Visualizing 2-dimensional Manifolds with Curve Handles in 4D

Hui Zhang, Jianguang Weng, and Guangchen Ruan

Abstract— In this paper, we present a mathematical visualization paradigm for exploring curves embedded in 3D and surfaces in 4D mathematical world. The basic problem is that, 3D figures of 4D mathematical entities often twist, turn, and fold back on themselves, leaving important properties behind the surface sheets. We propose an interactive system to visualize the topological features of the original 4D surface by slicing its 3D figure into a series of feature diagram. A novel 4D visualization interface is designed to allow users to control 4D topological shapes via the collection of diagram handles using the established curve manipulation mechanism. Our system can support rich mathematical interaction of 4D mathematical objects which is very difficult with any existing approach. We further demonstrate the effectiveness of the proposed visualization tool using various experimental results and cases studies.

Index Terms—math visualization, 4D, deformation, Reidemeister theorem

1 Introduction

People generally learn better in a situation that resembles a past experience, and knowledge of shape comes from a combination of sight, touch, and exploration [1]. Mathematical experience is not an exception. Mathematicians popularize unfamiliar concepts starting with drawing familiar analogous diagrams.

Our task in this paper is to show how one can fully understand, propose, and trace the evolution of unfamiliar 4D surfaces by decomposing the task into familiar 2-dimensional steps. We typically would like to investigate 4D surfaces, which are an important class in descriptive topology [18]. Surfaces in 4D play many roles analogues to those of familiar curves in 3D; in particular, spheres are the analogs of closed curves and knots can be generalized to "knotted surfaces" (closed 2D surfaces embedded in 4D). We start from the implementation of a multi-cursor-enabled interface that allows us to take control of one-dimensional curves embedded in 3D space with a set of elementary string interactions. Having established the mechanisms and intuition of this artifice, we proceed to a novel mathematical visualization interface capable of moving two-dimensional surfaces around in 4D space by decomposing the unfamiliar 4D tasks into a sequence of familiar string interactions. In this way, we can proceed to attack families of significant challenges in 4D intuitive visualization such as twisting and turning in four dimensions with feature curve handles, canceling/adding pinch-points in topological constructions, modeling 4D "chain" structures consisting of linked spheres and ribbons, and visualizing the refinement of mathematical curves and surfaces by generating feature-aware snapshots.

2 Motivation

The idea of cross-dimensional understanding has long been a subject of fascination. Carter's *How Surfaces Intersect in Space* [8] contains numerous hand-drawn diagrams to unfold the visual secrets of the evolution of 3D and 4D geometric topology with a sequence of feature diagrams in a *time-lapse* form. Figure 1 illustrates one such picture story that starts with a set of familiar string interactions: curves in 3D can be transformed into one another by means of such "flat" string interactions applied to their 2-dimensional "knot-crossing" diagrams, mathematically named "Reidemeister moves" (see Figure 1① and [22]). The interesting part of the story is about the close relation between the evolution of curves and that of surfaces in space. In Figure 1② a set of hand-designed diagrams are created by cutting a curved surfaces with parallel cutting planes. The diagrams, called a *movie* of the surface, exposes cross-sections of the surface and reveals its internal features and structure. Figure 1③-④ disclose the mathematical meaning of these curved surfaces with self-intersections: the 3D figures are used to depict smooth topological disks in 4-dimensional space by giving "broken" surface diagrams, and they are bounded by curves whose 2-dimensional "knot-crossing" diagrams indicate which sheet is *above* another from the projection point into 3-space. (Carter's book thinks of 4-dimensional space as a collection of curves in 3-dimensional space, in a time-lapse form.) With these established mechanisms, Carter's picture story proceeds to indicate how surfaces in 4-space can be evolved into one another: the unfamiliar evolution of a surface in 4D can be proposed and traced by interpolating the successive stages of "moves-to-movies", i.e., the more picture-friendly evolution of a collection of "knot-crossing" diagrams. The interpolations applied to 3D figures indeed depict how surfaces interact with one another in 4-dimensions (see e.g., Figure 1⑤↦⑥, the 4D evolution is one of the Roseman 4D moves, which illustrates the canceling or adding branch points through a saddle accomplished by a fairly familiar Reidemeister move on its cross-section.)

In our work, we develop a computer realization of drawings from Carter's book that enhances the user's understanding of topological spaces, in particular, curves embedded in 3D and surfaces embedded in 4D. The key contributions of our work are summarized below:

- This paper reports the first interactive visualization system that naturally couples the visual metaphors of mathematical entities and intuition-building interactions based on their geometric and topological features. Our interface allows one to propose the evolution of unfamiliar 4D surfaces by decomposing the task into *understandable* 2-dimensional steps. Each step can be understood and built from familiar atomic pieces. Furthermore, as the mathematical entity changes, its topological change is traced in a *time-elapse* form.
- We introduce an innovative "flat" style represenation to describe 3D-embedded curves in 2D (Section 3), and 4D-embedded surfaces in 3D (Section 4). Our representation is simple enough to transform mathematical entities into understandable lower dimensional analogues, and still powerful enough to depict the evolution of curves and surfaces with topology preserving deformations in the full dimensional space.
- Last, we investigate and implement a family of novel interaction and exploration methods for building mathematical visualization applications involving both static structures, such as 4D "chain" structure, and changing structures requiring interactions, such as reidemeister-type moves for surfaces in four-dimensional, or the

- *Hui Zhang is with Pervasive Technology Institute at Indiana University. E-mail: huizhang@iu.edu.*
- *Jianguang Weng is with Zhejiang University of Media and Communication. E-mail: wengjg@zju.edu.cn.*
- *Guangchen Ruan is with School of Informatics and Computing at Indiana University. E-mail: gruan@umail.iu.edu.*

Manuscript received 31 Mar. 2014; accepted 1 Aug. 2014. Date of publication 11 Aug. 2014; date of current version 9 Nov. 2014.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.
Digital Object Identifier 10.1109/TVCG.2014.2346425

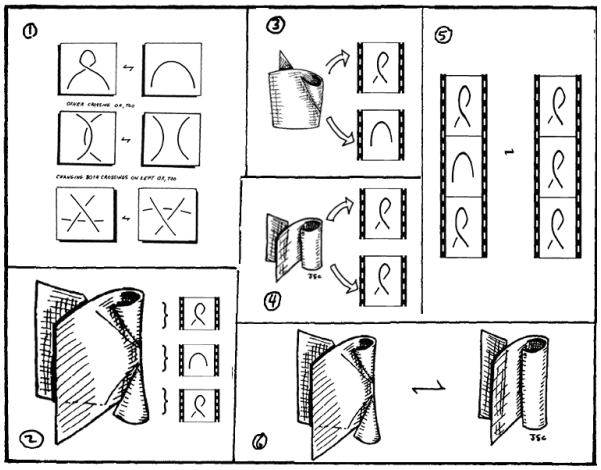


Fig. 1. Evolving 4D surfaces illustrated by sequences of hand-drawn diagrams (original drawing by Scott Carter, modified and re-arranged by the authors for pedagogical purposes). ①: the three types of Reidemeister moves. ②: a sequence of diagrams are obtained by cutting the curved surface sheet with hyperplanes parallel to the horizon; the curved surface is indeed the 3D representation of a topological disk smoothly embedded in 4-space. ③④: the underlying structure of the topological disks is revealed by the "crossing-diagrams" of their boundary curves. ⑤↦⑥: depicting and deforming the 4D surface by generating and manipulating the diagram collection obtained along the surface's longitudinal axis (the illustrated evolution of the surface in this specific example causes a hyperbolic confluence of branch points [10].)

Roseman 4D moves (Section 5). Users can propose mathematical moves and understand the evolution of the underlying mathematical structures with various redundant visual cues in our interface.

