $R = (R_2 \cdot (R_1)^{-1})$ and solve for θ and $\hat{\mathbf{n}}$. Clearly the value of θ gives a sensible measure of the closeness of the two frames.

- **Quaternion distance.** We remark that essentially the same procedure is required to obtain the parameters of R directly or to find the value of the equivalent quaternion. If we work in quaternion space, we compute $q_1(\theta_1, \hat{\mathbf{t}}_1)$ and $q_2(\theta_2, \hat{\mathbf{t}}_2)$, and then find rather more straightforwardly an equivalent result by noting that the zeroth component of $q = q_2 * (q_1)^{-1}$ is identical to the rotation-invariant scalar product of the two quaternions, $q_1 \cdot q_2$, and thus provides the

needed angle at once:

$$\theta = 2 \arccos(q_1 \cdot q_2) \ .$$

- **Approximation by Euclidean distance.** One can in principle compute quaternions in polar coordinates and use the induced metric on the sphere to compute precise arc-length distance integrals. However, one generally can expect to be dealing with fine tessellations of smoothly varying geometric objects; in this case, it may be sufficient for numerical purposes to estimate frame-to-frame distances using the Euclidean distance in R^4, since the chord of an arc approximates the arc length well for small angles.

Optimal Path Choice Strategies. Why would one want to choose one particular set of values of the frame parameters over another? The most obvious is to keep a tubing from making wild twists such as those that occur for the Frenet frame of a curve with inflection points. In general, one can imagine wanting to minimize the total twisting, the aggregate angular acceleration, etc., subject to a variety of boundary conditions. A bewildering variety of energy functions to minimize can be found in the literature (see, e.g., [6]). We summarize a selection of such criteria for choosing a space of frames below, with the caveat that one certainly may think of others!

- **Minimal Length and Area.** The most obvious criterion is to minimize the total turning angle experienced by the curve frames. Fixing the frames at the ends of a curve may be required by periodicity or external conditions, so one good solution would be one that minimizes the sum total of the turning angles needed to get from the starting to the ending frame. The length to minimize is just the sum of the angles rotated between successive frame choices, as noted above, either exact or approximate. Similar arguments apply to the area of a surface's quaternion Gauss map.

- **Parallel Transport along Geodesics.** Given a particular initial frame, and no further boundary constraints, one may also choose the frame that uses the minimum *local* distance to get between each neighboring frame. Since the parallel transport algorithm corresponding to the Bishop frame uses precisely the smallest possible rotation to get from one frame to the next, this gives the minimal free path that could be computed frame-by-frame. On a surface, the resulting paths are essentially geodesics, but, as noted in Figure 2, there is no obvious analog of a global parallel transport approach to surface framing.

- **Minimal Acceleration.** Barr, Currin, Gabriel, and Hughes [2] proposed a direct generalization of the "no-acceleration" criterion of cubic Euclidean splines for quaternion curves constrained to the three-sphere; the basic concept was to globally minimize the squared tangential acceleration experienced by a curve of unit quaternions. Though the main application of that paper was animation, the principles are entirely valid for numerically computing optimal frames for curves and surfaces in our context.

- **Keyframe splines and constraints.** If for some reason one must pass through or near certain specified frames with possible derivative constraints, then a direct spline construction in the quaternion space may actually be preferred; see, e.g., [37, 35, 31, 39, 23]. Most splines can

be viewed in some sense as solving an optimization problem with various constraints and conditions, and so the keyframe problem essentially reverts again to an optimization.

General Remarks. For both curves and surfaces, there is a single degree of freedom in the frame choice at each point where we have sampled the tangent or normal direction, respectively. This degree of freedom corresponds to a relatively common "sliding ring" constraint that occurs frequently in minimization problems. General packages for solving constraints are mentioned in Barr, et al. [2], who chose MINOS [30]. For our own experiments, we have chosen Brakke's Surface Evolver package [6], which has a very simple interface for handling parametric constraints as "boundary" conditions, and can be used for a wide variety of general optimization problems. Two enhancements to the Evolver have recently been added to handle the specific issues related to quaternion optimization; a new symmetry " `symmetry_group "central_symmetry"` " identifies the quaternion q with $-q$ if desired during the variation to prevent reflected double traversals like that in Figure 13 from varying independently, and the system is now able to use the pull-back metric on the sphere

$$ds^2 = \sum_{i,j} dx_i \, dx_j \, r^{-4} \left(r^2 \, \delta_{i,j} - x_i \, x_j\right)$$

to compute distances directly on the three-sphere. Computation using the metric, however, is very slow, and so in practice we have used the Euclidean R^4 chord approximation, which works quite well for closely spaced samples and is much faster. Yet another alternative proposed by Brakke (private communication) is to use periodic coordinates on S^3 of the form

$$\left(x_1 = \sin r \cos s, \; x_2 = \sin r \sin s, \; x_3 = \cos r \cos t, \; x_4 = \cos r \sin t\right) ,$$

and to vary directly on an R^3 space with $(x = r, \; y = s, \; z = t)$ and the metric

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \sin^2 x & 0 \\ 0 & 0 & \cos^2 x \end{bmatrix} .$$

Our own use of the Evolver required only changing the parameter "`#define BDRYMAX 20`" in `skeleton.h` to the desired (large) value and recompiling. Then, remembering to set "`space_dimension 4`" when working in R^4, one needs in addition a piece of code similar to the following MATHEMATICA fragment to define the boundary constraints for each fixed vector (tangent or normal) and the chosen initial quaternion frame:

```
Do[ringeqn = Qprod[makeQfromVec[veclist[[i]],P1],
                   q0list[[i]]]//Chop;
   Write[file, " boundary ",i," parameters 1"];
   Write[file, "x1: ",  CForm[ ringeqn[[2]]]];
   Write[file, "x2: ",  CForm[ ringeqn[[3]]]];
   Write[file, "x3: ",  CForm[ ringeqn[[4]]]];
   Write[file, "x4: ",  CForm[ ringeqn[[1]]]],
   {i,1,Length[veclist]}]
```

Note that, since Evolver displays only the first three coordinates, we have moved the scalar quaternion to the end; then the Evolver will display our preferred projection automatically.

General Remarks on Optimization in Quaternion Space. Numerical optimization remains a bit of an art, requiring patience and resourcefulness on the part of the investigator. We found, for example, that curve optimization was relatively more stable than surface optimization because single curve outliers add huge amounts to the length, whereas single surface points stuck in a far away crevice may contribute only a tiny amount to the area of a large surface. Although the Evolver in principle handles spherical distances, we used the default 4D Euclidean distance measure as an approximation; this generally corresponded well to explicit area calculations using solid angle performed on the same data sets. However, we did find that extremely random initial conditions (unrealistic for most applications), could produce isolated points stuck in local minima diametrically across quaternion space, at $q \to -q$, from where they should be. This type of problem can be largely avoided simply by running a consistency preprocessor to force nearby neighbors to be on the same side of the three-sphere. Another useful technique is to organize the data into hierarchies and optimize from the coarse scale down to the fine scale. In other cases when things seem unreasonably stuck, a manual "simulated annealing" procedure like that afforded by the Evolver's `jiggle` option often helps.

6 Examples

We now present some examples of frame choices computed using the Evolver to minimize the length of the total path among sliding ring constraints for selected curves, and the total area spanned by analogous sliding rings for surfaces. One interesting result is that there appear to be families of distinct minima: if the initial data for a periodic surface, for example, are set up to return directly to the same point in quaternion space after one period, one has two disjoint surfaces, one the $q \to (-q)$ image of the other; if the data do not naturally repeat after one cycle, they must after two, since there are only two quaternion values that map to the same frame. The family of frame surfaces containing their own reflected images have a minimum distinct from the disjoint family.

Minimal Quaternion Frames for Space Curves. The helix provides a good initial example of the procedure we have formulated. We know that we can always find an initial framing of a curve based on the Geodesic Reference algorithm; however, suppose we wish to impose minimal length in quaternion space on the framing we select, and we do not know whether this frame is optimal with respect to that measure. Then, as illustrated in Figure 24, we can send the ring constraints on the possible quaternion frames at each sample point to the Surface Evolver and let it automatically find the optimal framing. The results and energies for several stages of this evolution are shown in the figure; the final configuration is indistinguishable from the Parallel Transport frame, confirming experimentally our theoretical expectation that parallel transport produces the minimal possible twisting.

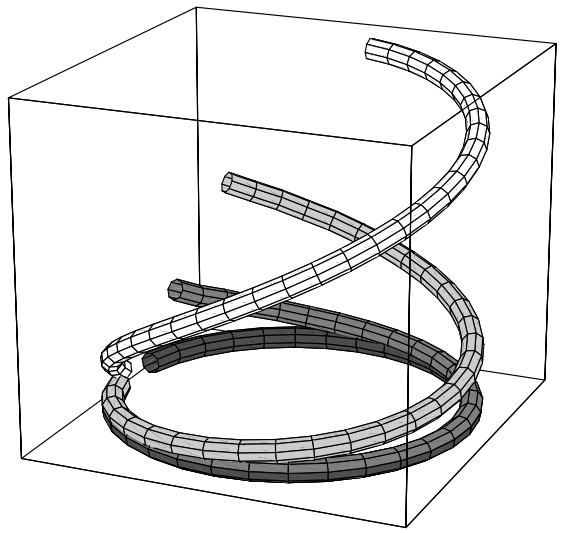


Figure 24: Starting from the Geodesic Reference quaternion frame for a single turn of the helix, the very dark gray circle, the Evolver produces these intermediate steps while minimizing the total quaternion curve length subject to the constraints in the space of frames. The final result is the white curve, which is identical to several decimal points with the Parallel Transport quaternion frame for the same helix. The numerical energies of the curves, from dark to light in color, are 3.03, 2.91, 2.82, and 2.66 for the Parallel Transport frame. The individual tubings used to display these curves are in fact created using the Parallel Transport frame for each individual curve.

In Figure 1, we introduced the question of finding an optimal framing of a particular (3,5) torus knot whose almost-optimal Parallel Transport framing was not periodic. In Figure 25, we show the solution to this problem achieved by clamping the initial and final quaternion frames to coincide, then letting the Evolver pick the shortest quaternion path for all the other frames. It would be possible, as in the case of the (2,3) torus knot framing shown in Figure 13, to have different conditions produce a framing solution containing its own reflected image rather than having a distinct reflected image as is the case for Figure 25.

The types of solutions we find are remarkable in that they should be essentially the same for all reparameterizations of the curve; regardless of the spacing of the sampling, the continuous surface of possible frames is geometrically the same in quaternion space, so paths that are minimal for one sampling should be approximately identical to paths for any reasonable sampling. On the other hand, if we *want* special conditions for certain parameter values, it is easy to fix any number of particular orientations at other points on the curve, just as we fixed the starting points above; derivative values and smoothness constraints leading to generalized splines can be similarly specified (see [2]).

Surface Patch Framings. A classic simple example of a surface patch framing problem was presented in the discussion of Figures 2 and 22. This problem can also be handled by the Evolver:

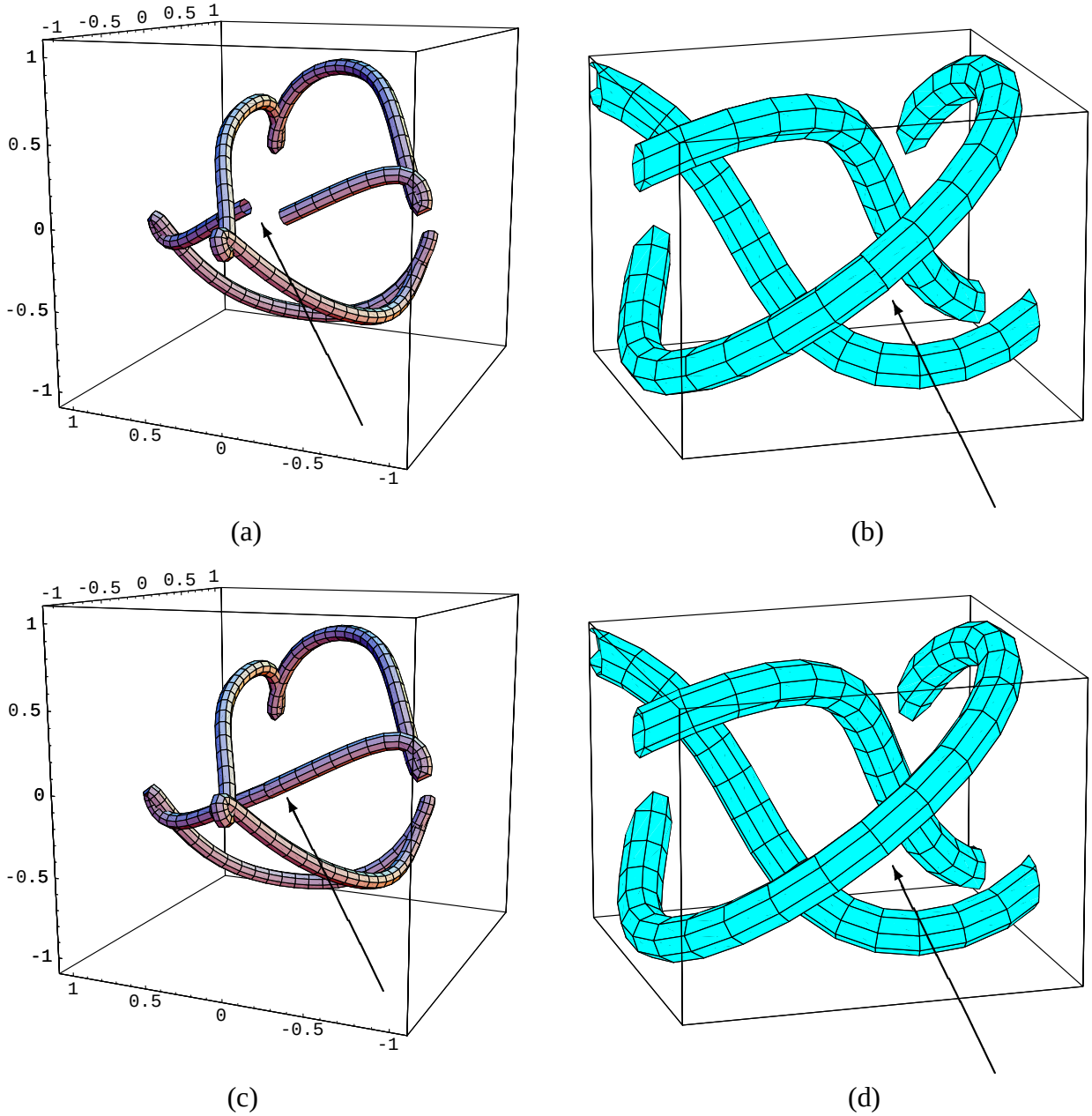


Figure 25: Optimization of the non-periodic parallel transport frame of the (3,5) torus knot introduced in Figure 1 to produce a nearby periodic framing. (a) The original quaternion parallel transport frame used to produce the tubing in Figure 1(b,c). (b) The frame mismatch, repeated for completeness. (c) The result of fixing the final frame to coincide with the initial frame, leaving the other frames free to move on the constraint rings, and minimizing the resulting total length in quaternion space. The length of the original curve was 13.777 and that of the final was 13.700, not a large difference, but noticeable enough in the tube and the quaternion space plot. (d) Closeup of the corresponding framing of the knot in ordinary 3D space, showing that the mismatch problem has been successfully resolved. This tube can *now* be textured, since the frames match exactly.

we choose an initial quaternion frame for the mesh corresponding to one of the arbitrary choices noted, and minimize the area in quaternion space subject to the constraints that the normals remain unchanged, and hence the frame choices may only slide around the rings depicted in Figure 22(b). The results are shown in Figures 26, and 27. As a test, we started one case with a random initial state with a range of 2π in the starting values. All converged to the same optimal final framing. A basic observation is that while none of the standard guesses appeared optimal, the Geodesic Reference frame is very close to optimal for patches that do not bend too much.

Minimal Surfaces. Minimal surfaces possess many special properties following from the fact that the mean curvature is everywhere the vanishing sum of two canceling local principal curvatures [12]. We present a family of classic examples here that is remarkable for the fact that the usual framings are already very close or exactly optimal; thus we do not have much work to do except to admire the results, though there may be some interesting theorems implicit that would be beyond the scope of this paper to pursue.

In Figure 28(a,b,c), we present the following classical minimal surfaces:

$$\mathbf{x}_{\text{catenoid}}(u,\, v) = \cos u\, \cosh v\, \hat{\mathbf{x}} + \sin u\, \cosh v\, \hat{\mathbf{y}} + v\, \hat{\mathbf{z}} \tag{47}$$

$$\mathbf{x}_{\text{helicoid}}(u,\, v) = v \cos u\, \hat{\mathbf{x}} + v \sin u\, \hat{\mathbf{y}} + u\, \hat{\mathbf{z}} \tag{48}$$

$$\mathbf{x}_{\text{enneper}}(u,\, v) = (u - u^3/3 + uv^2)\, \hat{\mathbf{x}} + (v - v^3/3 + vu^2)\, \hat{\mathbf{y}} + (u^2 - v^2)\, \hat{\mathbf{z}} \tag{49}$$

The quaternion Gauss map choices determined by these parameterizations and by the Geodesic Reference algorithm are shown in Figure 29. The coordinate-based catenoid map and helicoid map are 4π double coverings, while Enneper's surface curiously has a coordinate system map that is exactly identical to the Geodesic Reference framing. For the periodic framings of the catenoid and helicoid, we find the noteworthy result that the Geodesic Reference frame, which has a disjoint quaternion reflected image, is a minimum under variations of the surface that is distinct from the quaternion frames derived from the coordinate systems which are *also* minima, but contain their own $q \rightarrow (-q)$ reflected images. The Enneper surface quaternion frames, which are the same, appear to move very slightly around the borders under minimization, but it is not clear to what extent this is significant as opposed to a numerical border effect in the variation. The resulting 3D frame triads are shown in Figure 30 for comparison. A theoretical analysis of the general properties of quaternion Gauss maps for minimal surfaces is beyond the scope of this paper, but experimentally we see that there could be very interesting general properties.

Manifolds. For general manifolds, one must treat patches one at a time in any event, since global frames may not exist at all. Although the locally optimal patches cannot be globally joined to one another, we conjecture that some applications might benefit from the next best thing: matching boundary frames of neighboring patches using transitional rotations (see, e.g., [29, 13]). We have carried this out explicitly for simple cases, but omit it here for brevity.

