Meshview: Visualizing the Fourth Dimension

Andrew J. Hanson Konstantine I. Ishkov* Jeff H. Ma†

Computer Science Department
Indiana University
Bloomington, IN 47405 USA

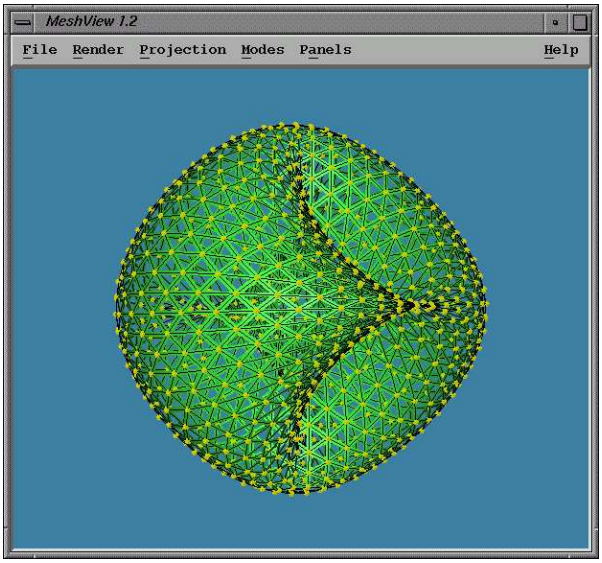

Figure 1: Meshview's interface window with a four-torus drawn using edges, vertices, and negative screen door transparency.

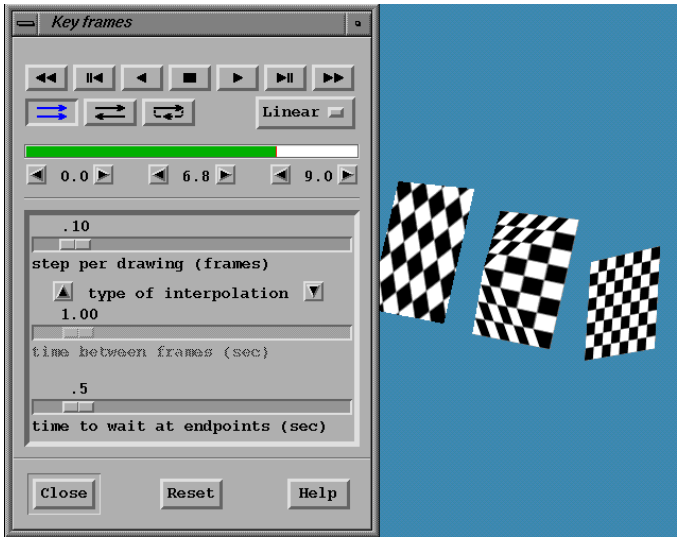

Figure 2: Meshview's key-frame animation interface controlling a set of animated polygons; the texture coordinates are also key-framed.

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$^{\text{TM}}$ [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.