3 Understandable 3D Example: Touchable "Flat" Strings

In this section, we focus on an understandable and familiar 3D visualization task. The aim here is to model and manipulate 3-dimensional curves with touchable strings in 2-dimensional interaction space. We describe the families of methods used to implement the elementary string interaction procedures and user interface elements.

3.1 Interaction Rules

For our principal test case of curves positioned in $\mathbb{R}^3$, we have considered a set of interaction rules that are inspired by hand-designed topological illustrations. For example, our approach models the curves to be infinitely malleable: once shaped they retain the shape. The length is of no concern: sometimes in meters, and sometimes in inches. The evolution in space respects topological constrains: it does not involve cutting the string or passing the string through itself. The interaction is simple and still powerful: each evolution in 3D is proposed by an understandable 2D action. Our fundamental techniques are based on a wide variety of prior art, including 2D Reidemeister move interface for curve manipulation [37], sketching based interfaces focusing on mathematical knot construction and manipulation (see, e.g., Cohen [12], Scharein [27], and Zhang [36]), and multi-touch variants on graph and curve exploration (see, e.g., Schmidt [28]).

3.2 Implementation Details

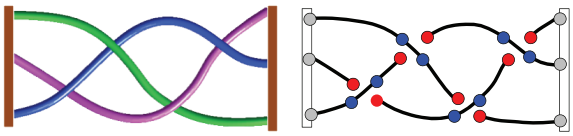
Our first task is to transform a 3D curve into a hybrid structure that can be simple enough to be represented as a 2D diagram yet powerful enough to extract the curve's 3D topology information. Unlike all previous efforts (see e.g., Scharein's 3D dynamical model [27] and Zhang's 2D knot representation [37]), our representation uses a "flat" style topological structure incorporating *balanced ternary* depth values [5].

3.2.1 Generating the Diagrams

An initial diagram of a 3D cruve can be obtained by projecting each vertex from $\mathbb{R}^3$ ($xyz-$space) to $\mathbb{R}^2$ ($xy-$plane), but various parts of the diagram appear to touch each other due to the projective collapse of the z dimension. Let $\mathbf{K} = (\mathbf{V}, \mathbf{E})$ represent this initial diagram of a given smooth curve in $\mathbb{R}^3$, where $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2 ..., \mathbf{v}_n\}$ is the finite set of *vertices* of the polygon and $\mathbf{E}$ is the set of *edges* $\{\mathbf{e}_1, \mathbf{e}_2 ..., \mathbf{e}_n\}$. Our method walks along the planar diagram, and restores the 3D curve's topology by assigning each vertex a ternary "eye-coordinate" or, depth z: $z = 1$ for vertices on strands crossing *over* another section, $z = -1$ for vertices on strands crossing *under* another, and $z = 0$ for vertices on un-interrupted strands. Consider Figure 2 for an illustration of our algorithm. Figure 2(a) gives three pieces of smoothly embedded curves in $\mathbb{R}^3$. Figure 2(b) shows the generated flat representation where these curves' salient global structures can be identified and the original topology information is extracted. Let $C = \{1, 2, ..., n\}$ be the set of indices of all vertices in $\mathbf{V}$. Our ternary "eye-coordinate" representation splits C into three representative subsets:

- $C_\phi^+ = \{\phi_1^+, \phi_2^+, ..., \phi_i^+\}$, the set of indices of the vertices with positive 3D-depth;
- $C_\phi^- = \{\phi_1^-, \phi_2^-, ..., \phi_i^-\}$, the set of indices of the vertices with negative 3D-depth;
- $C_\theta = C - C_\phi^+ - C_\phi^-$, the set of indices of the vertices with zero 3D-depth.

In Figure 2(b) we draw those vertices in C_ϕ^+ in blue, C_ϕ^- in red, and C_θ in grey (selectively). Meanwhile, the diagram is rendered in a pen-and-ink style to create the classical "knot-crossing diagrams" by attaching a thickened curve segment in background color behind each of the curve segments in foreground color [35], or, with more difficulty, using a two-pass gap rendering approach [21]. In this way, we render a two-dimensional projected curve that is broken each time it is occluded by a piece of the whole curve that is nearer to the 3D projection point (e.g., the pieces between vertices in red are occluded by those in blue.)



(a) (b)

Fig. 2. Turning smoothly embedded curves into flat style diagrams with ternary depth values. (a) Examples of mathematical braids embedded in 3D, though there are obviously many other shapes that could be used. (b) Diagrams in 2-dimensional space, where each vertex's $z-$coordinates is mapped to a color-coded balanced ternary value (*blue* is over, *red* is under, and *grey* is flat).

3.2.2 Topology-aware Diagram Evolution

In this section we implement the types of elementary string interactions on the generated diagrams to depict 3D curves' evolution. The fundamental idea of the proposed paradigm is based on a clever but simple geometric construction: the three Reidemeister moves (see e.g., Figure 1①). In our implementation we propose an algorithm which consists of three main steps: the determination of the evolved string segment (region of interest, or ROI), string deformation in the least square sense, and topology reconstruction.

Region of Interest at "Fingertips". We consider topological manipulation of 2D diagrams with multiple virtual mouse cursors, and assume each contact point as a "cursor". One value of doing this is to interface our algorithms and implementations with an abstract input device layer, and can later support both mouse-keyboard and multi-touch settings. The identification of ROI starts with the initialization

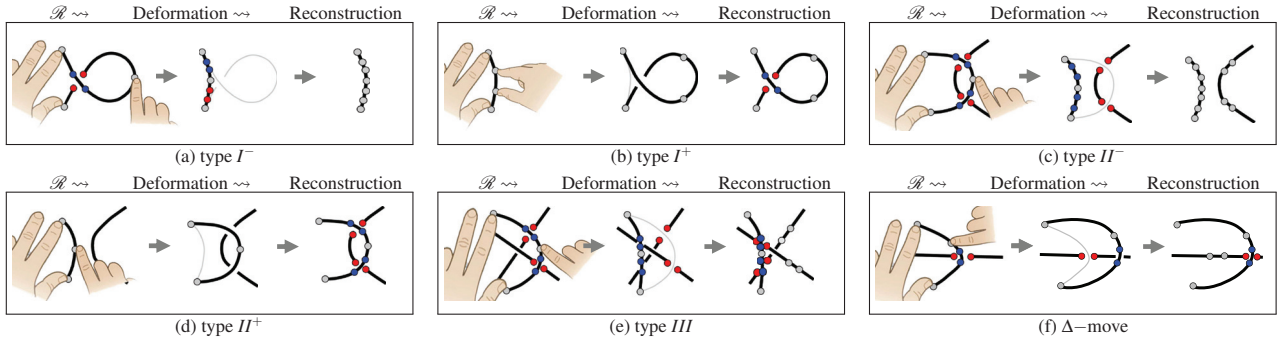


Fig. 3. The six types of elementary string interactions in a multi-cursor interface. Each string interaction is proposed by three steps: *(1) ROI identification, (2) deformation with the shape handle vertice and depth handle vertice, and* (3) topology reconstruction post-deformation.

of a set of handle vertices in C_θ (see those vertices colored in grey[1] in Figure 3) by matching those vertices in their geometric space to the contact points' pixels in the input device's screen space. In practice, our system recognizes 3-point and 4-point multi-touch gestures by initializing the set of indices of the handle vertices:

$$C_s = \{s_1, s_2, ..., s_m\},$$

where $m = 3 \mid 4$ and $C_s \in C_\theta$. When the user attempts to manipulate the diagram with moves (by touch gesture or, mouse pointer), C_s is re-ordered in place such that s_2 (or $s_2 s_3$, in a 4-point manipulation) denotes the index of the moving handle vertex and $s_1 s_3$ (or $s_1 s_4$, in a 4-point manipulation) denotes the indices of the boundary (i.e., fixed) vertices. We can thus identify ROI as a polygonal chain $\mathscr{R}$, specified by a sequence of vertices $(\mathbf{v}_{s_1}, ..., \mathbf{v}_{s_2}, ..., \mathbf{v}_{s_3})$ or, $(\mathbf{v}_{s_1}, ..., \mathbf{v}_{s_2}, ..., \mathbf{v}_{s_3}, ..., \mathbf{v}_{s_4})$ in a 4-point manipulation.