Extensions to Other Domains. We have focussed for expository purposes in this paper on frames with intrinsic natural constraints imposed by the tangents to curves and normals to sur-

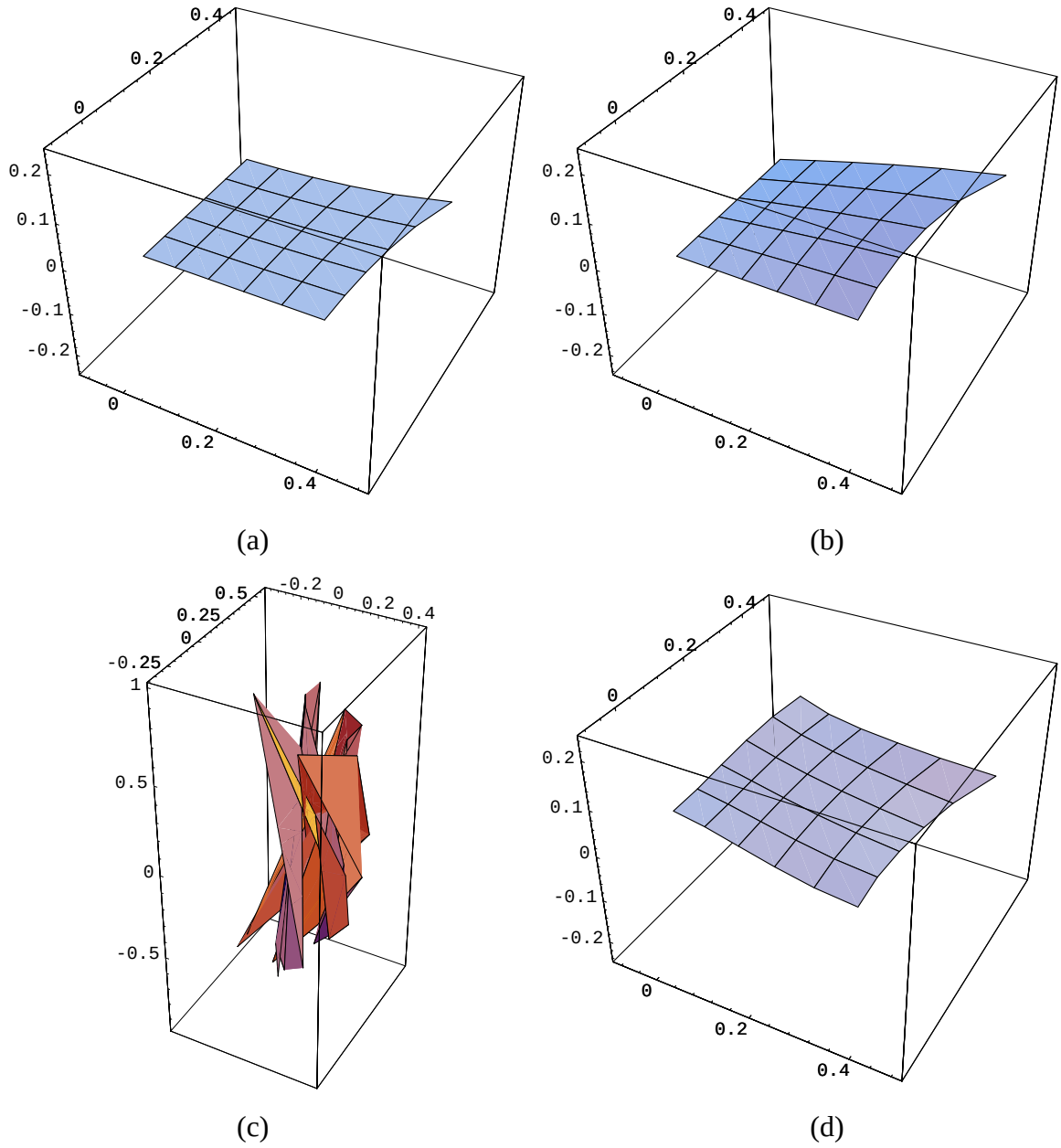


Figure 26: (a) The initial Geodesic Reference quaternions for the small patch shown in Figure 2. (b) Initial quaternions from parallel transporting the vertex frame down one edge, and then across line by line. (c) A random starting configuration with the single same fixed corner point as (a) and (b) and a range of $-\pi$ to $+\pi$ relative to the Geodesic Reference frame. (d) The result of minimization of the quaternion area is the same for all starting points. The relative areas are: 0.147, 0.154, 0.296, and 0.141, respectively. Thus the Geodesic Reference is very close to optimal.

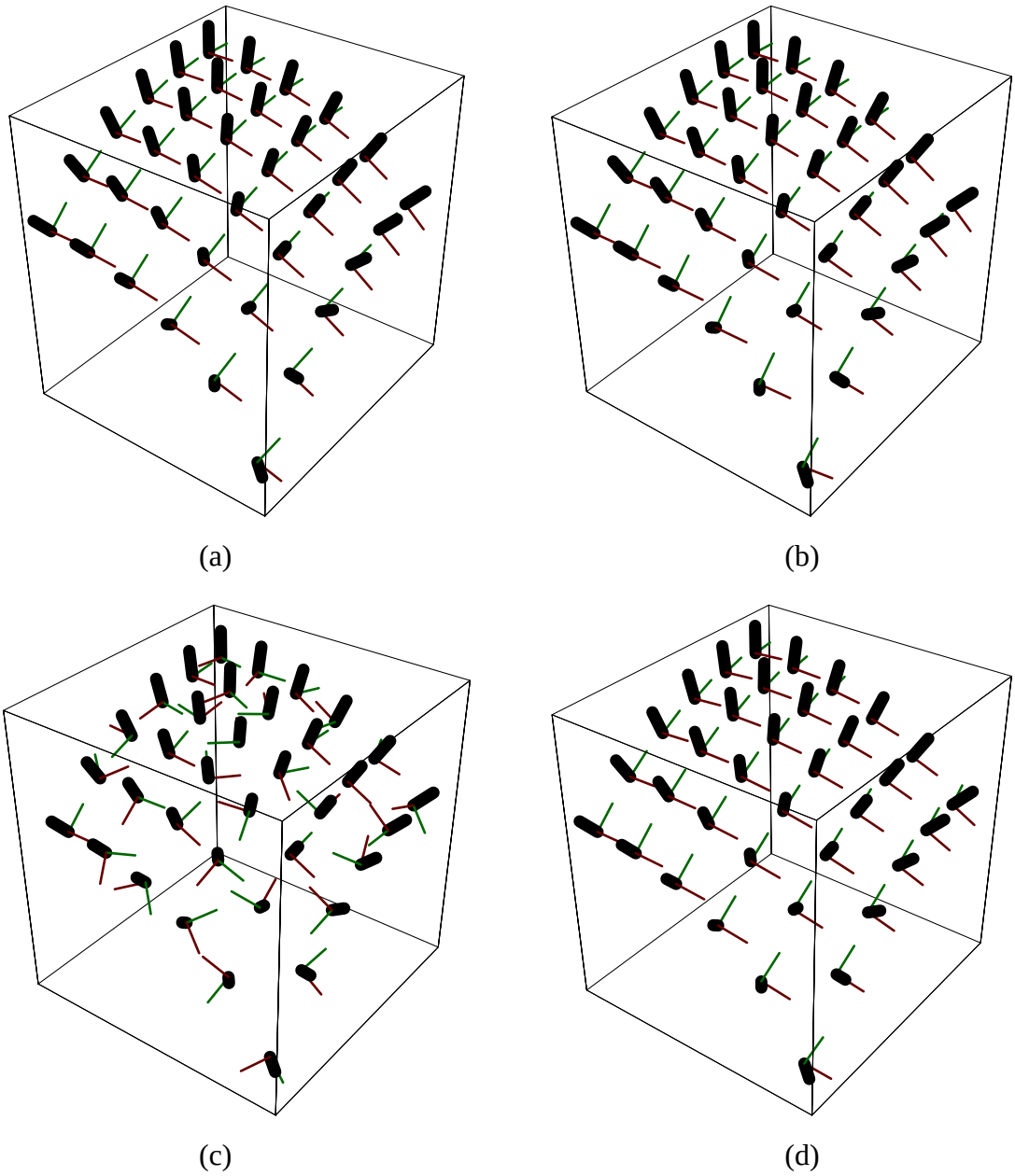


Figure 27: The 3D frame configurations corresponding to the quaternion fields in Figure 26. (a) The Geodesic Reference frame. (b) Two-step parallel transport frame. (c) Random frames. (d) The frame configuration resulting from minimizing area in quaternion space.

faces. However, the method extends almost trivially to applications involving externally specified constraints on frames. Geometric construction algorithms based on extrusions reduce to the tubing problem. For ordinary camera control interpolation, one could constrain any direction of the camera frame to be fixed by calculating its appropriate constraint ring in the quaternion Gauss map, and then extend a method like that of Barr et al. [2, 34]) to smoothly compute intermediate frames subject to the constraints. For more general constrained navigation methods like those described by Hanson and Wernert [20]), the camera vertical direction could be fixed at chosen points over the entire constraint manifold, and the remaining frame parameters determined by optimization within the manifold of ring constraints, possibly subject to fixing entire key-frames at selected locations or boundaries.

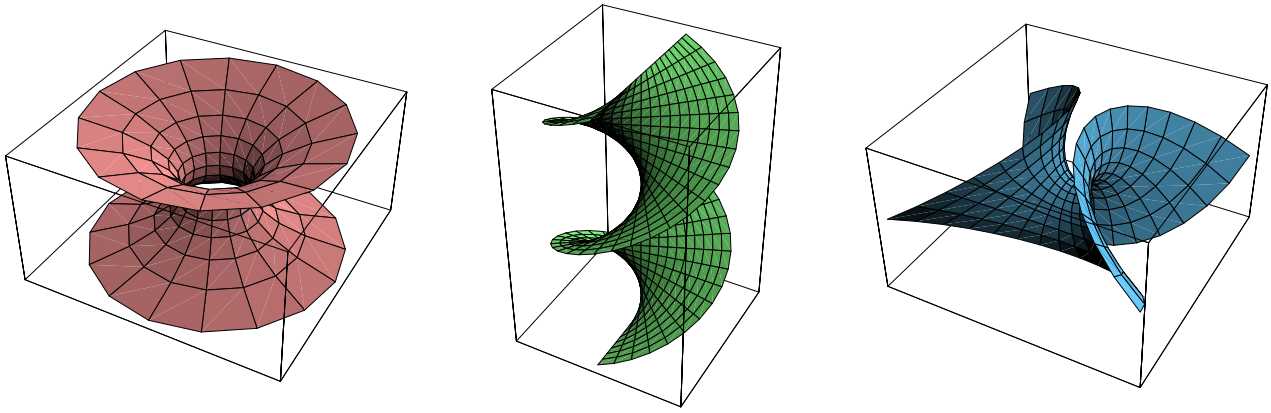
7 Conclusion and Future Directions

We have introduced a general framework derived from the quaternion Gauss map for studying and selecting appropriate families of coordinate frames for curves and surface patches in 3D space. Minimizing length for quaternion curve maps and area for surfaces is proposed as the appropriate generalization of parallel transport for the selection of optimal frame fields. These smooth frames can be used to generate tubular surfaces based on the space curves, thus allowing their effective display on polygon-based graphics engines. The analogous results for surface patches allow the selection of optimal local coordinate systems that may be adapted for display purposes and related applications such as oriented particle systems. Our principal new tool is the space of all possible frames, a manifold of constraints immersed in the space of quaternion frames. By defining energies and boundary conditions in this space one can produce a rich variety of application-adapted criteria for specifying optimal families of frames.

Topics for future investigation include the treatment of manifolds in higher dimensional spaces, improved interfaces for visualizing the quaternion optimization process and its results, and further analysis of the pure mathematics implied by the general framework. N-dimensional generalizations of the Frenet frame equations have been studied in the literature (see, e.g., Forsyth [10]), but the analogs of quaternions in higher dimensions are much more complex and involve Clifford algebras and the corresponding Spin groups (see, e.g., Lawson and Michelson [26]). Special simplifications do occur for the 4D case, however, allowing a treatment in terms of pairs of unit quaternions (see, e.g., [15]); this case must in fact be investigated to produce a more rigorous formulation of the 4D surface tubings proposed in [17]. Among other applications that may be approached by the quaternion formulation of coordinate frames we note the description of anisotropic surfaces (see, e.g., Kajiya [22]), the quaternion generalization of bump-mapping, and the dynamics of anisotropic particle systems. Another possible application could be the determination of optimal configurations for long-chain molecules and similar 1D and 2D structures. There are thus ample challenges for future work.

Acknowledgments

The author gratefully acknowledges the cordial hospitality of Claude Puech and the members of the iMAGIS project, a joint project of CNRS, INRIA, Institut National Polytechnique de Grenoble, and Université Joseph Fourier. We were also fortunate to have the collaboration of Hui Ma in developing the early foundations of this work. Finally, we are deeply indebted to Ken Brakke for his readiness to provide a wealth of mathematical insight as well as assistance in effectively using the Evolver system. This research was made possible in part by NSF infrastructure grant CDA 93-03189.



(a) (b) (c)

Figure 28: (a) The catenoid, a classic minimal surface in 3D space with a natural orthonormal parameterization. (b) The helicoid. (c) Enneper's surface.

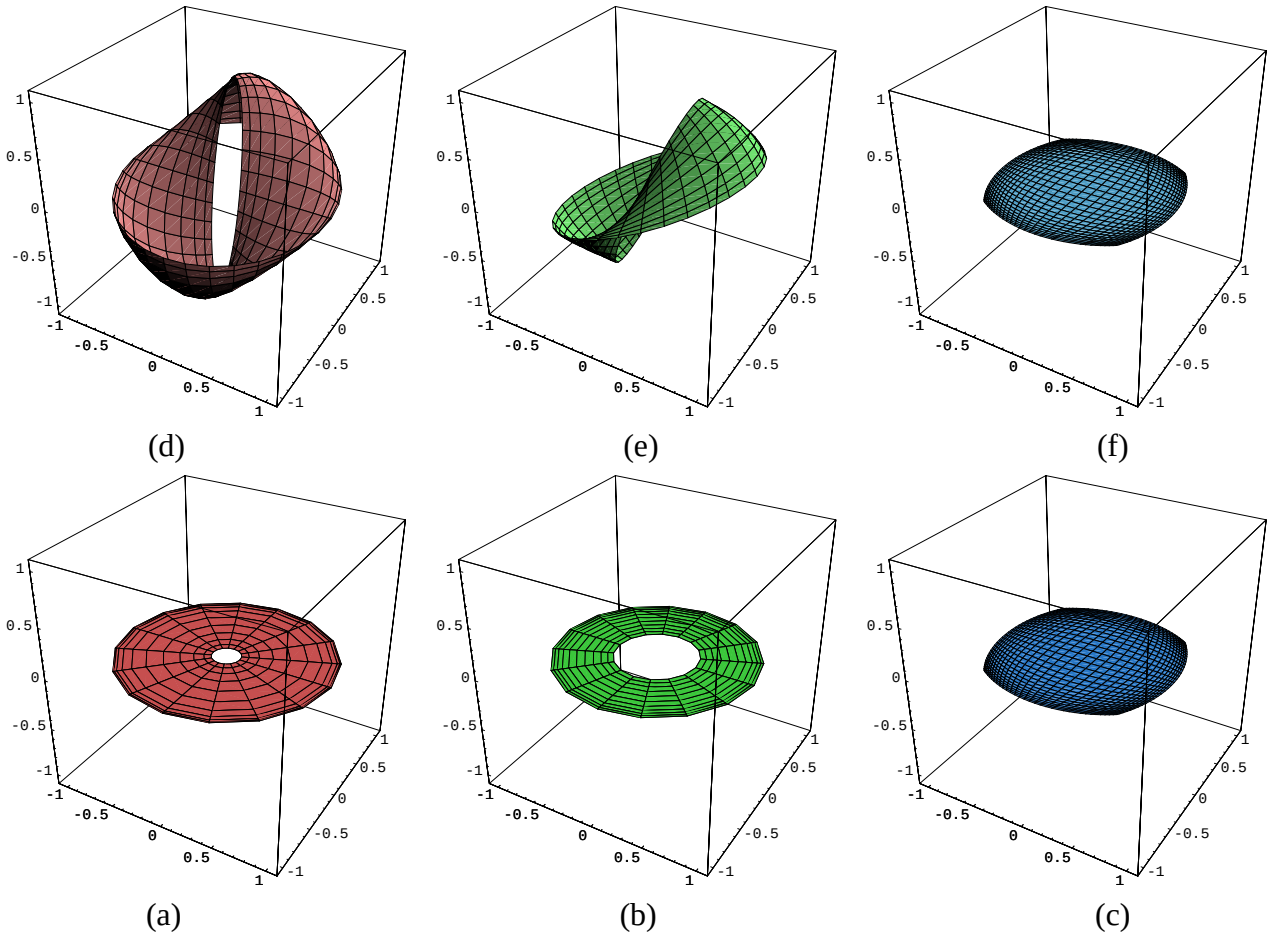


Figure 29: The Geodesic Reference quaternion frames of (a) the catenoid, (b) the helicoid, and (c) Enneper's surface. (d, e, f) The corresponding quaternion Gauss maps determined directly from the parameterization. Both the catenoid and the helicoid fail to be cyclic in quaternion space without a 4π turn around the repeating direction, so these are doubled maps. The Enneper's surface framing turns out to be identical to its Geodesic Reference frame.

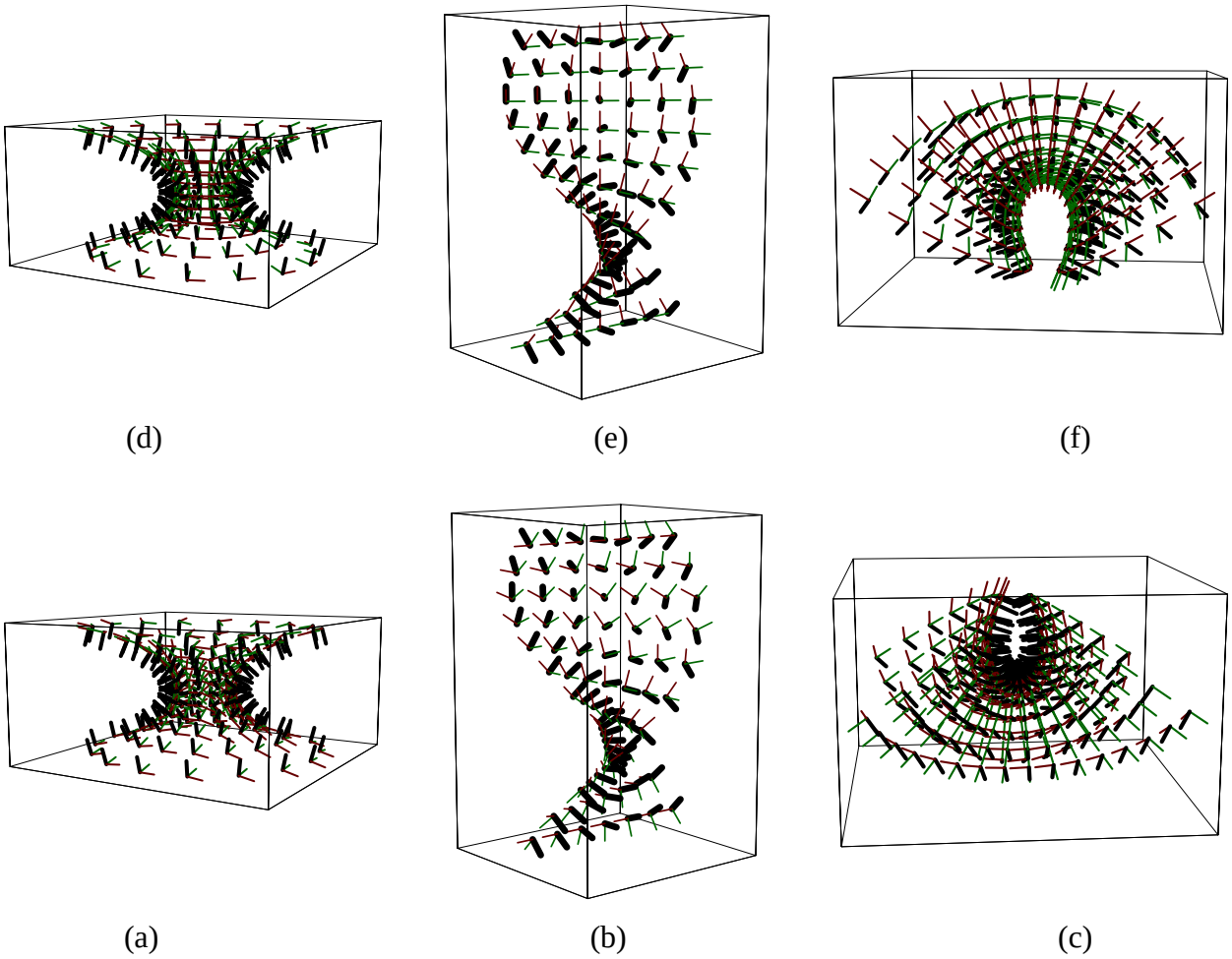


Figure 30: The 3D Geodesic Reference frames displayed directly on the surfaces of (a) the catenoid, (b) the helicoid, and (c) Enneper's surface. (d,e,f) The 3D frames computed directly from the standard parameterizations; since Enneper's surface is the same, we show in (f) a different viewpoint of the same frames.