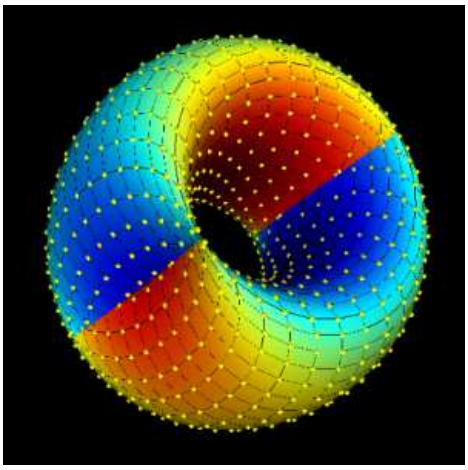

Figure 3: 4D depth colored 4-torus

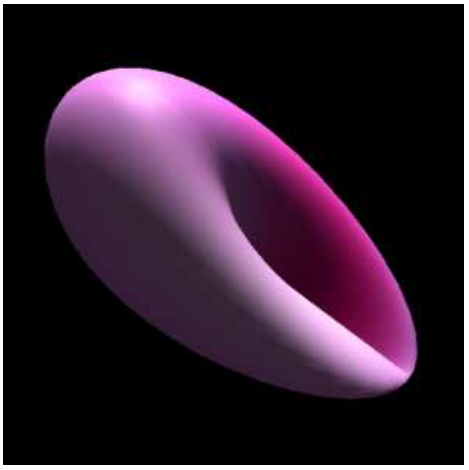

Figure 4: Crosscap = 4D rotated Roman surface

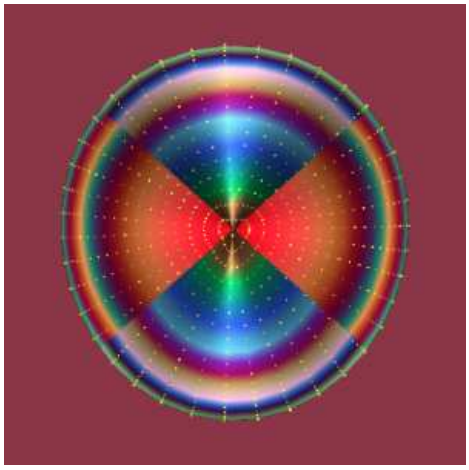

Figure 5: Steiner Roman surface

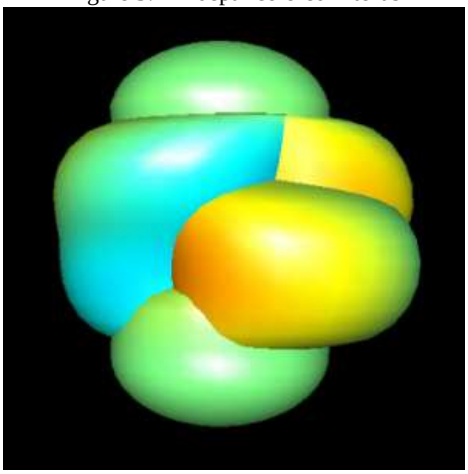

Figure 6: Twist-spun trefoil knot

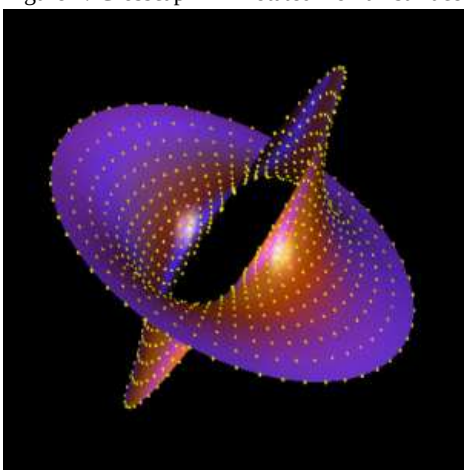

Figure 7: $z_1 z_2 = 1$

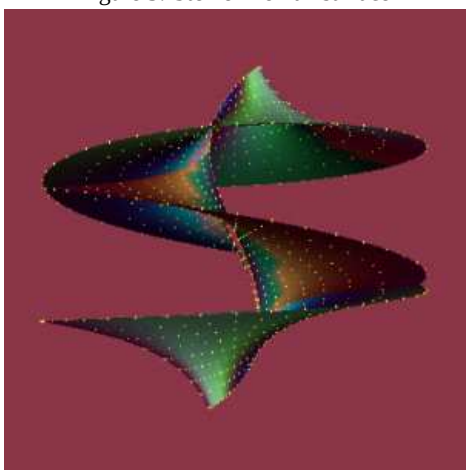

Figure 8: $z_1 (z_2)^2 = 1$

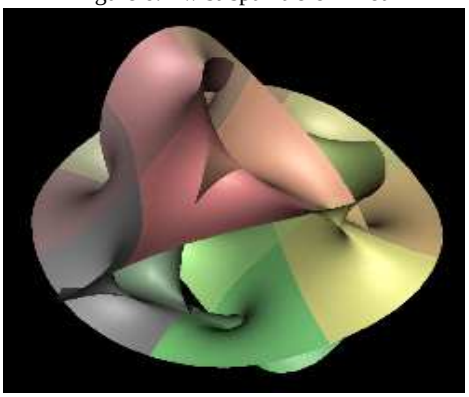

Figure 9: $N = 4$ Fermat surface coded by 2D complex phase transform