With the identified ROI, we next remember within $\mathscr{R}$ another set of indices of vertices whose depth values are *not* zero (see e.g., Figure 3, vertices colored in blue or red):

$$C_d = \{d_1, d_2, ..., d_n\},$$

where $C_d = (C_\phi^+ \cup C_\phi^-) \cap \{s_1, s_1 + 1, ..., s_m\}$.

String Deformation. We have implemented an algorithm to deform strings using Laplacian. The string representation can be exploited to allow manipulating the diagram while having the underlying spatial embedding adopt the topology to the deformation. The basic idea is to construct the Laplacian matrix [29] corresponding to $\mathscr{R}$'s link-node structure and set the two types of corresponding control vertices: vertices in C_s are used as the shape handle vertices, and those in C_d as the depth handle vertices, while nodes in the rest of $\mathscr{R}$ are reconstructed in the least square sense. Let $\mathbf{x}, \mathbf{y}, \mathbf{z}$ represent the $n \times 1$ vectors containing the x, y and z coordinates of $\mathscr{R}$'s vertices. Our system first constructs the Laplacian matrix L based on the connectivity meshes (in our case, linked nodes), and then adds the following equations of control vertices to solve in the least-square sense in $\mathbb{R}^3$:

$$\begin{aligned} \mathbf{x}_s &= x_s, \quad && \mathrm{s} \in C_s \\ \mathbf{y}_s &= y_s, \quad && \mathrm{s} \in C_s \\ \mathbf{z}_d &= z_d, \quad && \mathrm{d} \in C_d \cup \{s_1, s_m\} \ . \end{aligned}$$

While our proposed diagram evolution algorithm can be applied to general topology-preserving curve deformation scenarios, this paper focuses on the six types of Reidemeister-style curve manipulations[2] by restricting the allowed number of vertices in C_d. Figure 3 illustrates the six elementary moves supported in our interface and highlights all possible distribution of shape handle vertices and depth handle vertices during the deformation.

Geometry Reconstruction. After deformation, geometry reconstruction is executed to fix, resolve, and reconstruct the evolved curve's geometry to ensure mathematically valid moves being produced with each touch. The six elementary string moves in our implementation fall into two categories:

1. moves that have crossings resolved after deformation, including the moves that have all crossings removed after deformation (e.g., type I^- in Figure 3(a) and type II^- in Figure 3(c), whose vertices in C_d (in blue or red) turn grey after deformation), and the moves that preserve the same crossing signs after deformation (e.g., type III in Figure 3(e) and $\Delta-$move in Figure 3(f), whose vertices in C_d are moved around and adjusted to the new locations);

2. moves that need to resolve crossings after deformation. For example, type I^+ in Figure 3(b) and type II^+ in Figure 3(d) are adding new vertices to C_d after deformation. (We note that the illustrated type I^+ and type II^+ have appeared in only one of the two possible directions that our interface allows.)

We note that it is geometry reconstruction's task to resolve ambiguity raised by moves in the second category by explicitly query users' decision for a crossing sign. Geometry reconstruction process then walks through the diagram and re-assign the ternary depth values for each vertex involved in the deformation.

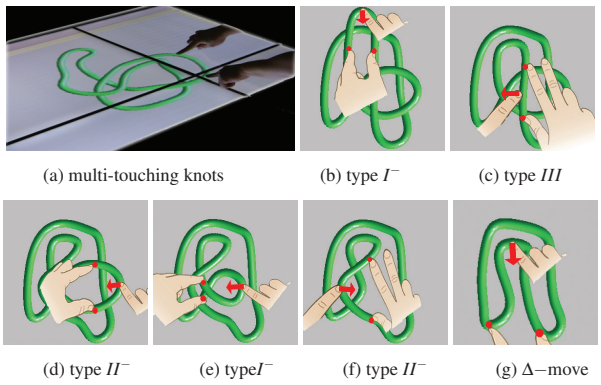


Fig. 4. Unknotting using a sequence of elementary string moves, supported with multi-touch. The multi-touch interface frees the user from manually specifying the ROI before starting deformation and makes possible various types of mathematical evolutions of the ROI simply at fingertips.

[1]Note that in the Reidemeister theorem, each move starts with handle vertices that are *not* on the crossing strands.

[2]The Reidemeister moves can be applied in two directions each, thus the three Reidemeister moves can be decomposed into six elementary moves in total.

3.3 Curve's Evolution in a Time-lapse Form

With the string manipulation interface, we can now transform mathematical experience with knots and curves into step-by-step understandable pieces, and thus can afford insight of topological equivalence through the entire connected interactions. Figure 4 shows an unknotting process where the six types of elementary string interactions are involved in representing the manipulation of a knot. The process begins with the uncrossing (disappearance) of twin crossings at the top (Figure 4(b)), followed by the passage over a crossing (Figure 4(c)), the disappearance of a pair of twin crossing (Figure 4(d)), the disappearance of a small loop (Figure 4(e)), and, finally, the disappearance of a pair of twin crossings (Figure 4(f)).

4 Unfamiliar 4D Example: Movable 4D Topological Surfaces

We now turn to our main objective, which is to create unique mathematical experience for the 4D world that can begin to make the strange more familiar. Our principal test cases are 4D-embedded 2-dimensional manifolds, described by (u,v) parametric space and geometrically shaped to a planar quad (PQ) mesh.

The basic idea is to think of a surface in 4-dimensional space as a collection of "flat" diagrams: we cut the 3D figures of 4D topological surfaces with a stack of 3-dimensional "slices", and the intersection is a stack of diagrams that we can manipulate with the mechanisms we just established (see Figure 1②-⑥). If the surface is positioned *appropriately* with respect to the "slices", the resultant diagrams for these successive cuts will differ at most by one critical change (e.g., a Reidemeister move, a saddle point, or a local maximum or minimum). These cross-sectional pieces, then, are modeled as touchable diagrams to define the evolution of the surface. We next detail the 3D representation of topological surfaces in 4D, the mechanism of positioning the surface and placing cutting "slices", and the interpolation algorithm for evolving 4D surfaces.

4.1 Generating 3D Figures of 4D Topological Surfaces

We used a "flat" style 2D diagram (see Figure 2) to represent curves in 3-space. In this section , a similar scheme will be used to describe a class of 2-manifold deformable objects embedded in 4D. The same "flat" style convention is now applied to these surfaces bounded by curves to schematize surfaces in 4-dimensions.

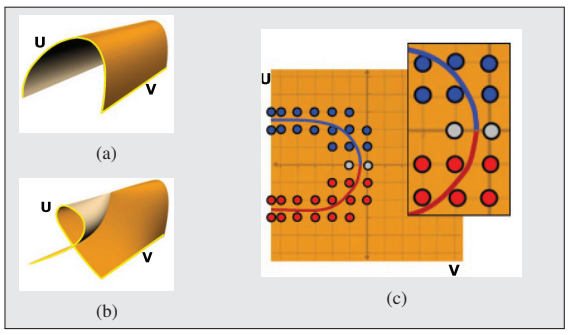


Fig. 5. Turning smoothly embedded 4D surfaces into 3D figures with ternary 4D-depth values. (a) A *flattened* 4D surface that appears just like a 3D surface. (b) With an applied evolution, the surface appears to be self-intersecting in the 3D projection while in fact having no actual intersection of any kind in the 4-space. (c) Crossing diagram drawn on the 4D surface's parametric domain, where each vertexs w-coordinates is mapped to a color-coded balanced ternary value (*blue* is "in front", or nearer the projection point; *red* is "behind", or farther from the projection point; and *grey* means $w = 0$.)