Appendix: Quaternion Frames

2D "Quaternion" Frames

We provide below an exercise that may give some insight into the quaternion world. We show that we may express the 2D frame equations in terms of a new set of variables exactly analogous to quaternions in 3D. We begin by guessing a double-valued quadratic form for the frame:

$$\begin{bmatrix}\hat{\mathbf{N}} & \hat{\mathbf{T}}\end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} a^2-b^2 & -2ab \\ 2ab & a^2-b^2 \end{bmatrix} . \tag{50}$$

We can easily verify that if $a^2+b^2=1$, this is an orthonormal parameterization of the frame, and that θ is related to (a,b) by the half-angle formulas:

$$a=\cos(\theta/2),\ b=\sin(\theta/2) .$$

If desired, the redundant parameter can be eliminated locally by using projective coordinates such as $c=b/a=\tan(\theta/2)$ to get the form (compare Eq. (30))

$$\begin{bmatrix}\hat{\mathbf{N}} & \hat{\mathbf{T}}\end{bmatrix} = \frac{1}{1+c^2}\begin{bmatrix} 1-c^2 & -2c \\ 2c & 1-c^2 \end{bmatrix} . \tag{51}$$

If we now define $W_1=\begin{bmatrix} a & -b \\ b & a \end{bmatrix}$ and $W_2=\begin{bmatrix} -b & -a \\ a & -b \end{bmatrix}$, then we may write

$$2W_1\cdot\begin{bmatrix} a' \\ b' \end{bmatrix} = \hat{\mathbf{N}}',\ \ 2W_2\cdot\begin{bmatrix} a' \\ b' \end{bmatrix} = \hat{\mathbf{T}}'$$

and we may also express the right-hand side of the 2D frame equations as

$$W_1\cdot\begin{bmatrix} 0 & -\kappa \\ +\kappa & 0 \end{bmatrix}\cdot\begin{bmatrix} a \\ b \end{bmatrix} = +v\kappa\hat{\mathbf{T}} .$$

The analogous expression for W_2 yields $\hat{\mathbf{T}}'=-v\kappa\hat{\mathbf{N}}$. Matching terms and multiplying by $W_i^T=W_i^{-1}$, we find that the equation

$$\begin{bmatrix} a' \\ b' \end{bmatrix} = \frac{1}{2}v\begin{bmatrix} 0 & -\kappa \\ +\kappa & 0 \end{bmatrix}\cdot\begin{bmatrix} a \\ b \end{bmatrix} \tag{52}$$

contains *both* the frame equations $\hat{\mathbf{T}}'=-\kappa\hat{\mathbf{N}}$ and $\hat{\mathbf{N}}'=+\kappa\hat{\mathbf{T}}$, but now in 2D "quaternion" space! If we take the angular range from $0\to 4\pi$ instead of 2π, we have a $2:1$ quadratic mapping from (a,b) to $(\hat{\mathbf{N}},\ \hat{\mathbf{T}})$ because $(a,b)\sim(-a,-b)$ in Eq. (50).

Equation (52) is the *square root* of the frame equations (note the factor of $(1/2)$). The curvature matrix is basically $g^{-1}\,dg$, an element of the Lie algebra for the 2D rotation "spin group," and takes the explicit form,

$$\begin{bmatrix} a & b \\ -b & a \end{bmatrix}\begin{bmatrix} a' & -b' \\ b' & a' \end{bmatrix} = \begin{bmatrix} aa'+bb' & -ab'+ba' \\ ab'-ba' & aa'+bb' \end{bmatrix} .$$

Here $aa' + bb' = 0$ due to the constraint $a^2 + b^2 = 1$, and

$$\begin{aligned} ab' - ba' &= \cos\frac{\theta}{2}\left[\frac{\theta'}{2}\cos\frac{\theta}{2}\right] - \sin\frac{\theta}{2}\left[-\frac{\theta'}{2}\sin\frac{\theta}{2}\right] \\ &= \frac{\theta'}{2}\,, \end{aligned}$$

giving the identification $v\kappa = \theta'$ when we pull out the factor of $1/2$ as in Eq. (52). The actual group properties in (a, b) space follow from the multiplication rule (easily deduced from the formulas for the trigonometric functions of sums of angles)

$$(a,\, b) * (\tilde{a},\, \tilde{b}) = (a\tilde{a} - b\tilde{b},\, a\tilde{b} + b\tilde{a})\,,$$

which is in turn isomorphic to complex multiplication with $(a, b) = a + ib$. This is no surprise, since SO(2) and its double covering spin group are subgroups of the corresponding 3D rotation groups, and complex numbers are a subset of the quaternions.

3D Quaternion Frames

We next outline the basic features of quaternion frames; see, e.g., [1] for a nice textbook treatment of quaternions and their properties.

A quaternion frame is a four-vector $q = (q_0, q_1, q_2, q_3) = (q_0, \vec{q})$ with the following features:

- **Unit Norm.** If we define the inner product of two quaternions as

$$q \cdot p = q_0 p_0 + q_1 p_1 + q_2 p_2 + q_3 p_3\,, \tag{53}$$

then the components of a quaternion frame obey the constraint

$$q \cdot q = (q_0)^2 + (q_1)^2 + (q_2)^2 + (q_3)^2 = 1\,, \tag{54}$$

and therefore lie on S^3, the three-sphere embedded in four-dimensional Euclidean space R^4.

- **Multiplication rule.** The quaternion product of two quaternions p and q is defined to give a positive cross-product in the vector part, and may be written as

$$p * q = (p_0 q_0 - \mathbf{p}\cdot\mathbf{q},\, p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p}\times\mathbf{q})\,,$$

or more explicitly in component form as

$$p * q = \begin{bmatrix} [p*q]_0 \\ [p*q]_1 \\ [p*q]_2 \\ [p*q]_3 \end{bmatrix} = \begin{bmatrix} p_0 q_0 - p_1 q_1 - p_2 q_2 - p_3 q_3 \\ p_1 q_0 + p_0 q_1 + p_2 q_3 - p_3 q_2 \\ p_2 q_0 + p_0 q_2 + p_3 q_1 - p_1 q_3 \\ p_3 q_0 + p_0 q_3 + p_1 q_2 - p_2 q_1 \end{bmatrix}\,. \tag{55}$$

This rule is isomorphic to left multiplication in the group SU(2), the double covering of the ordinary 3D rotation group SO(3). What is more useful for our purposes is the fact that it

is also isomorphic to multiplication by a member of the group of (possibly sign-reversing) orthogonal transformations in R^4:

$$p * q = [P]\, q = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & -p_3 & p_2 \\ p_2 & p_3 & p_0 & -p_1 \\ p_3 & -p_2 & p_1 & p_0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} , \tag{56}$$

where $[P]$ is an orthogonal matrix, $[P]^t \cdot [P] = I_4$; since $[P]$ has only 3 free parameters, it does not itself include all 4D rotations. However, we may recover the remaining 3 parameters by considering transformation by right multiplication to be an independent operation, resulting in a similar matrix but with the signs in the lower right-hand off-diagonal 3×3 section reversed. (This corresponds to the well-known decomposition of the 4D rotation group into two 3D rotations; see, e.g., [14].)

If two quaternions a and b are transformed by multiplying them by the same quaternion p, their inner product $a \cdot b$ transforms as

$$(p * a) \cdot (p * b) = (a \cdot b)(p \cdot p) \tag{57}$$

and so is invariant if p is a unit quaternion frame representing a rotation. This also follows trivially from the fact that $[P]$ is orthogonal.

The *inverse* of a unit quaternion satisfies $q * q^{-1} = (1,\, \mathbf{0})$ and is easily shown to take the form $q^{-1} = (q_0,\, -\mathbf{q})$. The relative quaternion rotation t transforming between two quaternions may be represented using the product

$$t = p * q^{-1} = (p_0 q_0 + \mathbf{p} \cdot \mathbf{q},\, q_0 \mathbf{p} - p_0 \mathbf{q} - \mathbf{p} \times \mathbf{q}) \, .$$

This has the convenient property that the zeroth component is the invariant 4D inner product $p \cdot q = \cos(\theta/2)$, where θ is the angle of the rotation in 3D space needed to rotate along a geodesic from the frame denoted by q to that given by p. In fact, the 4D inner product reduces to

$$p * q^{-1} + q * p^{-1} = (2q \cdot p,\, \mathbf{0}) \, ,$$

while the 3D dot product and cross product arise from the symmetric and antisymmetric sums of quaternions containing only a 3-vector part:

$$\begin{aligned} \mathbf{p} * \mathbf{q} &\equiv (0,\, \mathbf{p}) * (0,\, \mathbf{q}) = (-\mathbf{p} \cdot \mathbf{q},\, \mathbf{p} \times \mathbf{q}) \\ \mathbf{p} * \mathbf{q} + \mathbf{q} * \mathbf{p} &= -2\, \mathbf{p} \cdot \mathbf{q} \\ \mathbf{p} * \mathbf{q} - \mathbf{q} * \mathbf{p} &= 2\, \mathbf{p} \times \mathbf{q} \end{aligned}$$

- **Mapping to 3D rotations.** Every possible 3D rotation R (a 3×3 orthogonal matrix) can be constructed from either of two related quaternions, $q = (q_0, q_1, q_2, q_3)$ or $-q =$

$(-q_0, -q_1, -q_2, -q_3)$, using the quadratic relationship $R_q(\mathbf{V}) = q * (0, \mathbf{V}) * q^{-1}$, written explicitly as

$$R = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} . \tag{58}$$

The signs here result from choosing the left multiplication convention $R_pR_q(\mathbf{V}) = R_{pq}(\mathbf{V}) = (p * q) * (0, \mathbf{V}) * (p * q)^{-1}$. Algorithms for the inverse mapping from R to q require careful singularity checking, and are detailed, e.g., in [31, 38].

The analog for Eq. (58) of the projective coordinates for 2D rotations noted in Eq. (51) is obtained by converting to the projective variable $\mathbf{c} = \mathbf{q}/q_0 = \tan(\theta/2)\,\hat{\mathbf{n}}$ and factoring out

$$(q_0)^2 = \frac{(q_0)^2}{(q_0)^2 + \mathbf{q}\cdot\mathbf{q}} = \frac{1}{1 + \mathbf{q}\cdot\mathbf{q}/(q_0)^2} = \frac{1}{1 + \|\mathbf{c}\|^2} .$$

We then find

$$R = \frac{1}{1 + \|\mathbf{c}\|^2} \begin{bmatrix} 1 + c_1^2 - c_2^2 - c_3^2 & 2c_1c_2 - 2c_3 & 2c_1c_3 + 2c_2 \\ 2c_1c_2 + 2c_3 & 1 - c_1^2 + c_2^2 - c_3^2 & 2c_2c_3 - 2c_1 \\ 2c_1c_3 - 2c_2 & 2c_2c_3 + 2c_1 & 1 - c_1^2 - c_2^2 + c_3^2 \end{bmatrix} . \tag{59}$$

- **Rotation Correspondence.** When we substitute $q(\theta, \hat{\mathbf{n}}) = (\cos\frac{\theta}{2}, \hat{\mathbf{n}}\sin\frac{\theta}{2})$ into Eq. (58), where $\hat{\mathbf{n}}\cdot\hat{\mathbf{n}} = 1$ is a unit three-vector lying on the two-sphere S^2, $R(\theta, \hat{\mathbf{n}})$ becomes the standard matrix for a rotation by θ in the plane perpendicular to $\hat{\mathbf{n}}$. The quadratic form ensures that the two distinct unit quaternions q and $-q$ in S^3 correspond to the *same* $\mathrm{SO}(3)$ rotation. For reference, the explicit form of $R(\theta, \hat{\mathbf{n}})$ is [9]

$$\mathbf{R}(\theta, \hat{\mathbf{n}}) = \begin{bmatrix} c + (n_1)^2(1-c) & n_1n_2(1-c) - sn_3 & n_3n_1(1-c) + sn_2 \\ n_1n_2(1-c) + sn_3 & c + (n_2)^2(1-c) & n_3n_2(1-c) - sn_1 \\ n_1n_3(1-c) - sn_2 & n_2n_3(1-c) + sn_1 & c + (n_3)^2(1-c) \end{bmatrix} , \tag{60}$$

where $c = \cos\theta$, $s = \sin\theta$, and $\hat{\mathbf{n}}\cdot\hat{\mathbf{n}} = 1$. For example, choosing the quaternion $q = (\cos\frac{\theta}{2}, 0, 0, \sin\frac{\theta}{2})$ yields the rotation matrix

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} ,$$

producing a right-handed rotation of the basis vectors $\hat{\mathbf{x}} = (1, 0, 0)$ and $\hat{\mathbf{y}} = (0, 1, 0)$ around the $\hat{\mathbf{z}}$ axis.

- **Quaternion Frame Evolution.** All 3D coordinate frames can be expressed in the form of quaternions using Eq. (58). If we assume the columns of Eq. (58) are the vectors $(\hat{\mathbf{N}}_1, \hat{\mathbf{N}}_2, \hat{\mathbf{T}})$,

respectively, one can explicitly express each vector in terms of the following matrices

$$[W_1] = \begin{bmatrix} q_0 & q_1 & -q_2 & -q_3 \\ q_3 & q_2 & q_1 & q_0 \\ -q_2 & q_3 & -q_0 & q_1 \end{bmatrix} \tag{61}$$

$$[W_2] = \begin{bmatrix} -q_3 & q_2 & q_1 & -q_0 \\ q_0 & -q_1 & q_2 & -q_3 \\ q_1 & q_0 & q_3 & q_2 \end{bmatrix} \tag{62}$$

$$[W_3] = \begin{bmatrix} q_2 & q_3 & q_0 & q_1 \\ -q_1 & -q_0 & q_3 & q_2 \\ q_0 & -q_1 & -q_2 & q_3 \end{bmatrix} , \tag{63}$$

with the result that $\hat{\mathbf{N}}_1 = [W_1] \cdot [q]$, $\hat{\mathbf{N}}_2 = [W_2] \cdot [q]$, and $\hat{\mathbf{T}} = [W_3] \cdot [q]$, where $[q]$ is the column vector with components (q_0, q_1, q_2, q_3). Differentiating each of these expressions and substituting Eq. (9), one finds that factors of the matrices $[W_i]$ can be pulled out and a single universal equation linear in the quaternions remains:

$$\begin{bmatrix} q_0' \\ q_1' \\ q_2' \\ q_3' \end{bmatrix} = \frac{v}{2} \begin{bmatrix} 0 & -k_x & -k_y & -k_z \\ +k_x & 0 & -k_z & +k_y \\ +k_y & +k_z & 0 & -k_x \\ +k_z & -k_y & +k_x & 0 \end{bmatrix} \cdot \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} . \tag{64}$$

The first occurrence of this equation that we are aware of is in the works of Tait [41]. Here $v(t) = \|\mathbf{x}'(t)\|$ is the scalar magnitude of the curve derivative if a unit-speed parameterization is not being used for the curve. One may consider Eq. (64) to be in some sense the *square root* of the 3D frame equations. Alternatively, we can deduce directly from $R_q(\mathbf{V}) = q * (0, \mathbf{V}) * q^{-1}$, $dq = q * (q^{-1} * dq)$, and $(q^{-1} * dq) = -(dq^{-1} * q)$, that the 3D vector equations are equivalent to the quaternion form

$$q' = \frac{1}{2} v\, q * (0,\, k_x,\, k_y,\, k_z) = \frac{1}{2}\, v\, q * (0,\, \mathbf{k}) \tag{65}$$

$$(q^{-1})' = -\frac{1}{2}\, v\, (0,\, \mathbf{k}) * q^{-1} , \tag{66}$$

where $\mathbf{k} = 2(q_0\, \mathbf{dq} - \mathbf{q}\, dq_0 - \mathbf{q} \times \mathbf{dq})$, or, explicitly,

$$\begin{aligned} k_0 &= 2(dq_x q_x + dq_y q_y + dq_z q_z + dq_0 q_0) = 0 \\ k_x &= 2(q_0 dq_x - q_x dq_0 - q_y dq_z + q_z dq_y) \\ k_y &= 2(q_0 dq_y - q_y dq_0 - q_z dq_x + q_x dq_z) \\ k_z &= 2(q_0 dq_z - q_z dq_0 - q_x dq_y + q_y dq_x) . \end{aligned}$$

Here $k_0 = 0$ is the diagonal value in Eq. (64).

The quaternion approach to the frame equations exemplified by Eq. (64) (or Eq. (65)) has the following key properties:

- $q(t) \cdot q'(t) = 0$ by construction. Thus all unit quaternions remain unit quaternions as they evolve by this equation.
- The number of equations has been reduced from nine coupled equations with six orthonormality constraints to four coupled equations incorporating a single constraint that keeps the solution vector confined to the three-sphere.

- **Quaternion Surface Evolution.** The same set of equations can be considered to work on curves that are paths in a surface, thus permitting a quaternion equivalent to the Weingarten equations for the classical differential geometry of surfaces as well. Explicit forms permitting the recovery of the classical equations follow from re-expressing Eqs. (22) and (23) in quaternion form. The curvature equation is essentially the cross-product of two derivatives of the form of Eqs. (65,66), and thus obtainable by a quaternion multiplication:

$$\begin{aligned} q_u * q_v^{-1} &= -\frac{1}{4}\, q * (0, \mathbf{a}) * (0, \mathbf{b}) * q^{-1} \\ &= -\frac{1}{4}\, q * (-\mathbf{a} \cdot \mathbf{b},\, \mathbf{a} \times \mathbf{b}) * q^{-1} \\ &= -\frac{1}{4}\, \left[-\mathbf{a} \cdot \mathbf{b}\, \hat{\mathbf{I}} + (\mathbf{a} \times \mathbf{b})_x\, \hat{\mathbf{T}}_1 + (\mathbf{a} \times \mathbf{b})_y\, \hat{\mathbf{T}}_2 + (\mathbf{a} \times \mathbf{b})_z\, \hat{\mathbf{N}} \right] . \end{aligned} \tag{67}$$

Here Eq. (31) defines the quaternion frame vectors, $\hat{\mathbf{N}} \equiv (0,\, \hat{\mathbf{N}}) = q * (0, \hat{\mathbf{z}}) * q^{-1}$, etc., and we have introduced the identity element $\hat{\mathbf{I}} = (1,\, \mathbf{0}) = q * (1,\, \mathbf{0}) * q^{-1}$ as a fourth quaternion basis vector. The mean curvature equation has only one derivative and a free vector field; an expression producing the right combination of terms is

$$\begin{aligned} \hat{\mathbf{T}}_1 * q * q_u^{-1} + \hat{\mathbf{T}}_2 * q * q_v^{-1} &= -\frac{1}{2} q * (-\hat{\mathbf{x}} \cdot \mathbf{a} - \hat{\mathbf{y}} \cdot \mathbf{b},\, \hat{\mathbf{x}} \times \mathbf{a} + \hat{\mathbf{y}} \times \mathbf{b}) * q^{-1} \\ &= -\frac{1}{2} \left[-(a_x + b_y)\hat{\mathbf{I}} + b_z \hat{\mathbf{T}}_1 - a_z \hat{\mathbf{T}}_2 + (a_y - b_x)\hat{\mathbf{N}} \right] \end{aligned} \tag{68}$$

Projecting out the $\hat{\mathbf{N}}$ component of these equations recovers the scalar and mean curvatures:

$$\begin{aligned} K &= \det \begin{bmatrix} +a_y(u,v) & -a_x(u,v) \\ +b_y(u,v) & -b_x(u,v) \end{bmatrix} = a_x b_y - a_y b_x \\ H &= \frac{1}{2} \operatorname{tr} \begin{bmatrix} +a_y(u,v) & -a_x(u,v) \\ +b_y(u,v) & -b_x(u,v) \end{bmatrix} = \frac{1}{2}(a_y - b_x) . \end{aligned}$$

References

[1] ALTMANN, S. L. *Rotations, Quaternions, and Double Groups*. Oxford University Press, 1986.