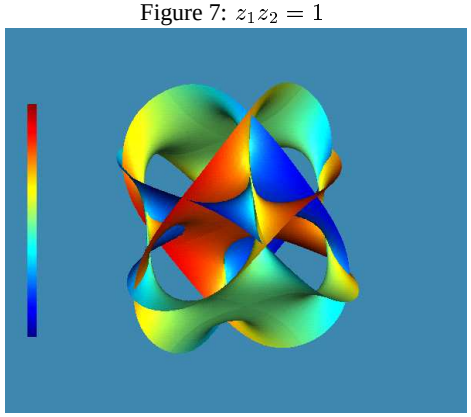

Figure 10: $N = 4$ Fermat surface with color coded 4D depth

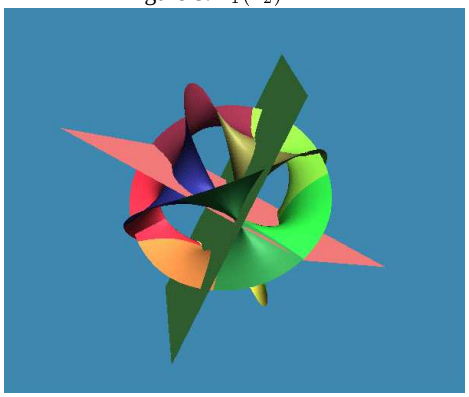

Figure 11: $N = 3$ Fermat surface with $z_1 = 0$ and $z_2 = 0$ complex planes

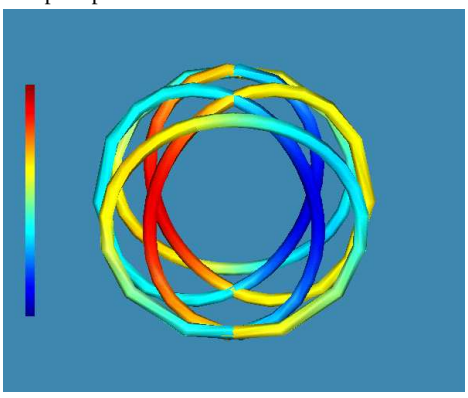

Figure 12: Quaternion Frenet frame of (2,3) torus knot with color coded 4D depth

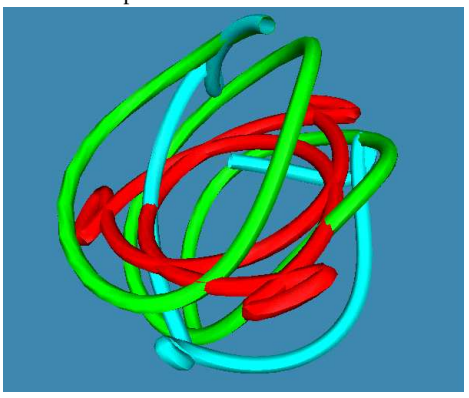

Figure 13: Selection of alternate quaternion tangent frames for (2,3) torus knot

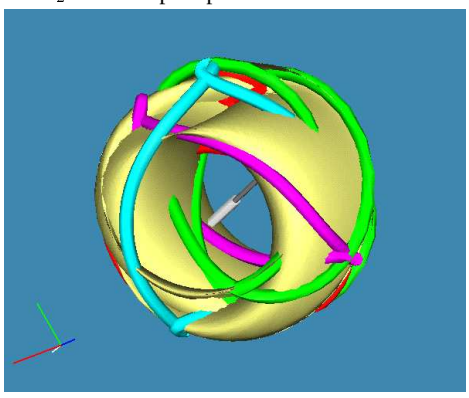

Figure 14: Quaternion manifold of allowed (2,3) torus knot tangent frames