Figure 5 shows a deformable 4D surface patch embedded in four dimensions. Initially the 4D surface has geometric information only in 3-space, which means the surface is flattened in the fourth dimension initially (i.e., each vertex has a 4D "eye coordinate" or depth $w = 0$). Therefore the surface behaves just like a piece of 3D surface (see Figure 5(a)). When evolved in 4-space, the surface can appear to intersect with itself in the projected 3D space (see Figure 5(b)). However, there are no real collisions in 4-space. The crossing-diagram on the the surface's parametric domain, which can be thought of as an unfolded view of the 4D surface (see Figure 5(c)), illustrates the secret: the "north" half of the intersection line is "in front" or, nearer the 4D projection point, and the south part of the intersection line is "behind"; the two parts have no intersections of any kind in 4-space.

Selective vertices are color-coded for 4D depth as shown in Figure 5(c) to illustrate the "flat" style represenation of 4D surfaces. We assign each vertex a ternary "eye-coordinate," or depth w: $w = 1$ for vertices on patches that are "in front" in the fourth dimension, $w = -1$ for vertices that are "behind", and $w = 0$ for vertices on surface patches where there is no self-intersection. Vertices shared by a "front" patch and a "behind" patch are assigned $w = 0$ (see e.g., in Figure 5(c), the two vertices in *grey* near where the blue and red intersection lines meet).

4.2 Elementary 4D-Surface Interactions

Now we extend our string interaction mechanisms to a 4D topological surface (say $\mathscr{M}$). For pedagogical purposes, the boundary curves on these 3D figures are planar and restricted to planes. Just as architectural designers use a few curves to dominate the aesthetic characteristic of shapes (see e.g., [38, 24]), we can prescribe a 4D surface's boundary curve and expect the rest of the surface to blend in.

Boundary Curves as 4D Control Handles. In order to support such curve-based 4D manipulations, we allow the user to select and deform one boundary curve (say $\mathbf{K}$) of a 4D topological surface using the 6 elementary string interactions (see Figure 6). A similar but revised topology reconstruction scheme is used: in 3D curve manipulation scenario we removed a small under-crossing arc to indicate which strand is "nearer" the 3D projection point, now on a 4D control handle we are going to remove a small under crossing arc on the boundary curve to indicate which strand is "nearer" the 4D projection point. Figure 6(a)-(d) list the four basic surface evolutions where the familiar string interactions are being applied to the surface's boundary curves. After the evolution, the vertices on a 4D curve handle are assigned ternary 4D-depth (the *blue* handle vertices are "nearer" to the 4D projection point than those *red* are.)

Interpolating 3D Shapes and 4D Depth. Let $C = \{k_1, k_2, ..., k_n\}$ be the set of indices of all vertices on the boundary curve handle $\mathbf{K}$ that is selected and deformed. C consists of two subsets: C_ϕ, the set of indices of the vertices with either positive or negative 4D-depth; and C_θ, the set of indices of the vertices with zero 4D-depth. The other three boundary curves are considered fixed during 4D deformations, and we use $C' = \{l_1, l_2, ..., l_m\}$ to represent the set of indices of their vertices. Following the same fashion of string deformation, we construct the Laplacian matrix [29] corresponding to $\mathscr{M}$'s mesh structure and set the two types of corresponding handle vertices: vertices in $C \cup C'$ are used as shape handle vertices, and those in $C_\phi \cup C'$ are used as 4D-depth handle vertices, while nodes in the rest of $\mathscr{M}$ are reconstructed in the least square sense. Let $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{w}$ represent the $n \times 1$ vectors containing the x, y, z and w coordinates of $\mathscr{M}$'s vertices. Our system now constructs the Laplacian matrix L, and then adds the following four equations of the control vertices to solve in the least-square sense in $\mathbb{R}^4$:

$$\begin{aligned} \mathbf{x}_s &= x_s, & \mathrm{s} \in \mathrm{C} \cup \mathrm{C'} \\ \mathbf{y}_s &= y_s, & \mathrm{s} \in \mathrm{C} \cup \mathrm{C'} \\ \mathbf{z}_s &= z_s, & \mathrm{s} \in \mathrm{C} \cup \mathrm{C'} \\ \mathbf{w}_d &= w_d, & \mathrm{d} \in C_\phi \cup \mathrm{C'} \ . \end{aligned}$$

4D Geometry Reconstruction. After the rest of the 4D surface blends in, our algorithm walks through the mesh and re-assign the ternary 4D-depth values for each vertex involved in the deformation. Each elementary move triggered by string manipulations on the 4D surface' boundary curve can be thought of as an evolution of the

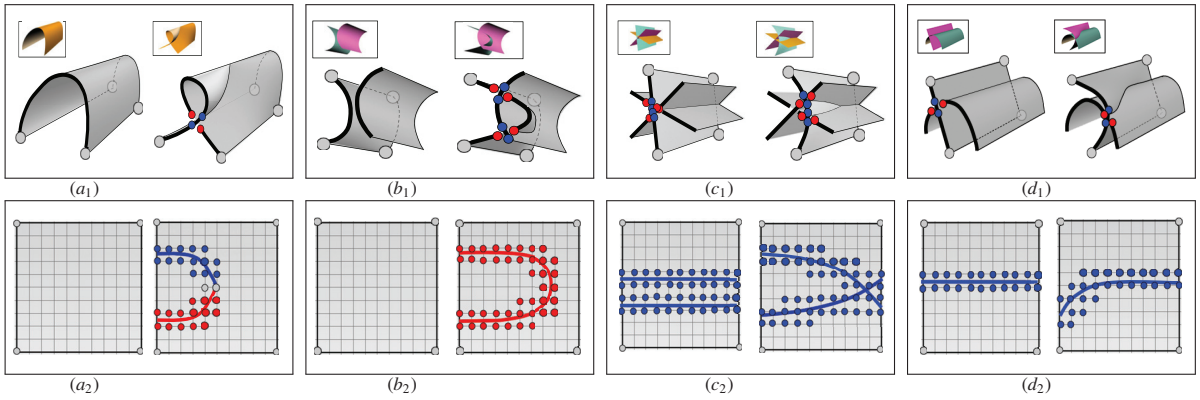


Fig. 6. Applying elementary string evolutions on a 4D surface's boundary curves. First row: the evolutions in geometric space. (a_1) Type I move on boundary curve, which introduces a branch point on the annulus. (b_1) Type II move on boundary curves, generating a minimum or a maximum on the double point curve that appears on the immersed surfaces. (c_1) Type III move on the boundary curves, bringing in a triple point in the surfaces. (d_1) $\Delta-$move applied on the boundary curves, altering the intersection line. Second row ($(a_2)\mapsto(d_2)$): the corresponding evolutions of the crossing-maps in the evolving surface's parametric space.

crossing-map[3] in the evolved surface's parametric space (see Figure 6$(a_2)\mapsto(d_2)$). For example, the 4D evolution in Figure 6(a_1) creates a branch point on the annulus in the geometric space. The evolution can be visualized as an introduction of a $v-$shape crossing-map in the surface's parametric space. The $v-$shape intersection consists of a *blue* north half ("in front" in 4D) and a *red* south half ("behind" in 4D) which do not intersect with one another in 4-space (see Figure 6(a_2). The evolution in Figure 6(a_2) involves two separate surface sheets in geometric space, where the first surface's evolution generates a minimum of the double point curve on the two immersed surfaces. The evolution corresponds to a $u-$shape crossing-map in the evolved surface's parametric space, and the intersection line is completely in *red*. There would be a *blue* counterpart in the second surface's parametric space if visualized. The color pattern indicates the two surfaces do not interact with each other in 4-space, and the intersection in their 3D figures is just an artifact of projection.