[2] BARR, A. H., CURRIN, B., GABRIEL, S., AND HUGHES, J. F. Smooth interpolation of orientations with angular velocity constraints using quaternions. In *Computer Graphics (SIGGRAPH '92 Proceedings)* (July 1992), E. E. Catmull, Ed., vol. 26, pp. 313–320.

[3] BERGER, M. *Geometry I, II*. Springer Verlag, Berlin, 1987.

[4] BISHOP, R. L. There is more than one way to frame a curve. *Amer. Math. Monthly 82*, 3 (March 1975), 246–251.

[5] BLOOMENTHAL, J. Calculation of reference frames along a space curve. In *Graphics Gems*, A. Glassner, Ed. Academic Press, Cambridge, MA, 1990, pp. 567–571.

[6] BRAKKE, K. A. The surface evolver. *Experimental Mathematics 1*, 2 (1992), 141–165. The "Evolver" system, manual, and sample data files are available by anonymous ftp from geom.umn.edu, The Geometry Center, Minneapolis MN.

[7] EGUCHI, T., GILKEY, P., AND HANSON, A. Gravitation, gauge theories and differential geometry. *Physics Reports 66*, 6 (December 1980), 213–393.

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

[9] FOLEY, J., VAN DAM, A., FEINER, S., AND HUGHES, J. *Computer Graphics, Principles and Practice*, second ed. Addison-Wesley, 1990. page 227.

[10] FORSYTH, A. R. *Geometry of Four Dimensions*. Cambridge University Press, 1930.

[11] GABRIEL, S., AND KAJIYA, J. T. Spline interpolation in curved space. In *State of the Art Image Synthesis* (1985). Siggraph '85 Course notes.

[12] GRAY, A. *Modern Differential Geometry of Curves and Surfaces with Mathematica*, second ed. CRC Press, Inc., Boca Raton, FL, 1998.

[13] GRIMM, C. M., AND HUGHES, J. F. Modeling surfaces with arbitrary topology using manifolds. In *Computer Graphics Proceedings, Annual Conference Series* (1995), pp. 359–368. Proceedings of SIGGRAPH '95.

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

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

[16] HANSON, A. J. Constrained optimal framings of curves and surfaces using quaternion gauss maps. In *Proceedings of Visualization '98* (1998), IEEE Computer Society Press, pp. 375–382. Missing pages 2 and 4 in original Proceedings; correct version in second printing of Procedings, on conference CDROM, or ftp://ftp.cs.indiana.edu/pub/hanson/Vis98/vis98.quat.pdf.

[17] HANSON, A. J., AND HENG, P. A. Illuminating the fourth dimension. *Computer Graphics and Applications 12*, 4 (July 1992), 54–62.

[18] HANSON, A. J., AND MA, H. Visualizing flow with quaternion frames. In *Proceedings of Visualization '94* (1994), IEEE Computer Society Press, pp. 108–115.

[19] HANSON, A. J., AND MA, H. Quaternion frame approach to streamline visualization. *IEEE Trans. on Visualiz. and Comp. Graphics 1*, 2 (June 1995), 164–174.

[20] HANSON, A. J., AND WERNERT, E. A. Constrained 3d navigation with 2d controllers. In *Proceedings of Visualization '97* (1997), IEEE Computer Society Press, pp. 175–182.

[21] HART, J. C., FRANCIS, G. K., AND KAUFFMAN, L. H. Visualizing quaternion rotation. *ACM Trans. on Graphics 13*, 3 (1994), 256–276.

[22] KAJIYA, J. T. Anisotropic reflection models. In *Computer Graphics* (1985), vol. 19, pp. 15–21. Proceedings of SIGGRAPH '85.

[23] KIM, M.-J., KIM, M.-S., AND SHIN, S. Y. A general construction scheme for unit quaternion curves with simple high order derivatives. In *Computer Graphics Proceedings, Annual Conference Series* (1995), pp. 369–376. Proceedings of SIGGRAPH '95.

[24] KLOCK, F. Two moving coordinate frames for sweeping along a 3d trajectory. *Computer Aided Geometric Design 3* (1986).

[25] KUIPERS, J. *Quaternions and Rotation Sequences*. Princeton University Press, 1999.

[26] LAWSON, H. B., AND MICHELSOHN, M.-L. *Spin Geometry*. Princeton University Press, 1989.

[27] MA, H. *Curve and Surface Framing for Scientific Visualization and Domain Dependent Navigation*. PhD thesis, Indiana University, February 1996.

[28] MAX, N. L. DNA animation, from atom to chromosome. *Journal of Molecular Graphics 3*, 2 (1985), 69–71.

[29] MILNOR, J. *Topology from the Differentiable Viewpoint*. The University Press of Virginia, Charlottesville, 1965.

[30] MURTAGH, B. A., AND SAUNDER, M. A. MINOS 5.0 user's guide. Technical Report SOL 83-20, Dept. of Operations Research, Stanford University, 1983.

[31] NIELSON, G. M. Smooth interpolation of orientations. In *Computer Animation '93* (Tokyo, June 1993), N. Thalman and D. Thalman, Eds., Springer-Verlag, pp. 75–93.

[32] PLATT, J. C., AND BARR, A. H. Constraint methods for flexible models. In *Computer Graphics (SIGGRAPH '88 Proceedings)* (Aug. 1988), J. Dill, Ed., vol. 22, pp. 279–288.

[33] PLETINCKS, D. Quaternion calculus as a basic tool in computer graphics. *The Visual Computer 5*, 1 (1989), 2–13.

[34] RAMAMOORTHI, R., AND BARR, A. H. Fast construction of accurate quaternion splines. In *SIGGRAPH 97 Conference Proceedings* (Aug. 1997), T. Whitted, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 287–292. ISBN 0-89791-896-7.

[35] SCHLAG, J. Using geometric constructions to interpolate orientation with quaternions. In *Graphics Gems II*, J. Arvo, Ed. Academic Press, 1991, pp. 377–380.

[36] SHANI, U., AND BALLARD, D. H. Splines as embeddings for generalized cylinders. *Computer Vision, Graphics, and Image Processing 27* (1984), 129–156.

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

[38] SHOEMAKE, K. Animation with quaternions. Siggraph Course Lecture Notes, 1987.

[39] SHOEMAKE, K. Fiber bundle twist reduction. In *Graphics Gems IV*, P. Heckbert, Ed. Academic Press, 1994, pp. 230–236.

[40] STEENROD, N. *The Topology of Fibre Bundles*. Princeton University Press, 1951. Princeton Mathematical Series 14.

[41] TAIT, P. *An Elementary Treatise on Quaternions*. Cambridge University Press, 1890.

Quaternion Frame Approach to Streamline Visualization

Andrew J. Hanson, *Member, IEEE Computer Society,* and Hui Ma

***Abstract*—Curves in space are difficult to perceive and analyze, especially when they form dense sets as in typical 3D flow and volume deformation applications. We propose a technique that exposes essential properties of space curves by attaching an appropriate moving coordinate frame to each point, reexpressing that moving frame as a unit quaternion, and supporting interaction with the resulting quaternion field. The original curves in three-space are associated with piecewise continuous four-vector quaternion fields, which map into new curves lying in the unit three-sphere in four-space. Since four-space clusters of curves with similar moving frames occur independently of the curves' original proximity in three-space, a powerful analysis tool results. We treat two separate moving-frame formalisms, the Frenet frame and the parallel-transport frame, and compare their properties. We describe several flexible approaches for interacting with and exploiting the properties of the four-dimensional quaternion fields.**

***Index Terms*—Quaternion, Frenet frame, orientation frame.**

I. Introduction

We introduce techniques and tools for visualizing streamline data that are based on the differential geometry of 3D space curves. Intrinsic properties of space curves give rise to scalar fields over the curves such as the curvature and torsion. A moving coordinate frame on a curve is a tensor field that is equivalent to a quaternion field; either may be understood as the solution to a set of differential equations driven by the intrinsic scalar fields.

Our fundamental thesis is that quaternion frame coordinates are useful for exposing the similarities and differences of sets of streamlines. Good analytic and visual measures for revealing similarities of curve shapes are rare. Because of the existence of a uniform distance measure in the quaternion space that we use, orientation similarities in the evolution of flow fields appear automatically in meaningful spatial groups. Identification of these similarities is useful for applications such as finding repeating patterns and related curve shapes, both on single curves and within large collections of curves. Conversely, if a large set of nearly-identical curves contains a small number of significant curves that differ from their neighbors due to subtle changes in their frame orientations, our method will distinguish them.

We study two distinct moving coordinate frames that may be assigned to curves in three-space. One is the classic Frenet frame, also called the Frenet-Serret frame (see, e.g., [9], [10]), which is defined locally by the tangent, normal and binormal at each point of each curve; the other is the parallel-transport frame (see, e.g., Bishop [4]), which retains the tangent vector, but uses a nonlocal approach borrowed from the parallel-transport methods of differential geometry to compute the frame components in the plane perpendicular to the curve. All such frames can be recast into quaternion frame coordinates.

Orientation spaces and their relationship to quaternions are described in Altmann [2]; an interesting approach to the visualization of the properties of quaternions was recently given by Hart, Francis, and Kauffman [19]. Systematic approaches for representing clusters of orientations in 3D spaces of angles have been suggested, for example, by Alpern et al. [1]. Among previous approaches to visualizing the geometry of space curves, we note the work of Gray [7], [10] which exploits the curvature and torsion scalar fields on a curve for visualization purposes; this method extends naturally to higher-dimensional manifolds with well-defined local curvature. We will give some examples of the application of curvature and torsion fields for completeness here, but will not pursue this approach in detail.

The use of quaternion frames in a 4D display was proposed as a visualization technique for stream manifold characteristics in Hanson and Ma [17]. The current article is based on the concepts of the latter work, and includes additional results on the comparative properties of the Frenet and parallel-transport frames, as well as further work on interactive methods.

II. The Differential Geometry of Space Curves

Dense families of space curves can be generated by many applications, ranging from time-dependent particle flow fields, to static streamlines generated by integrating a volume vector field, to deformations of a solid coordinate grid. Our fundamental approach singles out space curves, although variations could be used to treat individual point frames (see [1]), stream surfaces (see [20]), and orientation differences (which are themselves orientation fields) as well. Thus, we begin with the properties of a curve $\vec{\mathbf{x}}(t)$ in 3D space parameterized by the unnormalized arc length t. If $\vec{\mathbf{x}}(t)$ is once-differentiable, then the tangent vector at any point is

$$\vec{\mathbf{T}}(t) = \frac{\vec{\mathbf{x}}'(t)}{\|\vec{\mathbf{x}}'(t)\|}.$$

The standard arc-length differential is typically expressed as

$$v(t)^2 = \left(\frac{ds}{dt}\right)^2 = \left(\frac{d\vec{\mathbf{x}}(t)}{dt}\cdot\frac{d\vec{\mathbf{x}}(t)}{dt}\right) = \|\vec{\mathbf{x}}'(t)\|^2.$$

A.J. Hanson and H. Ma are with the Department of Computer Science, Indiana University, Bloomington, IN 47405; e-mail: hanson@cs.indiana.edu.
IEEECS Log Number V95009.

1077-2626/95$04.00 © 1995 IEEE

In practice, we never have smooth curves in numerical applications, but only piecewise linear curves that are presumed to be approximations to differentiable curves; thus we might typically take, for a curve given by the set of points $\{\vec{\mathbf{x}}_i\}$,

$$\vec{\mathbf{T}}_i = \frac{\vec{\mathbf{x}}_{i+1} - \vec{\mathbf{x}}_i}{\|\vec{\mathbf{x}}_{i+1} - \vec{\mathbf{x}}_i\|},$$

or any corresponding formula with additional sampling points and desirable symmetries. We use a five-point formula to get a smoother result; one could also produce finer intermediate states by spline interpolation.

If the curve is locally straight, i.e., $\vec{\mathbf{x}}''(t) = 0$ or $\vec{\mathbf{T}}_{i+1} = \vec{\mathbf{T}}_i$, then there is no locally-determinable coordinate frame component in the plane normal to $\vec{\mathbf{T}}$; a nonlocal definition must be used to decide on the remainder of the frame once $\vec{\mathbf{T}}$ is determined. Below, we formulate our two alternate coordinate frames, one of which, the Frenet frame, is completely local, but is indeterminable where the curve is locally straight, and the other of which, the parallel transport frame, is defined everywhere but depends on a numerical integration over the whole curve.

A. Frenet Frames

The *Frenet frame* (see, e.g., [9], [10]) is defined as follows: If $\vec{\mathbf{x}}(t)$ is any thrice-differentiable space curve, its tangent, binormal, and normal vectors at a point on the curve are given by

$$\begin{aligned} \vec{\mathbf{T}}(t) &= \frac{\vec{\mathbf{x}}'(t)}{\|\vec{\mathbf{x}}'(t)\|} \\ \vec{\mathbf{B}}(t) &= \frac{\vec{\mathbf{x}}'(t) \times \vec{\mathbf{x}}''(t)}{\|\vec{\mathbf{x}}'(t) \times \vec{\mathbf{x}}''(t)\|} \\ \vec{\mathbf{N}}(t) &= \vec{\mathbf{B}}(t) \times \vec{\mathbf{T}}(t). \end{aligned} \tag{1}$$

We illustrate this standard frame configuration in Fig. 1. When the second derivative vanishes on some interval, the Frenet frame is temporarily undefined, as illustrated in Fig. 2. Attempts to work around this problem involve various heuristics (see, e.g., [5], [24]).

The Frenet frame obeys the following differential equation in the parameter t,

$$\begin{bmatrix} \vec{\mathbf{T}}'(t) \\ \vec{\mathbf{N}}'(t) \\ \vec{\mathbf{B}}'(t) \end{bmatrix} = v(t) \begin{bmatrix} 0 & \kappa(t) & 0 \\ -\kappa(t) & 0 & \tau(t) \\ 0 & -\tau(t) & 0 \end{bmatrix} \begin{bmatrix} \vec{\mathbf{T}}(t) \\ \vec{\mathbf{N}}(t) \\ \vec{\mathbf{B}}(t) \end{bmatrix} \tag{2}$$

where $v(t) = \|\mathbf{x}'(t)\|$ is the scalar magnitude of the curve derivative, $\kappa(t)$ is the scalar curvature, and $\tau(t)$ is the torsion. These quantities can in principle be calculated in terms of the parameterized or numerical local values of $\vec{\mathbf{x}}(t)$ and its first three derivatives as follows:

$$\begin{aligned} \kappa(t) &= \frac{\|\vec{\mathbf{x}}'(t) \times \vec{\mathbf{x}}''(t)\|}{\|\vec{\mathbf{x}}'(t)\|^3} \\ \tau(t) &= \frac{(\vec{\mathbf{x}}'(t) \times \vec{\mathbf{x}}''(t)) \cdot \vec{\mathbf{x}}'''(t)}{\|\vec{\mathbf{x}}'(t) \times \vec{\mathbf{x}}''(t)\|^2}. \end{aligned} \tag{3}$$

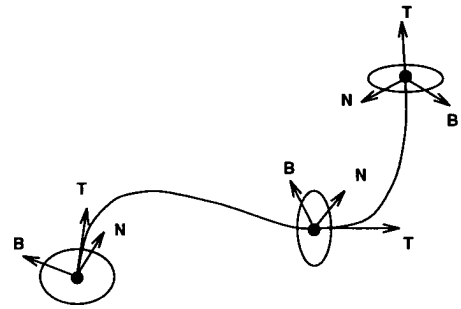


Fig. 1. The triad of orthogonal axes forming the Frenet frame for a curve with nonvanishing curvature.

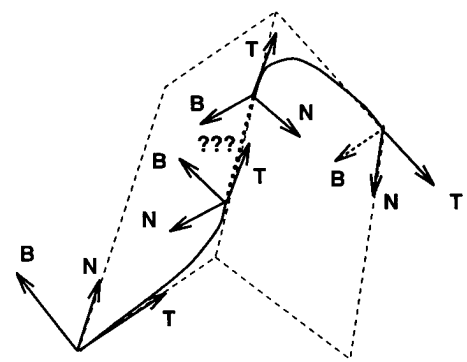


Fig. 2. The triad of orthogonal axes forming the Frenet frame for a curve with vanishing curvature on an interval; the frame is undefined on the interval.

If we are given a nonvanishing curvature and a torsion as smooth functions of t, we can theoretically integrate the system of equations to find the unique numerical values of the corresponding space curve $\vec{\mathbf{x}}(t)$ (up to a rigid motion).

B. Parallel Transport Frames

Bishop [4] noted that, while the Frenet frame has the advantage of consistent local computability at all points on a curve except those with vanishing second derivative, there is another natural frame, the *parallel transport frame*, that is well-defined everywhere; the distinguishing feature of the parallel transport frame is that it is essentially the solution to a differential equation, and thus depends on the initial conditions and is subject to numerical error for long curves. Operational methods of defining such frames have been noted (see, e.g., [5], [24]) but the underlying mathematical basis was not elaborated.

Geometrically, the parallel transport frame derives its name from the fact that it corresponds to the notion of moving a vector around a curved manifold in such a way that it remains as parallel to itself as possible. Its mathematical properties follow from the observation that, while $\vec{\mathbf{T}}(t)$ for a given curve model is unique, we may choose any convenient arbitrary basis $(\vec{\mathbf{N}}_1(t), \vec{\mathbf{N}}_2(t))$ for the remainder of the frame, so long as it is in the plane perpendicular to $\vec{\mathbf{T}}(t)$ at each point. If the derivatives of $(\vec{\mathbf{N}}_1(t), \vec{\mathbf{N}}_2(t))$ depend only on $\vec{\mathbf{T}}(t)$ and not each other, we can make $\vec{\mathbf{N}}_1(t)$ and $\vec{\mathbf{N}}_2(t)$ vary smoothly throughout the path regardless of the curvature. We may, therefore, choose the alternative frame equations

$$\begin{bmatrix} \vec{\mathbf{T}}' \\ \vec{\mathbf{N}}_1' \\ \vec{\mathbf{N}}_2 \end{bmatrix} = v \begin{bmatrix} 0 & k_1 & k_2 \\ -k_1 & 0 & 0 \\ -k_2 & 0 & 0 \end{bmatrix} \begin{bmatrix} \vec{\mathbf{T}} \\ \vec{\mathbf{N}}_1 \\ \vec{\mathbf{N}}_2 \end{bmatrix}, \tag{4}$$

illustrated in Fig. 3 for a curve with vanishing curvature on a segment. One can show that [4]

$$\kappa(t) = \left((k_1)^2 + (k_2)^2\right)^{1/2}$$

$$\theta(t) = \arctan\left(\frac{k_2}{k_1}\right)$$

$$\tau(t) = \frac{d\theta(t)}{dt},$$

so that k_1 and k_2 effectively correspond to a Cartesian coordinate system for the polar coordinates κ, θ with $\theta = \int \tau(t)dt$. A fundamental ambiguity in the parallel transport frame compared to the Frenet frame thus arises from the arbitrary choice of an integration constant for θ, which disappears from τ due to the differentiation.

A numerical method for computing the parallel transport frame with the desired properties is the following. Given a frame at $\vec{\mathbf{x}}_{i-1}$, compute two neighboring tangents $\vec{\mathbf{T}}_i$ and $\vec{\mathbf{T}}_{i-1}$ and their unit vectors $\hat{\mathbf{T}}_i = \vec{\mathbf{T}}_i / \|\vec{\mathbf{T}}_i\|$; find the angle $\theta = \arccos(\hat{\mathbf{T}}_i \cdot \hat{\mathbf{T}}_{i-1})$ between them and the perpendicular to the plane of the tangents given by $\vec{\mathbf{V}} = (\hat{\mathbf{T}}_{i-1} \times \hat{\mathbf{T}}_i)$; finally, rotate the frame at $\vec{\mathbf{x}}_{i-1}$ by θ about $\hat{\mathbf{V}}$ to get the frame at point $\vec{\mathbf{x}}_i$. Either 3D vector rotation or rotation by quaternion multiplication can be used to effect the rotation.

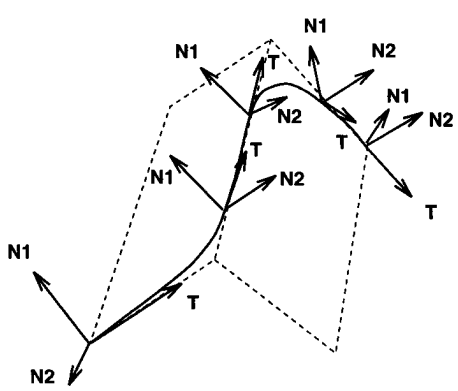


Fig. 3. The parallel-transport curve frame for the curve of Fig. 2 [4]. This frame, unlike the Frenet frame in Fig. 2, is continuous along the "roof peak" where the curvature vanishes.

Just as for the Frenet frame, one can begin with a curve $\vec{\mathbf{x}}(t)$ and an initial frame, or a pair of functions $(k_1(t), k_2(t))$ and an initial frame, or a frame over the entire curve, and then integrate where needed to compute the missing variables. It is also worthwhile noting that $(k_1(t), k_2(t))$ form a two-dimensional Cartesian vector field at each point of the curve, and thus allow a natural alternate characterization to Gray's (κ, τ) curve properties [7], [10].

III. Theory of Quaternion Frames

It is awkward to represent moving frames visually in high-density data because a frame consists of three 3D vectors, or nine components, yet it has only three independent degrees of freedom. Some approaches to representing these degrees of freedom in a three-dimensional space were suggested by Alpern et al. [1]. We propose instead to systematically exploit the representation of 3D orientation frames in four-dimensions using equivalent unit quaternions that correspond, in turn, to points on the three-sphere (see, e.g., [25]). A collection of oriented frames such as those of a crystal lattice can thus be represented by mapping their orientations to a point set in the 4D quaternion space. The moving frame of a 3D space curve can be transformed into a path in quaternion space corresponding pointwise to the 3D space curve.

The quaternion representation of rotations reexpressing a moving frame of a 3D space curve is an elegant unit four-vector field over the curve; the resulting quaternion frames can be displayed as curves in their own right, or can be used in combination with other methods to enrich the display of each 3D curve, e.g., by assigning a coded display color representing a quaternion component.

Properties. A quaternion frame is a unit-length four-vector $q = (q_0, q_1, q_2, q_3) = (q_0, \vec{q})$ that corresponds to exactly one 3D coordinate frame and is characterized by the following properties:

- *Unit Norm.* If we define the inner product of two quaternions as

  $$q \cdot p = q_0 p_0 + q_1 p_1 + q_2 p_2 + q_3 p_3,$$

  then the components of a unit quaternion obey the constraint

  $$q \cdot q = (q_0)^2 + (q_1)^2 + (q_2)^2 + (q_3)^2 = 1, \tag{5}$$

  and therefore lie on S^3, the three-sphere, which we will typically represent as embedded in four-dimensional Euclidean space R^4.
- *Multiplication Rule.* The quaternion product of two quaternions q and p, which we write as $q * p$, takes the form

  $$\begin{bmatrix} [q*p]_0 \\ [q*p]_1 \\ [q*p]_2 \\ [q*p]_3 \end{bmatrix} = \begin{bmatrix} q_0p_0 - q_1p_1 - q_2p_2 - q_3p_3 \\ q_0p_1 + q_1p_0 + q_2p_3 - q_3p_2 \\ q_0p_2 + q_2p_0 + q_3p_1 - q_1p_3 \\ q_0p_3 + q_3p_0 + q_1p_2 - q_2p_1 \end{bmatrix}$$

  This rule is isomorphic to multiplication in the group SU(2), the double covering of the ordinary 3D rotation group SO(3). If two quaternions a and b are transformed by multiplying them by the same quaternion q, their inner product $a \cdot b$ transforms as

  $$(q * a) \cdot (q * b) = (a \cdot b)\,(q \cdot q)$$

  and so is invariant if q is a unit quaternion.
- *Mapping to 3D rotations.* Every possible 3D rotation R (a 3×3 orthogonal matrix) can be constructed from either of two related quaternions, $q = (q_0, q_1, q_2, q_3)$ or $-q = (-q_0, -q_1, -q_2, -q_3)$, using the quadratic relationship

$$R = \begin{bmatrix} Q(+--) & D^-(123) & D^+(312) \\ D^+(123) & Q(-+-) & D^-(231) \\ D^-(312) & D^+(231) & Q(--+) \end{bmatrix} \tag{6}$$

where $Q(\pm\pm\pm) = q_0^2 \pm q_1^2 \pm q_2^2 \pm q_3^2$ and $D^{\pm}(ijk) = 2q_iq_j \pm 2q_0q_k$.

- *Rotation Correspondence.* When we substitute $q = (\cos\frac{\theta}{2}, \hat{\mathbf{n}}\sin\frac{\theta}{2})$, into (6), where $\hat{\mathbf{n}}\cdot\hat{\mathbf{n}} = 1$ is a unit three-vector lying on the two-sphere S^2, $R(\theta, \hat{\mathbf{n}})$ becomes the standard matrix for a rotation by θ in the plane perpendicular to $\hat{\mathbf{n}}$; the quadratic form ensures that the two distinct unit quaternions q and $-q$ in S^3 correspond to the *same* SO(3) rotation.

A. Quaternion Frenet Frames

All 3D coordinate frames can be expressed in the form of quaternions using (6). If we assume the columns of (6) are the vectors $((\vec{\mathbf{T}}, \vec{\mathbf{N}}, \vec{\mathbf{B}})$, respectively, one can show from (2) that $[q'(t)]$ takes the form (see [12])

$$\begin{bmatrix} q_0' \\ q_1' \\ q_2' \\ q_3' \end{bmatrix} = \frac{v}{2}\begin{bmatrix} 0 & -\tau & 0 & -\kappa \\ \tau & 0 & \kappa & 0 \\ 0 & -\kappa & 0 & \tau \\ \kappa & 0 & -\tau & 0 \end{bmatrix} \cdot \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}. \tag{7}$$

This equation has the following key properties:

- The matrix on the right hand side is antisymmetric, so that $q(t) \cdot q'(t) = 0$ by construction. Thus, all unit quaternions remain unit quaternions as they evolve by this equation.
- The number of equations has been reduced from nine coupled equations with six orthonormality constraints to four coupled equations incorporating a single constraint that keeps the solution vector confined to the three-sphere.

We verify that the matrices

$$A = \begin{bmatrix} q_0 & q_1 & -q_2 & -q_3 \\ q_3 & q_2 & q_1 & q_0 \\ -q_2 & q_3 & -q_0 & q_1 \end{bmatrix}$$

$$B = \begin{bmatrix} -q_3 & q_2 & q_1 & -q_0 \\ q_0 & -q_1 & q_2 & -q_3 \\ q_1 & q_0 & q_3 & q_2 \end{bmatrix}$$

$$C = \begin{bmatrix} q_2 & q_3 & q_0 & q_1 \\ -q_1 & -q_0 & q_3 & q_2 \\ q_0 & -q_1 & -q_2 & q_3 \end{bmatrix}$$

explicitly reproduce (2),

$$[A]\cdot[q'] = \vec{\mathbf{T}}' = v\kappa\vec{\mathbf{N}}$$
$$[B]\cdot[q'] = \vec{\mathbf{N}}' = -v\kappa\vec{\mathbf{T}} + v\tau\vec{\mathbf{B}}$$
$$[C]\cdot[q'] = \vec{\mathbf{B}}' = -v\tau\vec{\mathbf{N}},$$

where we have applied (7) to get the right-hand terms.

Just as the Frenet equations may be integrated to generate a unique moving frame with its space curve for nonvanishing $\kappa(t)$, we may integrate the much simpler quaternion equations (7).

B. Quaternion Parallel Transport Frames

Similarly, a parallel-transport frame system given by (4) with $(\vec{\mathbf{N}}_1, \vec{\mathbf{T}}, \vec{\mathbf{N}}_2)$ (in that order) corresponding to the columns of (6) is completely equivalent to the following parallel-transport quaternion frame equation for $[q'(t)]$:

$$\begin{bmatrix} q_0' \\ q_1' \\ q_2' \\ q_3' \end{bmatrix} = \frac{v}{2}\begin{bmatrix} 0 & -k_2 & 0 & k_1 \\ k_2 & 0 & -k_1 & 0 \\ 0 & k_1 & 0 & k_2 \\ -k_1 & 0 & -k_2 & 0 \end{bmatrix} \cdot \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{8}$$

where antisymmetry again guarantees that the quaternions remain constrained to the unit three-sphere. The correspondence to (4) is verified as follows:

$$[B]\cdot[q'] = \vec{\mathbf{T}}' = vk_1\vec{\mathbf{N}}_1 + vk_2\vec{\mathbf{N}}_2$$
$$[A]\cdot[q'] = \vec{\mathbf{N}}_1' = -vk_1\vec{\mathbf{T}}$$
$$[C]\cdot[q'] = \vec{\mathbf{N}}_2' = -vk_2\vec{\mathbf{T}}.$$

IV. Assigning Smooth Quaternion Frames

Given a particular curve, we are next faced with the task of assigning quaternion values to whatever moving frame sequence we have chosen.

A. Assigning Quaternions to Frenet Frames

The Frenet frame equations are pathological, for example, when the curve is perfectly straight for some distance or when the curvature vanishes momentarily. Thus, real numerical data for space curves will frequently exhibit behaviors that make the assignment of a smooth Frenet frame difficult, unstable, or impossible. In addition, since any given 3×3 orthogonal matrix corresponds to two quaternions that differ in sign, methods of deriving a quaternion from a Frenet frame are intrinsically ambiguous. Therefore, we prescribe the following procedure for assigning smooth quaternion Frenet frames to points on a space curve:

- Select a numerical approach to computing the tangent $\vec{\mathbf{T}}$ at a given curve point $\vec{\mathbf{x}}$; this typically depends on the chosen curve model and the number of points one wishes to sample.
- Compute the remaining numerical derivatives at a given point and use those to compute the Frenet frame according to (1). If any critical quantities vanish, tag the frame as undefined (or as needing a heuristic fix).
- Check the dot product of the previous binormal $\mathbf{B}(t)$ with the current value; if it is near zero, choose a correction procedure to handle this singular point. Among the correction procedures we have considered are: 1) simply jump discontinuously to the next frame to indicate the presence of a point with very small curvature; 2) create

an interpolating set of points and perform a geodesic interpolation [25]; or 3) deform the curve slightly before and after the singular point to "ease in" with a gradual rotation of the frame or apply an interpolation heuristic (see, e.g., [24]). Creating a jump in the frame assignment is our default choice, since it does not introduce any new information.

- Apply a suitable algorithm such as that of Shoemake [25] to compute a candidate for the quaternion corresponding to the Frenet frame.
- If the 3×3 Frenet frame is smoothly changing, make one last check on the 4D inner product of the quaternion frame with its own previous value: If there is a sign change, choose the opposite sign to keep the quaternion smoothly changing (this will have no effect on the corresponding 3×3 Frenet frame). If this inner product is near zero instead of ± 1, you have detected a radical change in the Frenet frame which should have been noticed in the previous tests.
- If the space curves of the data are too coarsely sampled to give the desired smoothness in the quaternion frames, but are still close enough to give consistent qualitative behavior, one may choose to smooth out the intervening frames using the desired level of recursive slerping [23], [25] to get smoothly splined intermediate quaternion frames.

In Fig. 4, we plot an example of a torus knot, a smooth space curve with everywhere nonzero curvature, together with its associated Frenet frames, its quaternion frame values, and the path of its quaternion frame field projected from four-space. Fig. 5 plots the same information, but this time for a curve with a discontinuous frame that flips too quickly at a zero-curvature point. This space curve has two planar parts drawn as though on separate pages of a partly-open book and meeting smoothly on the "crack" between pages. We see the obvious jump in the Frenet and quaternion frame graphs at the meeting point; if the two curves are joined by a long straight line, the Frenet frame is ambiguous and is essentially undefined in this segment. Rather than invent an interpolation, we generally prefer to use the parallel transport method described next.

B. Assigning Quaternions to Parallel Transport Frames

In order to determine the quaternion frames of an individual curve using the parallel transport method, we follow a similar, but distinct, procedure:

- Select a numerical approach to assigning a tangent at a given curve point as usual.
- Assign an initial reference orientation to each curve in the plane perpendicular to the initial tangent direction. The entire set of frames will be displaced from the origin in quaternion space by the corresponding value of this initial orientation matrix, but the shape of the entire curve will be the same regardless of the initial choice. This choice is intrinsically ambiguous and application dependent. However, one appealing strategy is to base the initial frame on the first well-defined Frenet frame, and then proceed from there using the parallel-transport frame evolution; this guarantees that identical curves have the same parallel-transport frames.

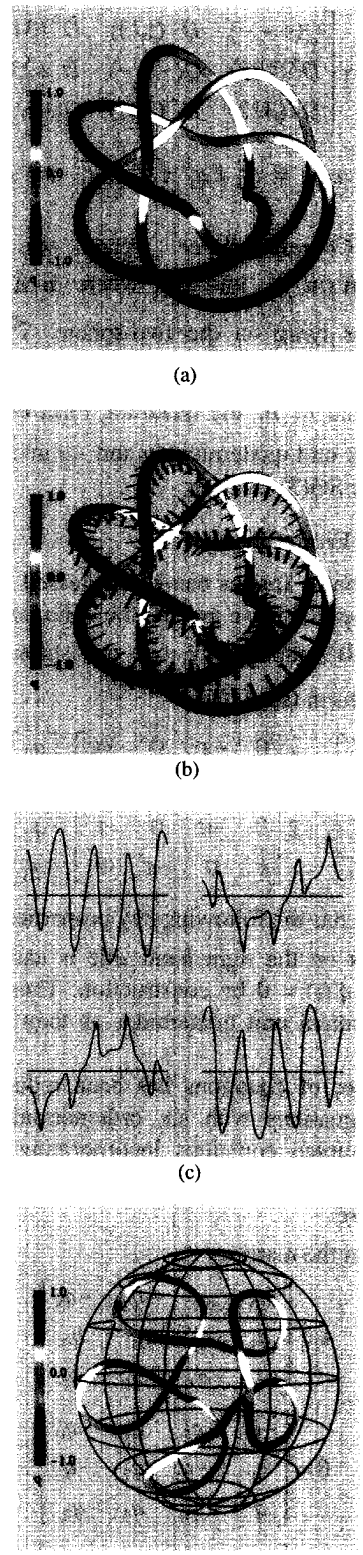


Fig. 4. (a) Projected image of a 3D (3, 5) torus knot. (b) Selected Frenet frame components displayed along the knot. (c) The corresponding smooth quaternion frame components. (d) The path of the quaternion frame components in the three-sphere projected from four-space. Color scales indicate the 0th component of the curve's four-vector frame (upper left graph in (c)).

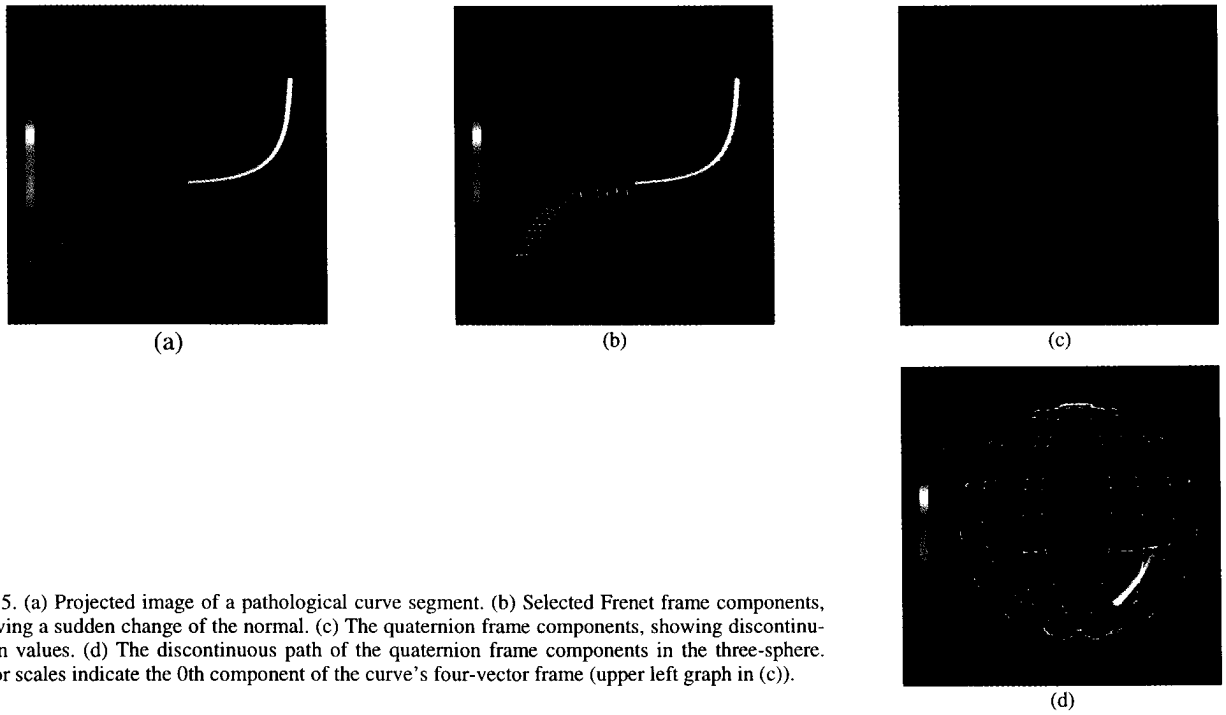


Fig. 5. (a) Projected image of a pathological curve segment. (b) Selected Frenet frame components, showing a sudden change of the normal. (c) The quaternion frame components, showing discontinuity in values. (d) The discontinuous path of the quaternion frame components in the three-sphere. Color scales indicate the 0th component of the curve's four-vector frame (upper left graph in (c)).

- Compute the angle between successive tangents, and rotate the frame by this angle in the plane of the two tangents to get the next frame value.
- If the curve is straight, the algorithm automatically makes no changes.
- Compute a candidate quaternion representation for the frame, applying consistency conditions as needed.

Note that the initial reference orientation and all discrete rotations can be represented *directly in terms of quaternions*, and thus quaternion multiplication can be used directly to apply frame rotations. Local consistency is then automatic.

An example is provided in Fig. 6, which shows the parallel transport analog of Fig. 4 for a torus knot. Fig. 7 is the parallel transport analog of the pathological case in Fig. 5, but this time the frame is continuous when the curvature vanishes.

V. Examples

We next present some typical examples of streamline data represented using the basic geometric properties we have described. Each data set is rendered in the following alternative modes: 1) as a 3D Euclidean space picture, pseudocolored by curvature value; 2) as a 3D Euclidean space picture, pseudocolored by torsion value; 3) as a four-vector quaternion Frenet frame field plotted in the three-sphere; and 4) as a four-vector quaternion parallel transport frame field plotted in the three-sphere.

- *Fig.* 8. A complicated set of streamlines derived from twisting a solid elastic Euclidean space as part of the process of tying a topological knot.
- *Fig.* 9. An AVS-generated streamline data set; the flow is obstructed somewhere in the center, causing sudden jumps of the streamlines in certain regions.

While our focus in this paper is specifically on the frames of space curves, we remark that collections of frames of isolated points, frames on stream surfaces [20], and volumetric frame fields could also be represented using a similar mapping into quaternion space.

VI. Visualization Methods

Once we have calculated the quaternion frames, the curvature, and the torsion for a point on the curve, we have a family of tensor and scalar quantities that we may exploit to expose the intrinsic properties of a single curve. Furthermore, and probably of greater interest, we also have the ability to make visual comparisons of the similarities and differences among families of neighboring space curves.

The moving frame field of a set of streamlines is potentially a rich source of detailed information about the data. However, the nine-component frame is unsuitable for direct superposition on dense data due to the high clutter resulting when its three orthogonal three-vectors are displayed; direct use of the frame is only practical at very sparse intervals, which prevents the viewer from grasping important structural details and changes at a glance. Displays based on 3D angular coordinates are potentially useful, but lack metric uniformity [1].