4.3 Evolving 4D Topological Surfaces

The key ideas of the overall scenario should now be clear. Our technique for moving 4D topological surfaces is to create a collection of *diagrams* by cutting the object into parallel slices and then separating the slices from each other along a *longitudinal axis* to expose the successive stages of its evolved features and structures. The diagrams incorporating responsive features can help the viewer understand and redefine the shape of the surface along the longitudinal axis. The logical series of modeling steps, the problems they induce, and the ultimate resolution of moving topological 4D surfaces are detailed in the following sections.

Positioning and Slicing the 3D Figures of 4D Surfaces. Just as viewpoint suggestion is important to improve the speed and efficiency of data understanding [6, 23], it is particularly useful that the 3D figure of a 4D topological surface is oriented and sliced "appropriately" so that its important features can be exposed across the generated diagrams. Two rules are adopted when we position the 3D figures:

1. **Suggesting the Longitudinal Axis.** As illustrated in Figure 6, the longitudinal axis orientation is often chosen to align as much as possible with both the direction of the longest extent capable of producing parallel slices and the direction of the maximum exposure of the intersection curves of the 3D figures. We turn the task of choosing the longitudinal axis into a fairly familiar "view selection" problem [6, 23]. We assume the longitudinal axis will be aligned to the z-axis, and cutting planes parallel to the xy-plane, and the viewpoint "goodness" measure we use takes into account the total projected area of the 3D figure, and the total projected length of the 3D intersection lines. The measure, corresponding to surface S projected from a chosen angle c, is given by

$$I(S,c) = \sum_{i=0}^{N_l} \left(\frac{L_i}{L_t} \log \frac{L_i}{L_t} \right) + \sum_{i=0}^{N_a} \left(\frac{A_i}{A_t} \log \frac{A_i}{A_t} \right) .$$

Here L_i represents the projected length of curve segment i and L_t is the total length of the knot curve embedded in 3D; A_i represents the projected area of polygon i and A_t is the total area of projected 3D surface. In our implementation, the measure is minimum, hence better.

[3] A crossing-map maps the surface intersection to the surface's parametric space.

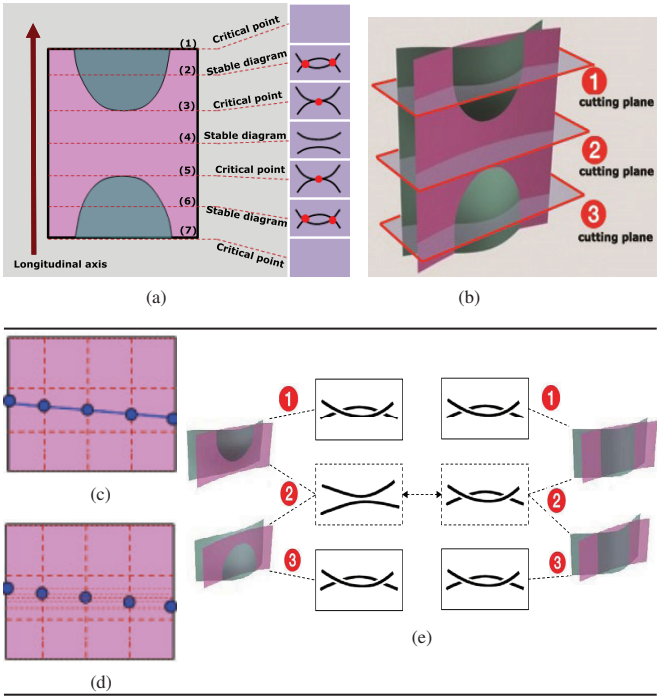
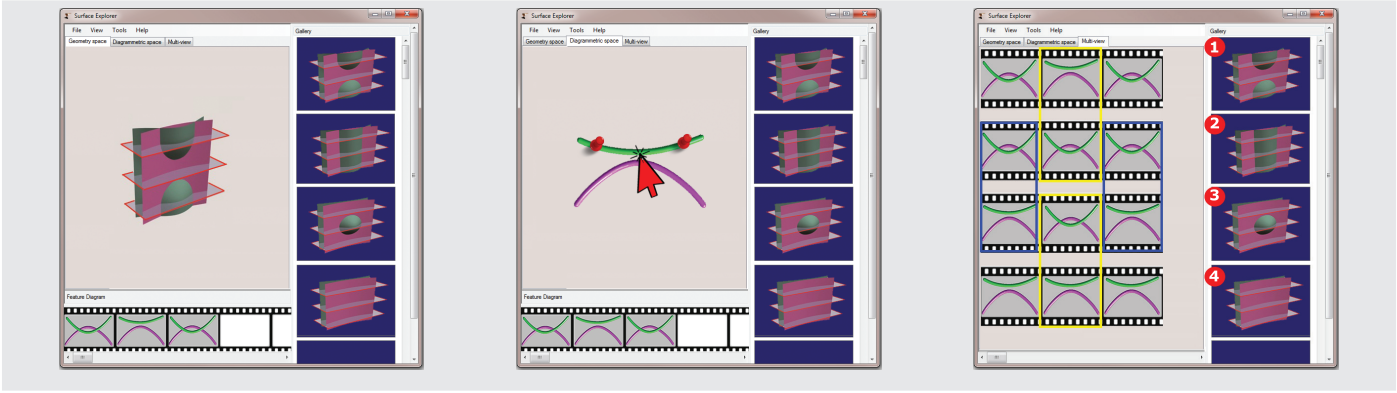


Fig. 7. (a)-(b) Choosing the longitudinal axis and generating diagrams by placing cutting planes. (c)-(d) Snapping the resultant cross-sectional pieces to mesh vertices. In the example, the cutting plane introduces 5 intersection points (c) and thus five horizontal rows are added to the mesh structure to represent the cross-sectional piece using mesh vertices (d). (e): Evolving a 4D surface with a sequence of three interactive diagrams. The evolution of the feature diagram in the middle cutting plane is shared by both the upper half and the bottom half of a double decker set [10].



(a) Geometry space (b) Diagram space (c) 4D evolution in time-lapse form

Fig. 8. Screen images of our interactive 4D visualization interface. The illustrated evolutions that correspond to optimal or saddle critical points of the 1-dimensional double point set of the deformable surfaces. ①↦②: a pair of double point arcs in which the top has a minimal point and the bottom has a maximal point are replaced by a pair of parallel double point arcs that bound two strips. ③↦④: the removal of a closed loop of double points when two surface sheets are pull apart from each other. Both are found in Roseman's list of moves to knotted surfaces [26]. The apparent depth order of the evolved arcs indicates which surface sheet is "in front" or "behind" in the fourth dimension.

2. **Generating the Feature Diagrams.** As pointed out by Carter [8], the cutting planes should be placed along the longitudinal axis so that the resultant diagrams for these successive cuts will differ at most by one critical change (e.g., a Reidemeister move, a saddle point, or a local maximum or minimum). Our basic implementation is inspired by Karpenko and Li's exploded view diagram generation [17, 20]: we identify all the change points — i.e., the points where the number of connected components, self-intersections, or sharp curvature extrema of the surface cross-section changes (Figure 7(a)); we then place cutting planes halfway between each pair of adjacent change points (Figure 7(b)).