The four-vector quaternion frame is potentially a more informative and flexible basis for frame visualizations; below, we discuss several alternative approaches to the exploitation of quaternion frames for data consisting of families of smooth curves.

A. Direct Three-Sphere Plot of Quaternion Frame Fields

We now repeat the crucial observation: *For each 3D space curve, the moving quaternion frames define completely new 4D space curves lying on the unit three-sphere embedded in 4D Euclidean space.*

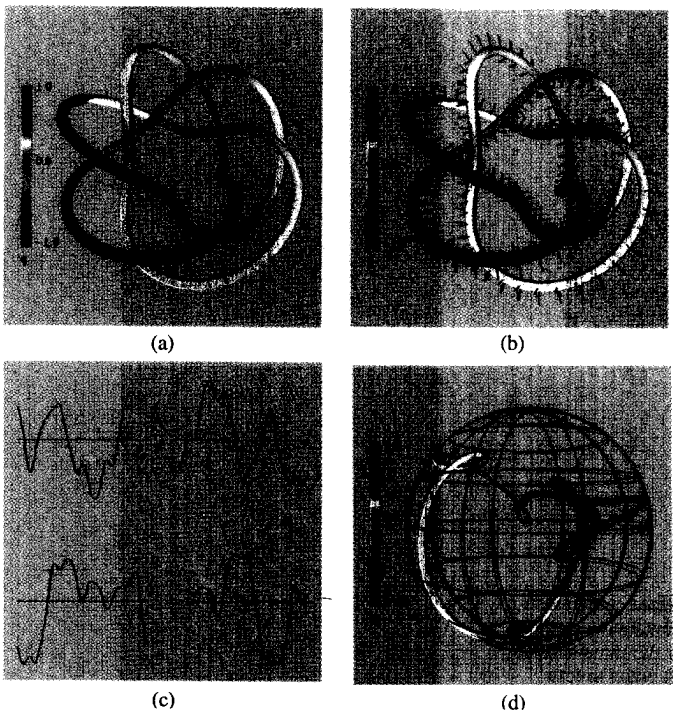


Fig. 6. (a) Projected image of a 3D (3, 5) torus knot. (b) Selected parallel-transport frame components displayed along knot. (c) The corresponding smooth quaternion frame components. (d) The path of the quaternion frame components in the three-sphere projected from four-space. Color scales indicate the 0th component of the curve's four-vector frame (upper left graph in (c)).

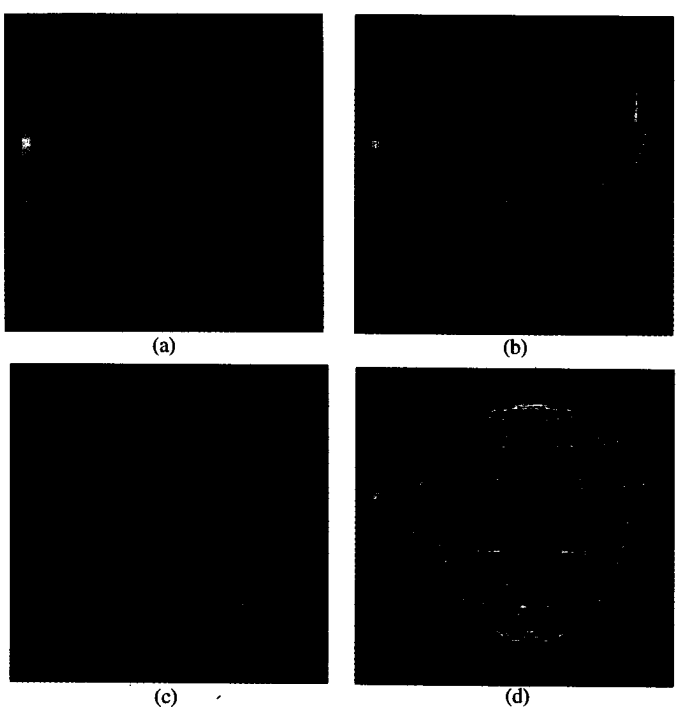


Fig. 7. (a) Projected image of a pathological curve segment. (b) Selected parallel transport frame components, showing smooth change of the normal. (c) The quaternion frame components, showing continuity in values. (d) The continuous path of the quaternion frame components in the three-sphere. Color scales indicate the 0th component of the curve's four-vector frame (upper left graph in (c)).

These curves can have *entirely different geometry* from the original space curve, since distinct points on the curve correspond to distinct orientations. Families of space curves with exactly the same shape will map to the *same* quaternion curve, while curves that fall away from their neighbors will stand out distinctly in the three-sphere plot. Regions of vanishing curvature will show up as discontinuous gaps in the otherwise continuous quaternion Frenet frame field curves, but will be well-behaved in the quaternion parallel transport frame fields. Straight 3D lines will of course map to single points in quaternion space, which may require special attention in the display.

Figs. 4d and 5d present elementary examples of the three-sphere plot for the Frenet frame, while Figs. 6d and 7d illustrate the parallel transport frame. Figs. 8c, 8d, 9c, and 9d present more realistic examples.

The quaternion frame curves displayed in these plots are 2D projections of two overlaid 3D solid balls corresponding to the "front" and "back" hemispheres of S^3. The three-sphere is projected from 4D to 3D along the 0th axis, so the "front" ball has points with $0 \le q_0 \le +1$, and the "back" ball has points with $-1 \le q_0 < 0$. The q_0 values of the frame at each point can be displayed as shades of gray or pseudocolor. In the default view projected along the q_0-axis, points that are projected from 4D to the 3D origin are in fact identity frames, since unit length of q requires $q = (\pm 1, 0, 0, 0)$ at these points. In Fig. 10, we show a sequence of views of the same quaternion curves from different 4D viewpoints using parallel projection; Fig. 11 shows the additional contrast in structure sizes resulting from a 4D perspective projection.

B. Scalar Geometric Fields

Gray [7], [10] has advocated the use of curvature and torsion-based color mapping to emphasize the geometric properties of single curves such as the torus knot. Since this information is trivial to obtain simultaneously with the Frenet frame, we also offer the alternative of encoding the curvature and torsion as scalar fields on a volumetric space populated either sparsely or densely with streamlines; examples are shown in Figs. 8a, 8b, 9a, and 9b.

C. Similarity Measures for Quaternion Frames

Quaternion frames carry with them a natural geometry that may be exploited to compute meaningful similarity measures. Rather than use the Euclidean distance in four-dimensional Euclidean space R^4, one may use the magnitude of the four-vector scalar product of unit quaternions

$$d(q, p) = |q \cdot p| = |\, q_0 p_0 + q_1 p_1 + q_2 p_2 + q_3 p_3 \,|,$$

or the corresponding angle,

$$\theta(q, p) = \arccos\,(d(q, p)),$$

which is the angular difference between the two 4D unit vectors and a natural measure of great-circle arc-length on S^3. Choosing this as a distance measure results in a quantity that is invariant under 4D rotations, invariant under 3D rotations represented by quaternion multiplication, and is also insensitive to the sign ambiguity in the quaternion representation for a given frame. Thus it may be used as a quantitative measure of the similarity of any two 3D frames. This is a natural way to compare either successive frames on a single streamline or pairs of frames on different streamlines.

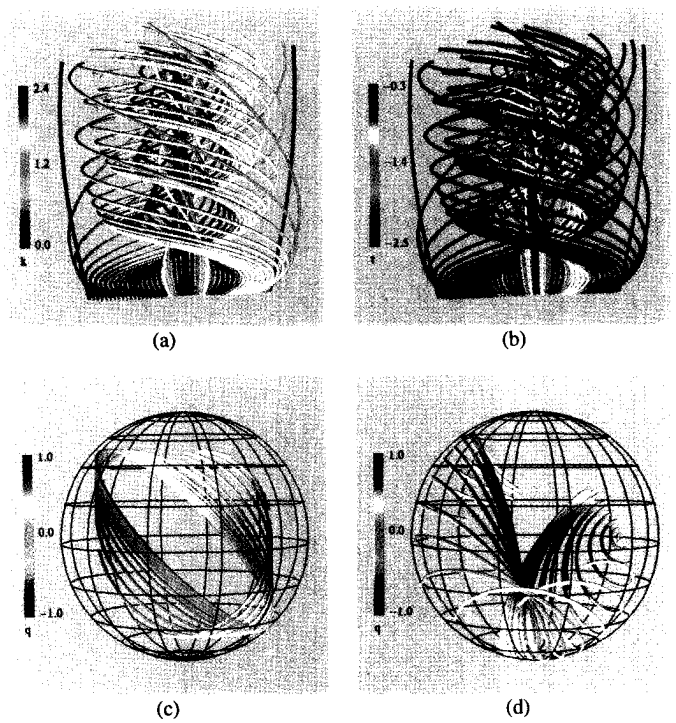


Fig. 8. (a) Deformed volume related to tying a knot, color coded by curvature. (b) Deformed volume related to tying a knot, color coded by torsion. (c) The corresponding quaternion field paths for the Frenet frames. (d) The corresponding quaternion field paths for the parallel transport frames. The color code is keyed to the value of the quaternion component q_0 that is collapsed in the projection from 4D to 3D.

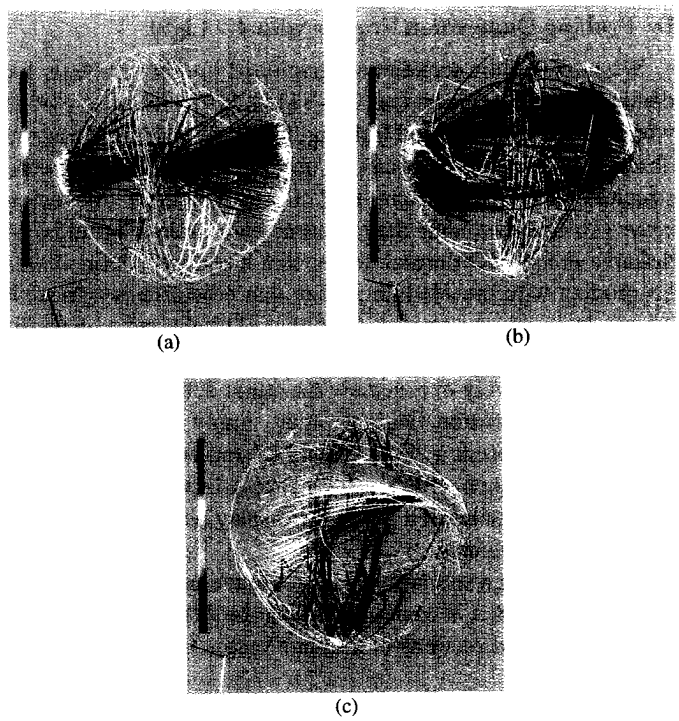


Fig.10. Successive frames in a 4D rotation of the parallel projected three-sphere display of the quaternion fields for a set of streamline data.

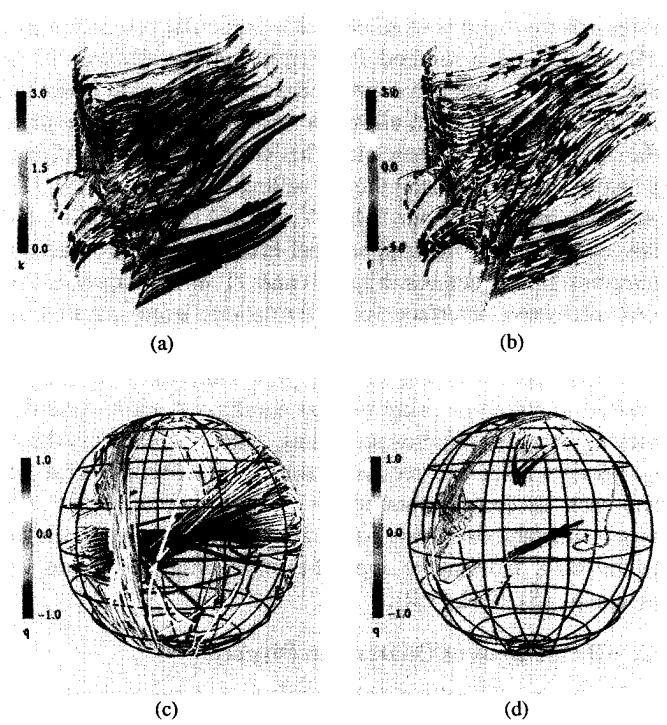


Fig. 9. (a) Vector field streamlines, color coded by curvature. (b) Vector field streamlines, color coded by torsion. (c) The corresponding quaternion field paths for the Frenet frames. (d) The corresponding quaternion field paths for the parallel transport frames. The color code is keyed to the value of the quaternion component q_0 that is collapsed in the projection from 4D to 3D. (AVS demonstration data set.)

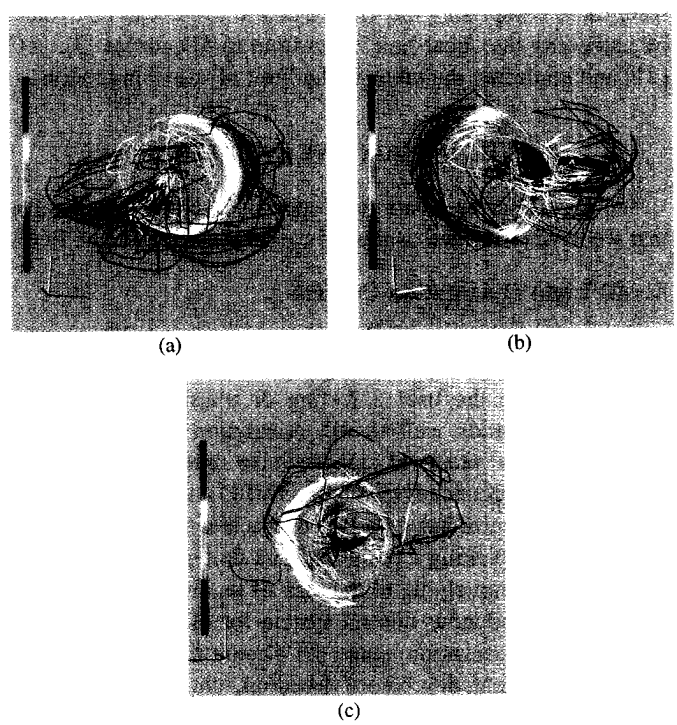


Fig. 11. Successive frames in a 4D rotation of the polar projected three-sphere display of the quaternion fields for a set of streamline data.

D. Probing Quaternion Frames with 4D Light

We next explore techniques developed in other contexts for dealing with 4D objects (see [14], [15], [16]). In our previous work on 4D geometry and lighting, the critical element was the observation that 4D light can be used as a probe of geometric structure provided we can find a way (such as thickening curves or surfaces until they become true three-manifolds) to define a unique 4D normal vector that has a well-defined scalar product with the 4D light; when that objective is achieved, we can interactively employ a moving 4D light and a generalization of the standard illumination equations to produce images that selectively expose new structural details.

Given a quaternion field, we may simply select a 4D unit vector L to represent a "light direction" and employ a standard lighting model such as $I(t) = L \cdot q(t)$ to select individual components of the quaternion fields for display using pseudo-color coding for the intensity.

Fig. 12 shows a streamline data set rendered by computing a pseudo-color index at each point using the 4D lighting formula and varying the directions of the four-vector L.

E. True 4D Illumination

The quaternion curves in 4D may also be displayed in an entirely different mode by thickening them to form three-manifolds using the method of Hanson and Heng [15], [16] and replacing $q(t)$ in the 4D lighting formula and its specular analogs by the 4D normal vector for each volume element or vertex. The massive expense of volume rendering the resulting solid tubes comprising the 4D projection to 3D can be avoided by extending the "bear-hair" algorithm to 4D curves [3], [14], [21] and rendering the tubes in the limit of vanishing radius.

VII. Interactive Interfaces

We next describe a variety of specific interactive techniques that we have examined as tools for exploring quaternion fields.

A. 4D Light Orientation Control

Direct manipulation of 3D orientation using a 2D mouse is typically handled using a rolling ball [11] or virtual sphere [6] method to give the user a feeling of physical control. This philosophy extends well to 4D orientation control (see [8], [13]), giving a practical approach to interacting with the visualization approaches of Sections VI.D and VI.E.

A 3D unit vector has only two degrees of freedom, and so is determined by picking a point within a unit circle to determine the direction uniquely up to the sign of its view-direction component. The analogous control system for 4D lighting is based on a similar observation: since the 4D normal vector has only three independent degrees of freedom, choosing an *interior point* in a solid sphere determines the vector uniquely up to the sign of its component in the unseen fourth dimension (the "4D view-direction component").

Fig. 12 shows an example with a series of snapshots of this interactive interface at work. An additional information display shows the components of the 4D light vector at any particular moment.

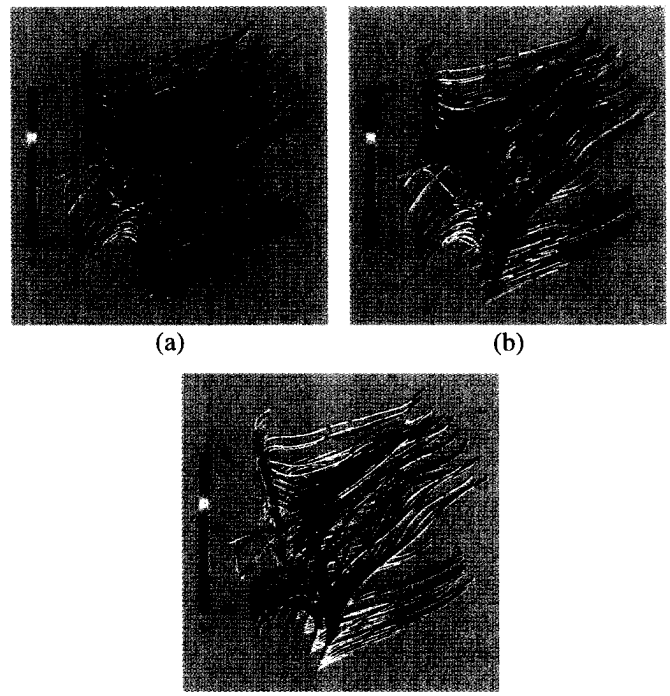


Fig. 12. Color coding a streamline data set using an interactively moving 4D "light" as a probe to isolate similar components of the quaternion fields associated to each point of each curve.

B. 4D Viewing and Three-Sphere Projection Control

Actually displaying quaternion field data mapped to the three-sphere requires us to choose a particular projection from 4D to 3D and a method for displaying the features of the streamlines. In order to expose all possible relevant structures, the user interface must allow the viewer to freely manipulate the 4D projection parameters. This control is easily and inexpensively provided using the 4D rolling ball interface [8], [13]. A special version of our "MeshView" 4D viewing utility [22] has been adapted to support real-time interaction with quaternion frame structures. Figs. 10 and 11 show snapshots from this interactive interface for 4D rotations using parallel and polar 4D projections, respectively.

The simplest viewing strategy plots wide lines that may be viewed in stereo or using motion parallax. A more expensive viewing strategy requires projecting a line or solid from the 4D quaternion space and reconstructing an ideal tube in real time for each projected streamline. The parallel transport techniques introduced in this paper are in fact extremely relevant to this task, and may be applied to the tubing problem as well (see, e.g., [5], [24], [18]).

C. 3D Rotations of Quaternion Displays

Using the 3D rolling ball interface, we can generate quaternion representations of 3D rotations of the form $q = (\cos\frac{\theta}{2}, \hat{\mathbf{n}}\sin\frac{\theta}{2})$, and transform the entire quaternion display by quaternion multiplication, i.e., by changing each point to $p' = q * p$. This effectively displaces the 3D identity frame in quaternion space from (1, 0, 0, 0) to q. This may be useful when trying to compare curves whose properties differ by a

rigid 3D rotation (a common occurrence in the parallel-transport frame due to the arbitrariness of the initial condition).