Representing Feature Diagrams using Mesh Vertices. Now the 4D surface is described as a sequence of feature diagrams such that successive terms in the sequence differ at most by a critical point. Each feature diagram is the intersection of the surface with a non-critical plane, and typically consists of a number of intersection points (see Figure 7(c)) along the cross-sectional curve meeting the polygon edges on the mesh. Before we manipulate the feature diagram, the feature diagram is turned into a representation using the mesh vertices: if an intersection point is close enough to an existing vertex on the mesh, we replace the intersection with the mesh vertex; otherwise, we re-sample the mesh by adding new rows (or columns) so that we can "link" the intersection to the vertex (see Figure 7(d)).

Evolving 4D Surfaces by Manipulating Feature Diagrams. Now we can evolve the 4D surface by applying the elementary string manipulations on these diagrams along the longitudinal axis, one at each time (see Figure 7(e)). One important key to a successful 4D evolution is that "the breaks (of the diagrams along the longitudinal axis) match up" [8]. Our paradigm can ensure each break match up the rest of cross-sectional pieces to produce a mathematically correct 4D evolution:

1. Type I^-, type II^-, type III, and $\Delta-$ moves on a feature diagram will guarantee to match up the other breaks, (there is indeed only one allowed "break" choice for these moves.) Examples include the elementary moves illustrated in Figure 6(c_1) and Figure 6(d_1).

2. Type I^+ and type II^+ moves introduce two possible "break" choices at the diagram level, however the topology information defined in the neighboring cross-sectional pieces may reduce the uncertainty. In our given example shown in Figure 7(e), the Reidemeister-style interaction on the feature diagram in the middle cutting-plane (labeled as ②) will automatically conform to the topological constraint showing the correct "break": the *shape* manipulation on the feature diagram will deform the corresponding vertices on the surface mesh, and the rest of the surface will just blend in, adopting the correct topology defined in the top and bottom cutting planes (see the "breaks" in Figure 7(e)①③); thus the interpolation produces the correct "break" for the feature diagram being manipulated.

3. If two possible "break" choices do arise when type I^+ and type II^+ moves are applied to a feature diagram, our interface will explicitly query the user's decision to resolve the evolving feature diagram's crossing sign which, in turn, defines the 4D surface's topological structure. Examples include, e.g., the elementary moves illustrated in Figure 6(a_1) and Figure 6(b_1).

The proposed evolutions to our 4D double decker example are summarized and illustrated in Figure 8(c). The evolutions illustrated involve a saddle and a bubble move applied on the double decker set. The evolution from Figure 8(c)① to Figure 8(c)② concerns a type II^+ move applied to the feature diagram in the middle cutting plane; on the immersed surfaces, this diagram manipulation replaces a pair of double point arcs in which the top has a minimal point and the bottom has a maximal point with a pair of parallel double point arcs that bound two strips. Figure 8(c)③↦④ shows the removal of a closed loop of double points when two surface sheets are pulled apart from each other by a fairly familiar type II^- move on the feature diagram in the middle cutting plane.

5 Implementation Environment and Experimental Results

Our user interface (Figure 8) is based on OpenGL and Windows Win-Form API. The software runs on a Dell PC desktop with 3.2GHz Intel Pentium 4 CPU, and can be configured to take multi-touch input to control the virtual cursors with local support of PQ Labs Multi-Touch libraries (see e.g., Figure 4(a)). In Figure 8(b) we show the regular mouse-keyboard input setting for manipulating feature diagrams in a 4D evolution task. The user can place virtual cursors in our diagram interface using a mouse device. Double clicking on a virtual cursor places a push pin to fix the cursor, much as we long press the cursors to indicate the boundary cursors with a multi-touch gesture.

5.1 4D Surface's Evolution in a 2-Dimensional Time-lapse Form

For the user's point, our 4D surface editing interface exploits a 2-dimensional time-lapse form: in one dimension, the 4D entity itself is evolved and tracked in a *gallery* panel of our interface (see e.g., the blue panel in Figure 8(a)); in the other dimension, each entity at

each time point is represented using a collection of feature diagrams, obtained from cutting planes placed along the surface's longitudinal axis.

The user can interactively refine the orientation of 4D surfaces, the placement of cutting planes in the "Geometry space" panel (see e.g., Figure 8(a)), and swtich to the "Diagram space" panel where the interface allows interactive manipulation of the selected feature diagrams (Figure 8(b)). In addition, our interface provides a visual-analysis view to summarize the evolution of a 4D surface using a 2-dimensional time-lapse form (see Figure 8(c)) where the understanding of a 4D surface can be transformed into the analysis based on a 2-dimensional array of feature diagrams.

5.2 Experimental Results

We next introduce a family of use scenarios and application results that have originally motivated and later refined the design of our user interface as well as the kernel implementation. Interacting with 4D surfaces using touchable diagrams can help users to develop a correct interactive experience with the intuitive nature of unfamiliar 4D geometry.

5.2.1 Implementing the Roseman 4D Moves

Our first case scenario concerns a group of "elementary" but rather complicated 4D moves, often referred to as the Roseman 4D moves. Dennis Roseman [26] showed that seven moves are sufficient to move embedded surfaces around in 4-dimensional space. These moves are like the Reidemeister moves for knot diagrams. If two surfaces can be deformed in 4-space into one another and they have given 3-dimensional diagrams that are different, then their diagrams can be transformed into one another by a sequence of the seven Roseman moves.

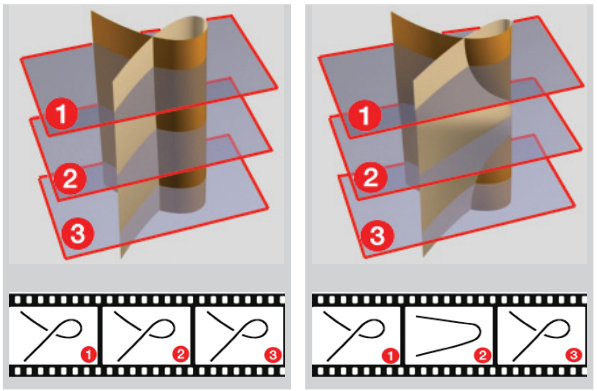


Fig. 9. An interactive visualization implementation of Carter's pictorial analogues shown in Figure 1. Adding ((a)↦(b) or canceling ((b)↦(a) branch points through a saddle move, modeled and implemented by a type *I* move on the second cross-sectional diagram.

Even though the seven Roseman moves are considered as "elementary moves" for evolving 4D topological surfaces, they are indeed rather complicated and involve non-rigid deformations applied to curved surface sheets. Visualizing and implementing the Roseman moves is very difficult with any existing 4D visualization approach and thus has only existed in the mathematical drawing's imagination (see Figure 1). By transforming the 4D evolution into a set of Reidemeister-style diagrams, we can begin to appreciate the Roseman moves. In Figure 9 we show one of the seven Roseman moves that can be modeled and implemented in our interface: the move adds or cancels branch points through a saddle move on the immersed surface, using a type I^+ move on the second cross-sectional diagram obtained with our interface.

5.2.2 Visualizing 4D Structures with Diagrammetric Analogues

The rigors of the higher dimensional worlds are encapsulated within equations, but our intuition is encapsulated in the much simpler figures. Often in topology, we use lower dimensional diagrams to depict phenomena in higher dimensional space. Some lower dimensional analogues can be used to depict 4D structures in an intuitive way.