Other refinements might include selecting and rotating single streamlines in the quaternion field display to make interactive comparisons with other streamlines differing only by rigid rotations. One might also use automated tools to select rotationally similar structures based on minimizing the 4D scalar product between quaternion field points as a measure of similarity.

D. Exploiting or Ignoring Double Points

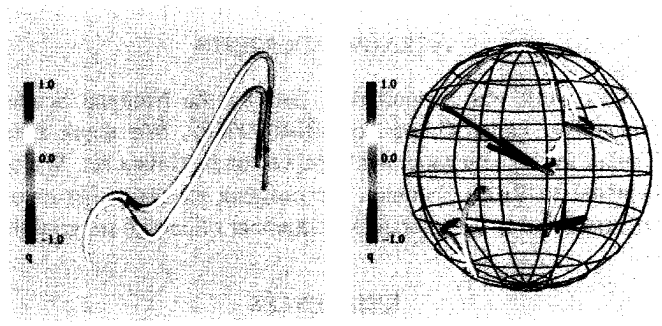
The unique feature of quaternion representations of orientation frames is that they are doubled. If we have a single curve, it does not matter which of the two points in S^3 is chosen as a starting point, since the others follow by continuously integrating small transformations. A collection of points with a uniform orientation as an initial condition similarly will evolve in tandem and normally requires only a single choice to see the pattern.

However, it is possible for a frame to rotate a full 2π radians back to its initial orientation, and be on the opposite side of S^3, or for a collection of streamlines to have a wide range of starting orientations that preclude a locally consistent method for choosing a particular quaternion q over its "neighbor" $-q$. We then have several alternatives:

- Include a reflected copy of every quaternion field in the display. This doubles the data density, but ensures that no two frame fields that are similar will appear diametrically opposite; the metric properties of similar curves will be easy to detect. In addition, 4D rotations will do no damage to the continuity of fields that are rotated to the outer surface and pass from the northern to the southern hyperhemisphere. If 4D depth is depicted by a color code, for example, a point that rotates up to the surface of the displayed solid ball will smoothly pass to the surface and then pass back towards the center while its color changes from positive to negative depth coding.
- Keep only one copy, effectively replacing q by $-q$ if it is not in the default viewing hyperhemisphere. This has the effect that each data point is unique, but that curve frames very near diametrically opposite points on the S^2 surface of the solid ball representing the north hyperhemisphere will be close in orientation but far away in the projection. In addition, when 4D rotations are applied, curves that reach the S^2 surface of the solid ball will jump to the diametrically opposite surface instead of passing smoothly "around" the edge to the southern hyperhemisphere.

E. Reciprocal Similarities and Differences

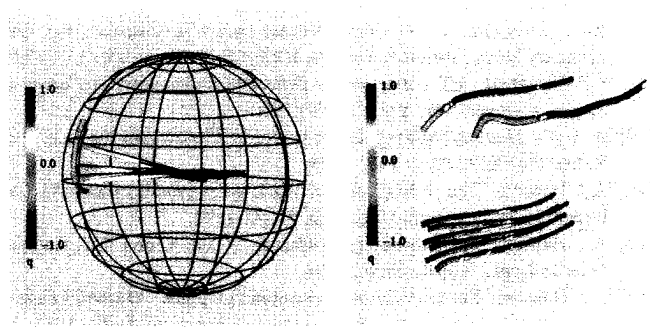
One of the most interesting properties of the quaternion frame method is the appearance of clusters of similar frame fields in the three-sphere display. Two reciprocal tools for exploring these properties immediately suggest themselves. In Fig. 13, we illustrate the effect of grabbing a cluster of streamlines that are spatially close in 3D space and then highlighting their counterparts in the 4D quaternion field space, thus allowing the separate study of their moving frame properties. This technique distinguishes curves that are similar in 3D space but have drastically different frame characteristics.



(a) (b)

Fig. 13. (a) Selecting stream fields that are close in the original 3D data display and (b) echoing them in the 4D quaternion Frenet frame display. The moving frames of these two curves are drastically different even though the curves appear superficially similar in 3D. The unseen component of 4D depth, with a range -1.0 to 1.0, is mapped to the color index.

Fig. 14, in contrast, shows the result of selecting a cluster of curves with similar frame-field properties and then highlighting the original streamlines back in the 3D space display. This method assists in the location of similar curves that could not be easily singled out in the original densely populated spatial display. We are examining a variety of alternative approaches to the design of such tools.



(a) (b)

Fig. 14. (a) Selecting stream fields that are close in the 4D quaternion Frenet frame display and (b) echoing them in the original 3D data display, thus showing the locations of similar curves that could not be easily singled out in the original 3D spatial display. The unseen component of 4D depth, with a range -1.0 to 1.0 is mapped to the color index.

VIII. Conclusion

In this paper, we have introduced a visualization method for distinguishing characteristic features of streamline-like volume data by assigning to each streamline a quaternion frame field derived from its moving Frenet or parallel-transport frame; curvature and torsion scalar fields may be incorporated as well. The quaternion frame is a four-vector field that is a piecewise smoothly varying map from each original space curve to a new curve in the three-sphere embedded in four-dimensional Euclidean space. This four-vector field can

be probed interactively using a variety of techniques, including 4D lighting, 4D view control, and interaction with selected portions of the data in tandem 3D streamline and 4D quaternion field displays.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under Grant No. IRI-91-06389. We thank Brian Kaplan for his assistance with the vector field data set. We are indebted to Bruce Solomon for bringing reference [4] to our attention, and to the referees for a number of helpful suggestions.

REFERENCES

[1] B. Alpern, L. Carter, M. Grayson, and C. Pelkie, "Orientation maps: Techniques for visualizing rotations (a consumer's guide)," *Proc. Visualization '93*, IEEE CS Press, pp. 183-188, 1993.

[2] S.L. Altmann, *Rotations, Quaternions, and Double Groups*. Oxford Univ. Press, 1986.

[3] D.C. Banks, "Illumination in diverse codimensions," *Computer Graphics, (Proc. SIGGRAPH '94)*, Ann. conf. series, pp. 327-334, 1994.

[4] R.L. Bishop, "There is more than one way to frame a curve," *Am. Math. Monthly*, vol. 82, no. 3, pp. 246-251, Mar. 1975.

[5] J. Bloomenthal, "Calculation of reference frames along a space curve," A. Glassner, ed., *Graphics Gems*, Cambridge, Mass.: Academic Press, pp. 567-571, 1990.

[6] M. Chen, S.J. Mountford, and A. Sellen, "A study in interactive 3d rotation using 2d control devices," *Computer Graphics, (Proc. SIGGRAPH '83)*, vol. 22, pp. 121-130, 1988.

[7] B. Cipra, "Mathematicians gather to play the numbers game," *Science 259*, pp. 894-895, 1993. Description of Alfred Gray's knots colored to represent variation in curvature and torsion.

[8] R.A. Cross and A.J. Hanson, "Virtual reality performance for virtual geometry" *Proc. Visualization '94*, IEEE CS Press, pp. 156-163, 1994.

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

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

[11] A.J. Hanson, "The rolling ball," D. Kirk, ed., *Graphics Gems III*, Cambridge, Mass.: Academic Press, pp. 51-60, 1992.

[12] A.J. Hanson, "Quaternion Frenet frames," Tech. Report 407, Computer Science Dept., Indiana Univ., 1994.

[13] A.J. Hanson, "Rotations for *n*-dimensional graphics," *Graphics Gems V*, A. Paeth, ed., Cambridge, Mass.: Academic Press, pp. 55-64, 1995.

[14] A.J. Hanson and R.A. Cross, "Interactive visualization methods for four dimensions," *Proc. Visualization '93*, IEEE CS Press, pp. 196-203, 1993

[15] A.J. Hanson and P.A. Heng, "Visualizing the fourth dimension using geometry and light," *Proc. Visualization '91*, IEEE CS Press, pp. 321-328, 1991.

[16] A.J. Hanson and P.A. Heng, "Illuminating the fourth dimension," *Computer Graphics and Applications*, vol. 12, no. 4, pp. 54-62, July 1992.

[17] A.J. Hanson and H. Ma, "Visualizing flow with quaternion frames," *Proc Visualization '94*, IEEE CS Press, pp. 108-115, 1994.

[18] A.J. Hanson and H. Ma, "Parallel transport approach to curve framing," Tech. Report 425, Computer Science Dept., Indiana Univ. 1995.

[19] J.C. Hart, G.K. Francis, and L.H. Kauffman, "Visualizing quaternion rotation," *ACM Trans. Graphics*, vol. 13, no. 3, pp. 256-276, 1994.

[20] J. Hultquist, "Constructing stream surfaces in steady 3d vector fields," *Proc. Visualization '92*, IEEE CS Press, pp. 171-178, 1992.

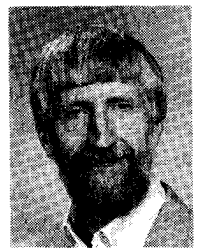
[21] J.T. Kajiya and T.L. Kay, "Rendering fur with three dimensional textures," J. Lane, ed., *Computer Graphics (Proc. SIGGRAPH '89)*, vol. 23, pp. 271-280, July 1989.

[22] H. Ma and A. Hanson, *Meshview*, A portable 4D geometry viewer written in OpenGL/Motif, available by anonymous ftp from geom.umn.edu, The Geometry Center, Minneapolis, Minn.

[23] J. Schlag, "Using geometric constructions to interpolate orientation with quaternions," J. Arvo, ed., *Graphics Gems II*, Academic Press, pp. 377-380, 1991.

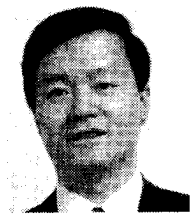
[24] U. Shani, and D.H. Ballard, "Splines as embeddings for generalized cylinders," *Computer Vision, Graphics and Image Processing*, vol. 27, pp. 129-156, 1984.

[25] K. Shoemake, "Animating rotations with quaternion curves," *Computer Graphics, (Proc. SIGGRAPH '85)*, vol. 19, pp. 245-254, 1985.



Andrew J. Hanson is a professor of computer science at Indiana University. Previously, he worked in theoretical physics and then with the perception research group at the SRI Artificial Intelligence Center. His research interests include scientific visualization with applications in mathematics and physics, machine vision, computer graphics, perception, and the design of interactive user interfaces for virtual reality and visualization applications.

Hanson was awarded his BA degree in chemistry and physics from Harvard College in 1966 and his PhD in theoretical physics from Massachusetts Institute of Technology in 1971. He is a member of AAAI, American Mathematical Society, American Physical Society, ACM Siggraph, IEEE Computer Society, and Sigma Xi.



Hui Ma was awarded the BS degree in applied mathematics from Zhejiang University, China in 1990. H. Ma is currently a doctoral candidate in computer science at Indiana University.

His research interests include computer graphics and visualization, user interface design, parallel and distributed systems.

Meshview: Visualizing the Fourth Dimension

Andrew J. Hanson Konstantine I. Ishkov* Jeff H. Ma†

Computer Science Department
Indiana University
Bloomington, IN 47405 USA

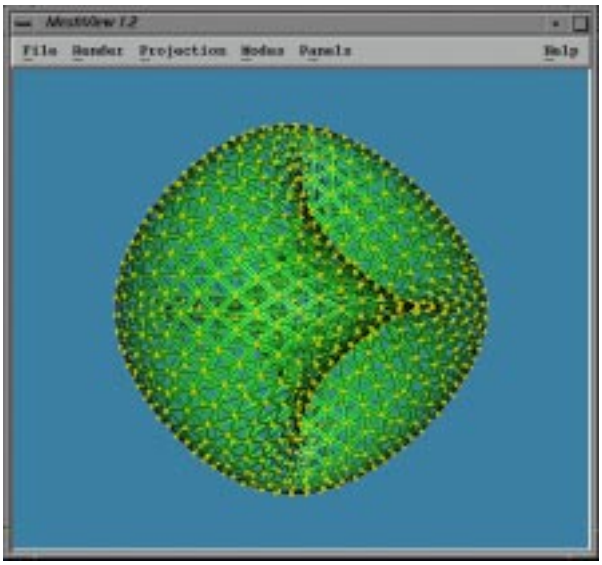


Figure 1: Meshview's interface window with a four-torus drawn using edges, vertices, and negative screen door transparency.

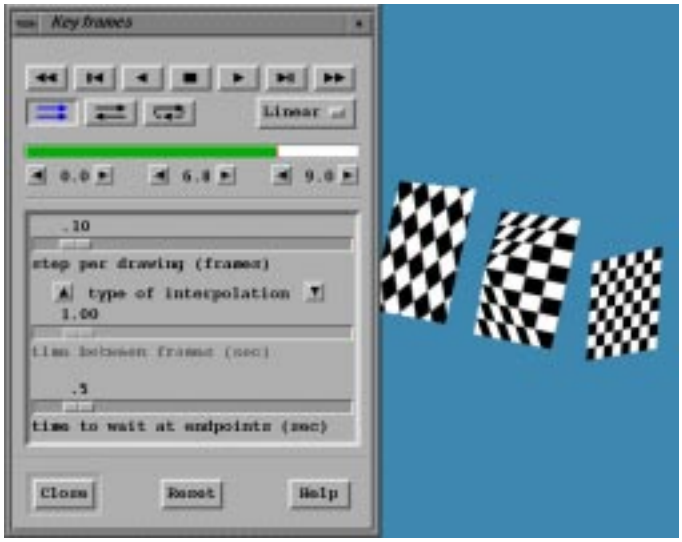


Figure 2: Meshview's key-frame animation interface controlling a set of animated polygons; the texture coordinates are also key-framed.

Abstract

Meshview is an interactive visualization system for viewing points, curves, and two-dimensional manifolds embedded in 3D or 4D, with the emphasis on handling 4D objects. All rigid motions in 3D and 4D can be performed under mouse (or 3D mouse) control, while key-frame animations support motions and deformations of such objects. Meshview is written in C, OpenGL, and X/Motif with the objective of being as compact, portable, and device-independent as possible within the given framework. The system has been used successfully to do research on a variety of problems such as 4D viewing interfaces, mathematical visualization of classical higher dimensional geometry, Riemann surfaces, functions of two complex variables, and 4D quaternion representations of 3D coordinate frames.

Keywords: four dimensions; curve and surface visualization

1 Introduction

Meshview is an interactive 4D viewing system that fluidly displays points, curves, and two-dimensional manifolds embedded in 3D or 4D, as well as key-frame animations representing motions and deformations of such objects. It is written in C, OpenGL, and X/Motif with the objective of being as compact, portable, and device-independent as possible within the given framework. It has been successfully compiled under Linux 2.2.2, SGI IRIX 5.2 to IRIX 6.5, and SUN SOLARIS 2.6. Meshview should in principle be portable to any workstation that supports OpenGL and X/Motif or appropriate simulators such as Mesa and LessTif.

Our basic motivations for developing yet another 4D viewer instead of using an existing system such as, e.g., Geomview [17], were twofold: (1) the available user interfaces, particularly for free-form rotational exploration, were poorly suited to both our performance needs and our preferences for interfaces that allow heads-up, context-free manipulation. We in fact conceived and implemented a particularly interesting context-free interface, the "rolling ball" in 4D (described below) that permits complete exploration of the 6 degree-of-freedom orientation space in 4D with only 3 controller parameters. (2) Our need for robust high-performance custom interfaces for perceptual psychology measurements and research into objects like quaternions and knotted spheres in 4D.

The basic design features of Meshview have evolved over a period of several years, beginning with the first release of version 1.0 in July of 1994, and continuing with a number of refinements, including screen-door transparency, animation, and texture, that were added during 1998–1999 to version 1.2. In the next sections, we outline the design philosophy and features of Meshview, point out its particular strong points, discuss the mathematical underpinnings of its unique interface features, and present a selection of applications, focusing in particular on the suitability of Meshview for building intuitions in classical mathematics and for the quaternion methods used routinely in computer graphics.

*Current address: Lucent Technologies, Holmdel, NJ
†Current address: Intel Corporation, Santa Clara, CA

2 Design Features of Meshview

The adjectives giving the overall goals of the design include:

- *Fast and general*. Use display lists in OpenGL. Enhance the Geomview/OOGL MESH and OFF file formats. Support color per object or color per vertex.
- *Small*. Keep as simple and independent as possible.
- *Portable*. Restrict to C, OpenGL, and X/Motif.
- *Freely distributable*. Non-proprietary.
- *Support document generation*. A straightforward image file generator is provided, and the state of any screen is easily saveable as a "setting" file for later restoration or refinement of a view.

The principal features of the design are:

- **Flexible file format.** Reads extensions of the Geomview/OOGL MESH, OFF and LIST file formats, plus its own enhancements FRAMES, DOT, and LINE.
- **Interactive examination support.** Rotates/translates/scales objects in 3D and 4D interactively under mouse control using the 3D and 4D rolling ball models for rotations.
- **Momentum.** The momentum option is available on all motions.
- **Drawing options.** Optionally draws faces, edges, vertices, normals, palette, unit sphere, the lighting vector, and a reference set of 4D axes.
- **Pseudocolor palettes.** A wide range of color palette options for 4D depth color coding a provided based on the NCSA palette library.
- **Geometry locator panel.** An interactive parametric space "picker" (or point locator) is supplied for any MESH file or list of MESH files. Any individual mesh in a set can be selected in turn.
- **Quaternion rotation panel.** Quaternion multiplication is isomorphic to multiplying a unit vector in the three-sphere S^3 by an orthogonal 4×4 matrix that can be derived directly from rotations acting on the 3D coordinate frame. This panel visualizes the change in orientation of the 3D frame, the unit quaternion to which it corresponds, and the action of the corresponding quaternion multiplication on the 4D object in the main window (which is not simply related to a 3D rotation).
- **Preservation of state.** System state is saved for later recovery, including current 3D and 4D viewing matrices, the camera setting, background color, light direction, and rendered `ppm` image of the current scene. This is useful for reconstructing the state of an illustration for a publication.
- **Restoration.** Loads palettes and saved system states.
- **Face shading options.** Surface facets can be flat shaded, smoothly interpolated, depicted with one color on both sides, depicted with two different colors for front and back surfaces, or textured. In addition, any arbitrary palette can be used to color code the 4D depth of each point in the current 3D projection: this is useful when rotating objects in 4D.
- **Projection options.** Both 3D and 4D permit perspective (polar projection) and orthogonal projection.
- **3D context can be disentangled.** Meshview supports a choice between applying the 4D rolling ball to the current screen coordinates of the object's 3D projection ("context-free," the default), or applying to the object's local 3D coordinate system context (using the "axes" display to help show the context). The latter is useful for looking at different sides of the object's 3D projection while performing a 4D rotation. This is especially useful for the 2D mouse interface.
- **Sample data.** The release includes a selection of example geometry files, including the 4D flat torus, Steiner surface (RP2 embedded in 4D), 4D Fermat surfaces and much more. (See the README file in the data directory for details, and see the color page of this article for examples.) A selection of short programs for generating such files is also available.
- **Help.** A simple online help file to remind the user of keyboard shortcuts and interface options is provided.

3 Fundamental Methods.

3D/4D Rolling Ball. The 4D rolling ball formula was derived in [4], and this is the method implemented in Meshview for both for the 2D mouse and the 3D mouse on the desktop. The remarkable property of this algorithm is that 4D orientation control requires exactly three control parameters, thus making it usable for a standard mouse with two buttons switching from (x, y)-plane control to (x, z)-plane control, and making it ideally suited to the "flying mouse" or CAVE "wand" 3-degree-of-freedom user interface devices. Let $\vec{X} = (X, Y, Z)$ be a displacement obtained from the 3-degree-of-freedom input device, and define $r^2 = X^2 + Y^2 + Z^2$. Take a constant R with units 10 or 20 times larger than the average value of r, compute $D^2 = R^2 + r^2$, compute the fundamental rotation coefficients $c = \cos\theta = R/D$, $s = \sin\theta = r/D$, and then take $x = X/r, y = Y/r, z = Z/r$, so $x^2 + y^2 + z^2 = 1$. Finally, rotate each 4-vector by the following matrix before reprojecting to the 3D volume image:

$$\begin{bmatrix} 1 - x^2(1-c) & -(1-c)xy & -(1-c)xz & sx \\ -(1-c)xy & 1 - y^2(1-c) & -(1-c)yz & sy \\ -(1-c)xz & -(1-c)yz & 1 - z^2(1-c) & sz \\ -sx & -sy & -sz & c \end{bmatrix}$$

The 3D rolling ball method is correspondingly used for 3D orientation control; it is basically the same formula except simplified by setting $z = 0$ and reducing the matrix to 3×3.