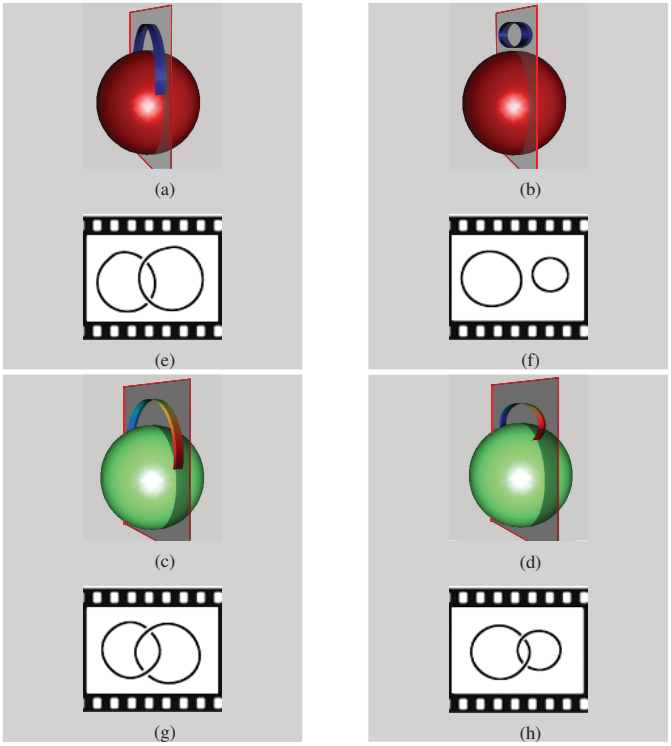


Fig. 10. Visualizing the 4D "chain" structure with lower dimensional diagrammetric analogues.

In Figure 10 we show how diagrammetric analogues can be used to visualize and manipulate the interactions of surfaces in 4D. Figure 10(a) is a "fake" chain structure by threading a ribbon "in front of" a closed 2-sphere embedded in the fourth dimension (the two objects are color-coded for 4D depth); there is no real interaction between the ribbon and the sphere (the ribbon, colored in blue, is "in front" of the 2-sphere, colored in red). The diagrammetric analog shows their structural relationship which is a circle well on top of another one, and of course can be fully de-attached from each other with a type II^- move (see e.g., Figure 10(b)). A real 4D chain structure can be constructed by threading a ribbon into the 2-sphere, first going "above" in 4D, pulling under, and finally coming out the other side "below" (see e.g., Figure 10(c), the ribbon goes inside one side the sphere "blue", and comes out the other side "red"; the "green" sphere remains in the middle in the fourth dimension.) In Figure 10 a $\Delta-$move is applied to the diagrammetric analogues, much as we physically lift the ribbon in four dimensions [33].

5.2.3 Mathematically Deforming 4D Torus

Our third case scenario concerns the 4D embedded torus, an object of fundamental interest in 4D visualization (see e.g., [15, 33]). The 4D torus is the product of two circles, technically written as T^2 with a standard model given by $\mathbf{x}(u,v) = (\cos u, \sin u, \cos v, \sin v)$. One interesting feature of the 4D torus is the conflict between its two lines of self-intersection on the orthogonal 3D graphics projection, and its actual smooth topological structure in four dimensions.

In [33], Zhang et al. suggest vertex-based manipulation where the 4D physically based modeling moves the free vertices to restrict the final 4D mesh to the correct topology. However such physically based

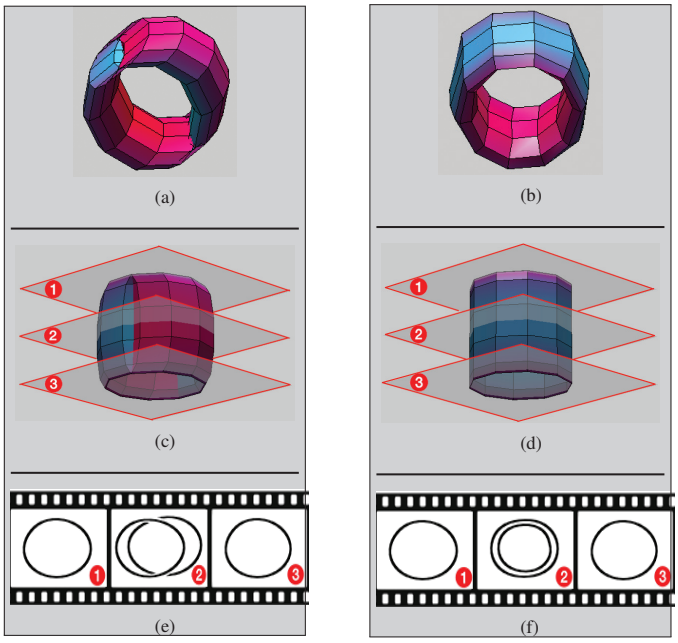


Fig. 11. Applying a mathematical move to the 4D-embedded torus helps expose the underlying topological structure of the surface. The 3D figure of 4D torus is color-coded for 4D depth, so the slight red bulge that remains in the middle is at a completely different depth from the blue surface; thus we can just push the red bulge through the blue surface with Reidemeister moves on the second cutting plane, yielding the final toroidal shape.

interactions can be counter-intuitive and difficult to understand. In absence of visual presentations and appropriate interaction support that can take into account the important mathematical features, the user is often left guessing how to interact with the rather complicated mathematical objects. This makes mathematical sense-making difficult.

In Figure 11(b)↦(e) we showcase our mathematically interactive solution with which we can visualize the actual smooth structure of 4D torus using an interactive and more visually friendly method. The middle cutting plane in Figure 11(c) reveals the visual secret under the torus' skin, which is a *circle times a circle*. The two circles are not entangled with each other, which can be visualized clearly as a feature diagram. Furthermore, a sequence of type *II* moves on the second cross-sectional diagram (see Figure 11(d)) can transform the 4D torus to actually eliminate the self-intersections in the 3D projection by altering the 4D surface within allowed deformation.

6 User Feedback and Discussion

A preliminary user study was performed in order to assess our visualization design and user interface implementation. The objective was to overall evaluate the potential pedagogical usefulness of our interface but also to receive feedback on the fulfillment of our design considerations and gather opinions regarding future development.

User Experience Evaluation. The study included a predominantly qualitative user experience evaluation, complemented by a minor quantitative questionnaire. Twelve people participated in the study, all associated with Zhejiang University of Media and Communication. Their level of expertise and knowledge ranged from one year in graduate program in mathematics to several years of research experience in mathematical visualization and computer interface design. In the study, a demonstration of our user interface was performed, including a step-by-step walk-through of each interface component, and the three major components of our visualization approach (feature diagram generation, surface evolution via interactive diagrams, illustrative visualization generation in a time-lapse form). Two case scenarios are chosen as evaluation tasks, which the participants experimented and interacted independently. The first task asked participants to use our interface to visualize the 4D "chain" structure, and then tell a false link structure from a true one. The second task asked participants to turn the 4D-embedded torus into the more friendly doughnut-shape. These two case scenarios were used because we would like the participants to compare our diagram based interface and Zhang's earlier 4D visualization efforts using physically based approaches [33].

While carrying out the tasks the participants were encourage to "think aloud", and also explain what they are performing. Our main interest was to gain as much information as possible about how the visualization could support visual thinking when users are working on these mathematical reasoning tasks. After the study participants completed a subjective satisfaction questionnaire. The responses were given on a 5-point rating scale: *Strongly unfavorable* (1), *Unfavorable* (2), *Unsure* (3), *Favorable* (4), and *Strongly favorable* (5). The questionnaire covered the following statements:

- Overall impression: the overall impression of the mathematical visualization design and implementation.
- Efficiency for interaction: whether turning 4D manipulation tasks into a sequence of 2D interactions is effective and feasible at all.
- Learnability: ease of learning the visualization interface for a novice user.
- Benefit over single pointer interface: whether the multi-cursor design for such mathematical visualization task is superior to the classic single pointer interface.
- Benefit over pure physical simulation: whether our diagram-driven mathematical interface is superior to the previous physical simulation approaches.
- Mathematical correctness: whether participants noticed non-topology-preserving results during manipulations.