4 Geometry

Meshview data formats are strongly influenced by the OOGL (Object Oriented Graphics Language) file format used by Geomview [17], but circumstances and practical experience with complex geometries led us to deviate from strict adherence to the OOGL format.

Meshview 1.2 now supports MESH, OFF, LIST, FRAMES, DOT, and LINE formats. MESH, OFF and LIST are very similar to the OOGL file formats. The OOGL formats are not fully implemented (e.g., there is currently no support for files with more than 4 dimensions), but on the other hand several enhancements have been added. The FRAMES, DOT and LINE formats are specific to Meshview, where the FRAMES format is used for key frame animation, and DOT and LINE are used to display dots and lines respectively.

The overall syntax is quite straightforward, and is documented in an accompanying README file that we cannot describe in detail here due to space limitations. The data files are composed of

lists of points in 3D or 4D, various color attachments, and texture coordinates, all of which get translated in the implementation into the obvious OpenGL representations.

5 Selected Controls

Below we present a selection of the possible controls in Meshview; (u, v) denotes the incremental mouse coordinates.

3D viewing:	
leftbutton	3D rotation (3D rolling ball) R3(u,v)
middlebutton	3D translation in x-y plane T3(u,v,0)
rightbutton	3D translation along z axis T3(0,0,-v)
Shift+right	3D rotation around z axis R2(u)
3D lighting:	
Ctrl+left	3D rotation (3D rolling ball) R3(u,v)
Ctrl+middle	3D rotation around z axis R2(u)
4D viewing:	
Shift+left	xyw rotation (4D rolling ball) R4(u,v,0)
Shift+middle	xzw rotation (4D rolling ball) R4(u,0,-v)
`<Key>r`	Reset the 4D and 3D matrices and 3D light direction, stop momentum.
`<Key>F3`	(3D mouse) Toggles 3D mouse. * Left 3D mouse: 4D rolling ball * Middle 3D mouse: 3D orientation and position * Right 3D mouse: reset
Appearances:	
`<Key>1`	both sides of face use same color
`<Key>2`	two sides of face use different colors
`<Key>3`	4D depth color coding
`<Key>4`	texture coding
`<Key>5,6,7`	screen-door off, positive, negative
Utilities:	
`<Key>f`	toggle face drawing (default on)
`<Key>e`	toggle edges
`<Key>v`	toggle vertices
`<Key>n`	toggle normals
`<Key>u`	toggle unit quaternion sphere
`<Key>p`	toggle palette
`<Key>l`	toggle light ray
`<Key>a`	toggle 4D orientation axes
Viewing:	
Ctrl+p	3D perspective projection (default)
Ctrl+o	3D orthogonal projection
`<Key>w,x,y,z`	4D projection along w(default), x, y, or z-axis
Shift+o	4D orthogonal projection (default)
Shift+p	4D polar projection

6 Applications

Meshview has been used in our laboratory to examine objects and create imagery for journal articles since its conception in 1994. With the addition of further features such as stereo, key-frame animation, texture, and screen door transparency during the last year, many additional applications are possible. Among the specific applications for which Meshview has been employed in its bare or task-enhanced forms, we note the following:

6.1 4D Mathematical Visualization

The production of the video animation "knot^4" [15], which concerned the visualization of knotted spheres embedded in four Euclidean dimensions, generated a family of very interesting objects that begged to be explored interactively. Meshview in some sense was originally motivated by our need for our own customizable system for this purpose. As a result, some of the earliest models created for Meshview came from the film; a typical 4D "spun knot," that is not even knotted is shown in Figure 6. Many other "classic" 4D mathematical figures are in the Meshview geometry library, including the 4-torus (just the product of two circles) in Figures (1,3), and the crosscap/Steiner Roman Surface in Figures (4,5), which can in fact be rotated into one another in Meshview (a fact uncovered during the early work by Banchoff on 4D visualization — see [1]); the equations for both of these figures can be found in the classic book *Geometry and the Imagination* [16].

6.2 Complex Functions

Some of the first author's earliest work on mathematical visualization started with attempts to visualize the Fermat surfaces [9, 10, 14], which are extensions of the Fermat-theorem equations to two complex variables of the form

$$(z_1)^n + (z_2)^n = 1$$

which are in effect the n-fold Riemann surfaces of the complex equation

$$f(z) = (1 - z^n)^{(1/n)} .$$

Meshview provides any number of ways of exploring the 2D manifolds that result from solving these two real equations in four real variables and looking at projections of the natural embedding in four real dimensions. In Figure 9, we show such a surface color-coded by the phase transformations from the fundamental domain; Figure 10 shows the same object with pseudocolor coded 4D depth. Figure 11 shows the two complex planes $z_1 = 0$ and $z_2 = 0$ superimposed on the $n = 3$ Fermat surface. Solutions of the closely related equations $(z_1)^m (z_2)^n = 1$ are shown in Figures 7 and 8.

6.3 Quaternion Visualization

The relation of 4D unit quaternions to rotations, orientations, and camera frame interpolation has been familiar to computer graphicists since their relevance was pointed out by Shoemake in 1985 [18]. Meshview was used extensively to create the figures and animations accompanying our research on mapping streamline orientation frames to quaternion spaces [11]. Subsequent research [5] employed Meshview as well to visualize the nature of optimal quaternion curves and surfaces corresponding to frame assignments for 3D curves and surfaces. Figures 12 and 13 show the quaternion form of several possible tangent frame assignments for a (2,3) torus knot; Figure 14 adds an actual quaternion surface representing the space of all possible such frames.

6.4 Context for Perceptual Experiments

Meshview's basic facilities have been adapted to a series of experiments currently underway in the Perception/Action laboratory at the Indiana University Department of Psychology. The recently added key-frame animation and deformation capabilities are essential here; future work on the perceptual nature of 3D and 4D rigid versus elastic motion is planned in this context.

6.5 Virtual Reality Features

Several features of the current Meshview support desktop virtual reality functionality. On a stereo-equipped SGI, the system will bring up a stereo screen that may be viewed with Stereographics CrystalEyes equipment. Various parameters can be adjusted to the

user's taste. We generally assume a non-moving user so that head-tracking, while feasible, is not a high priority in the unenhanced desktop system.

Perhaps of more interest is the support (implemented under IRIX 6.x, but not difficult in general) for the Logitech 3D mouse, which is a full six-degree-of-freedom device with four buttons. By employing the 4D rolling ball algorithm [4] in its purest form, the 3D position alone of the 3D mouse can naturally control all six rotation planes of a 4D mathematical object. This is accomplished by having (x, y, z)-motions rotate in the (x, w),(y, w), and (z, w) planes, respectively, while "rowing" circular motions of the mouse position in the (y, z),(z, x), and (x, y) planes, respectively, produce 4D rotations in the (y, z),(z, x), and (x, y) planes themselves, exhausting the entire 4D orientation space. The motion of a single 3D point at the tip of the 3D mouse can thus be used to seek out any possible projection from 4D into 3D.

7 Conclusion and Future Work

The Meshview system is a minimalist approach to a very flexible and full-featured utility for examining and building intuition about 4D structures, e.g., 4D geometry and topology, two complex variables, and quaternions. Its implementation strategy is to use only C, Motif, and OpenGL, thereby facilitating portability, maintainability, extensibility, and compactness of design. The supported data formats conform very closely to the OOGL formats implemented by Geomview [17], with a handful of extensions. A variety of desktop virtual reality techniques are incorporated, including the 4D rolling ball method for manipulating 4D displays, switched-field Stereographics stereography, and the Logitech flying mouse. We anticipate a limited CAVE^{TM} [3] implementation in the near future.

Future plans include a number of ambitious extensions to create robust and publicly available implementations of related high-dimensional visualization techniques, including interactive 4D rotation and and volume rendering of 3-manifolds embedded in 4D [8], 4D renderings of 3D scalar fields [7], and automatic generation and fast interactive rendering of "thickened" 2-manifolds in 4D [6, 2]. Specific extensions involving quaternion visualization and quaternion frame optimization are also envisioned, including quaternion maps of streamlines and stream surfaces in the manner of [11], and the automatic generation of optimal tubings of curves and framings of surfaces as described in [5]. The general approaches to more sophisticated data navigation strategies such as those proposed in [12, 13] would also be appropriate extensions to the Meshview family of interaction modes.

The URL for Meshview is `ftp://ftp.cs.indiana.edu/` and the file is `pub/hanson/Meshview.1.2.tar.gz`.

Acknowledgments

This research was made possible in part by NSF infrastructure grant CDA 93-03189. Thanks are due to John N. Huffman for his assistance with the Linux port.

References

[1] T. F. Banchoff. Beyond the third dimension: Geometry, computer graphics, and higher dimensions. *Scientific American Library*, 1990.

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

[3] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 135–142, August 1993.

[4] A. J. Hanson. Rotations for n-dimensional graphics. In Alan Paeth, editor, *Graphics Gems V*, pages 55–64. Academic Press, Cambridge, MA, 1995.

[5] A. J. Hanson. Constrained optimal framings of curves and surfaces using quaternion gauss maps. In *Proceedings of Visualization '98*, pages 375–382. IEEE Computer Society Press, 1998.

[6] A. J. Hanson and R. A. Cross. Interactive visualization methods for four dimensions. In *Proceedings of Visualization '93*, pages 196–203. IEEE Computer Society Press, 1993.

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

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

[9] A. J. Hanson, P. A. Heng, and B. C. Kaplan. Techniques for visualizing Fermat's last theorem: A case study. In *Proceedings of Visualization 90*, pages 97–106, San Francisco, October 1990. IEEE Computer Society Press.

[10] A. J. Hanson, P. A. Heng, and B. C. Kaplan. Visualizing Fermat's last theorem. *SIGGRAPH Video Review*, 61(4), 1990. 3:37 minute video animation.

[11] A. J. Hanson and H. Ma. Quaternion frame approach to streamline visualization. *IEEE Trans. on Visualiz. and Comp. Graphics*, 1(2):164–174, June 1995.

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

[13] A. J. Hanson and E. Wernert. Constrained 3D navigation with 2D controllers. In *Proceedings of Visualization '97*, pages 175–182. IEEE Computer Society Press, 1997.

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

[15] Andrew J. Hanson. knot^4. Video animation of knotted spheres in four dimensions. Published in Siggraph Video Review **93**, Scene 1 (1993).

[16] D. Hilbert and S. Cohn-Vossen. *Geometry and the Imagination.* Chelsea, New York, 1952.

[17] Mark Phillips, Silvio Levy, and Tamara Munzner. Geomview: An interactive geometry viewer. *Notices of the Amer. Math. Society*, 40(8):985–988, October 1993. Available by anonymous ftp from geom.umn.edu, The Geometry Center, Minneapolis MN.

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

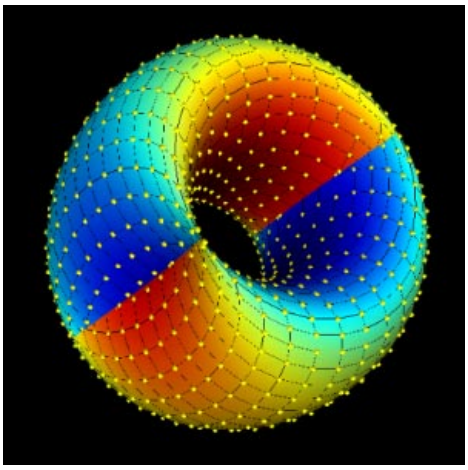


Figure 3: 4D depth colored 4-torus

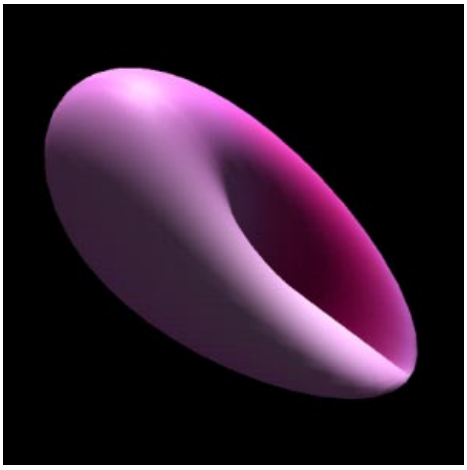


Figure 4: Crosscap = 4D rotated Roman surface

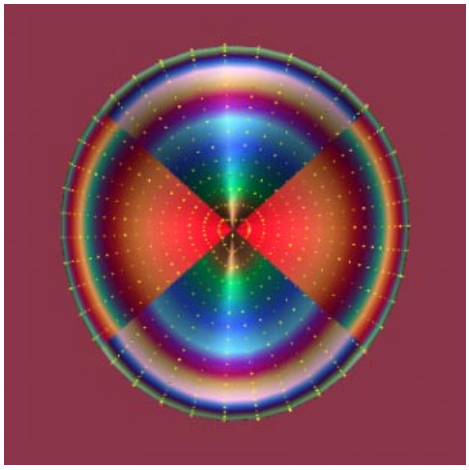


Figure 5: Steiner Roman surface

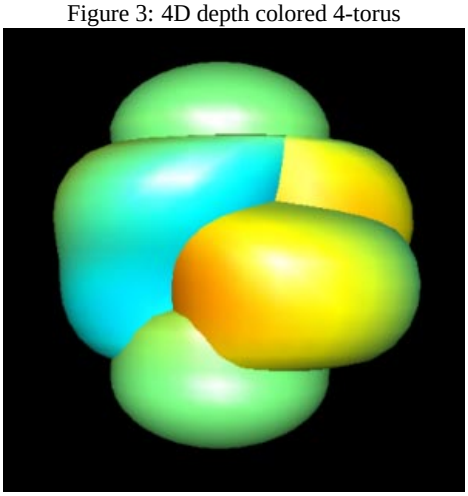


Figure 6: Twist-spun trefoil knot

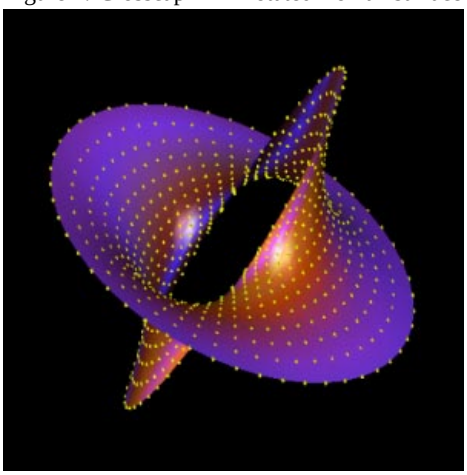


Figure 7: $z_1 z_2 = 1$

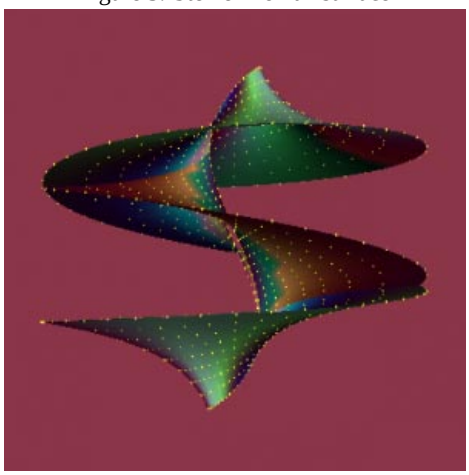


Figure 8: $z_1 (z_2)^2 = 1$

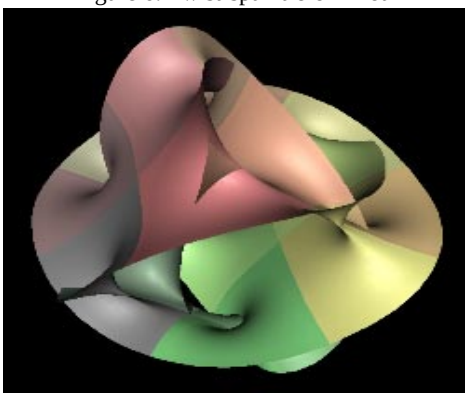


Figure 9: $N = 4$ Fermat surface coded by 2D complex phase transform

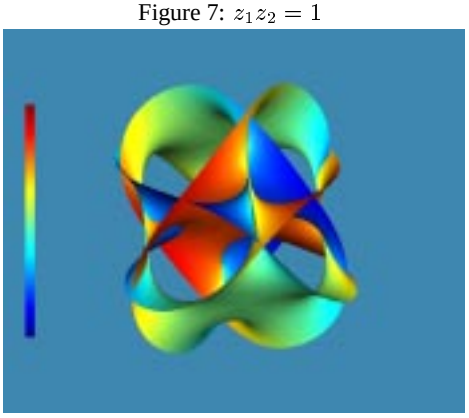


Figure 10: $N = 4$ Fermat surface with color coded 4D depth

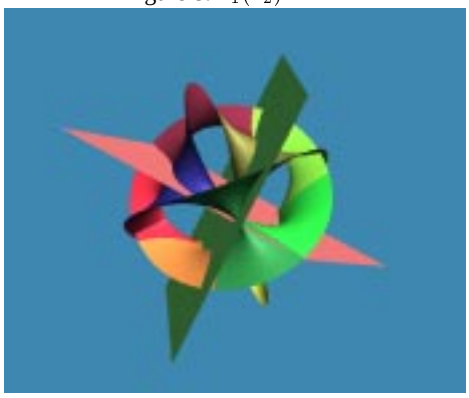


Figure 11: $N = 3$ Fermat surface with $z_1 = 0$ and $z_2 = 0$ complex planes

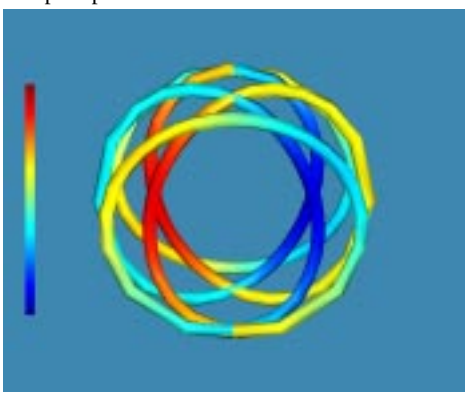


Figure 12: Quaternion Frenet frame of (2,3) torus knot with color coded 4D depth

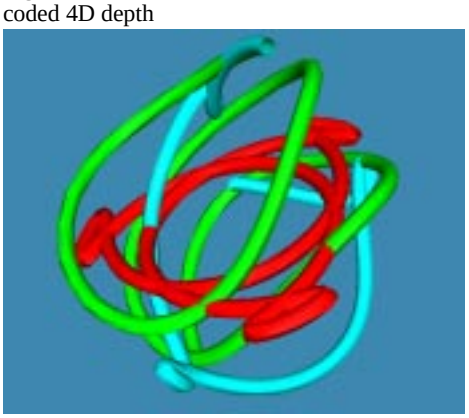


Figure 13: Selection of alternate quaternion tangent frames for (2,3) torus knot

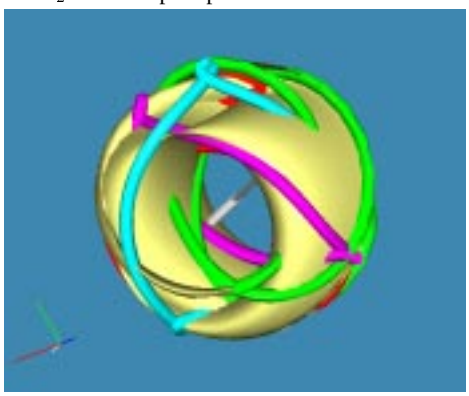


Figure 14: Quaternion manifold of allowed (2,3) torus knot tangent frames