The overall assessment is that our visualization design and interface would be useful in visual mathematics and computer interface design. This is illustrated by the numerical ratings in the post-evaluation questionnaire (see Figure 12). Responses for the six statements have an average score of 4.47, with each statement receiving a clearly favorable rating (Learnability statement received the highest rating of 4.91, Benefit over single pointer interface statement received the lowest rating of 3.83). In addition, most participants agreed that our 4D visualization paradigm provided a clearer interface to help understand how materials interact with each other in the 4-dimensional mathematical world. Several participants commented they liked the idea of controlling curves and strings in a computer interface supporting multi-cursor manipulation and multi-touch interaction, as is close to the real life experience of tying shoelaces and ropes with our manual dexterity.

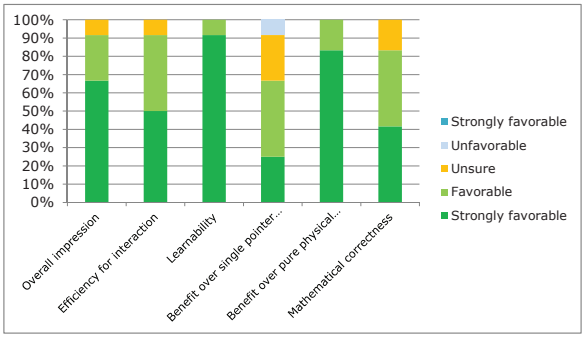


Fig. 12. The quantitative results of the user study questionnaire. 5-point rating scale is used from *Strongly unfavorable* (1) through *Unsure* (3) to *Strongly favorable* (5).

Limitations. Although our 4D visualization interface creates effective results for our principal test cases, our current approach has the following limitations:

- We use the so-called Least-Square mesh for modeling and deforming 4D surfaces. Although this approach is very simple and can be efficiently solved by using a sparse set of control points with geometry, the results only look visually satisfactory when the starting surface is a nice membrane or thin-plate. A potential enhancement is to add constraint shape optimization in freeform modeling framework [7].

- Our computational method to choose longitudinal axis and place cutting planes is still experimental. We still rely on manual adjustment of the axis and planes, with domain knowledges. Furthermore, one family of 4D knots consists of knotted spheres that are formed by spinning a knotted line segment in the fourth dimension to sweep out the surface (an example is the 4D spun trefoil knot.) Our current algorithms for axis finding and cutting-plane placing do not work well for these 4D-spun entities; domain knowledge is again used when we manually choose knotted line segments as feature diagrams.

7 Previous Work on Visualizing 3D Curves and 4D Surfaces

Visualizing curves and surfaces in space by drawing pictures started in the last century when mathematicians began to make special efforts to convey their ideas in a highly visual manner (see e.g., Hilbert [16], Francis [14], Carter [8, 9, 10]). Mathematics books contain many hand-drawn illustrations and diagrams, as well as images of hand-built models executed long before the availability of computer graphics.

The idea of making computer-generated mathematical pictures of curves and surfaces has developed in many directions with the recent advances in computer graphics technology. Wijk's SeifertView [31] visualizes the Seifert surface of knots and links for users to understand their shapes and structures. Banchoff pioneers the use of 3D computer-based projections to study 4D objects [2, 3]. Carter and Brewer generate nicely rendered figures for many most complicated yet beautiful examples in modern topology, such as the depiction of a 2-dimensional sphere turned inside out and manifolds embedded and evolved in 4-dimensional space [11]. Other representative efforts include a variety of ways to render topological surfaces embedded in 4-dimensional space (see, e.g., Banks [4], Roseman [26], Egli [13], Li [19], and Hanson [15]).

Interactive computer graphics has also been introduced to help describe the evolution of mathematical curves and surfaces in space. Weeks' SnapPea software displays and manipulates the over/under crossings of mathematical knots [32]. Scharein's *Knotplot* has been widely used to construct and deform mathematical knots physically in 3D [27]. Zhang's *KnotPad* suggests a computer interface to deform mathematical knots only with Reidemeister moves [37]. Others combine haptic interfaces and 3D graphics to simulate the dynamics of 3D curves and 4D knotted surfaces (see, e.g, Phillips [25], Spillmann [30], and Zhang [34, 33]).

While the existing efforts have produced many interesting visualizations, those generated mathematical figures and the proposed interaction paradigms by themselves are of limited value. For example, 3D figures of 4D mathematical entities often twist, turn, and fold back on themselves, creating many interesting properties behind the surface sheet [14]. On one hand, such self-intersecting surfaces are extremely difficulty to draw and manipulate in a self-explanatory way; on the other hand, these "hidden" properties are the keys to *understand*, *propose*, and *trace* the evolution of a mathematical entity within the allowed deformation. No previous effort, to our knowledge, has drawn into focus the natural coupling of appropriate visual representations and rich interactions with these mathematical properties. What is really needed is an enhancement of the existing visualization paradigms that allows structure-aware exploration of the shape itself and intuition-building interactions built from established understandable pieces. Our work presented in this paper answers this question — "How can we *code* these "hidden" properties into the interactive visualization of mathematical curves and surfaces in space?"

8 Conclusion and Future Work

In this work, we adopt for the most part a visualization researcher's perspective on the techniques and prospects of interactive visualization in descriptive topology, emphasizing those areas of 3D curves and 4D surfaces where interactive paradigms can transform the way of studying and understanding the higher-dimensional complexity. Toward this goal, we have introduced a family of visualization and interaction methods to help understand the important features hidden behind the surface sheets in 4D entities' shadow images. Our interface allows intuitively controlling the representations of 4D-embedded surfaces in their 3D spaces, by decomposing the unfamiliar 4D manipulation task into a sequence of understandable and familiar steps. By combining graphics, graph visualization, sketching based interface, and multi-touch interaction, we not only can make contributions to mathematical visualization by revealing the visual secrets in the classes of geometric and topological problems, but can potentially make a broader impact by addressing the research challenges involved in exploring the space where the flat digital world of surface computing meets the physical, spatially complex, 3D space in which we live. Starting from this basic framework, we plan to proceed to attack more 4D visualization problems such as the interactive manipulation of apparently knotted, but actually unknotted, spheres in 4D. Other planned future work will extend the range of objects for which we can support the interactive visualization of the smooth deformation between Boy and Roman surface, and the evolutions among various 3D figures of the Klein bottle that have given the same surface in 4-space.

References

[1] R. Arnheim. *Visual thinking*. Univ of California Press, 1969.

[2] 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.

[3] T. F. Banchoff. Beyond the third dimension: Geometry, computer graphics, and higher dimensions. *Scientific American Library*, 1990.

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

[5] E. J. Billington. Balanced n-ary designs: a combinatorial survey and some new results. *Ars Combin. A*, 17:37–72, 1984.

[6] U. Bordoloi and H.-W. Shen. View selection for volume rendering. In *Visualization, 2005. VIS 05. IEEE*, pages 487 – 494, oct. 2005.

[7] M. Botsch and L. Kobbelt. An intuitive framework for real-time freeform modeling. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 630–634, New York, NY, USA, 2004. ACM.

[8] J. Carter. *How Surfaces Intersect in Space: An Introduction to Topology*. K & E Series on Knots and Everything. World Scientific, 1995.

[9] J. Carter. *An Exursion in Diagrammatic Algebra: Turning a Sphere from Red to Blue*. Series on Knots and Everything. World Scientific Publishing Company Incorporated, 2011.

[10] J. Carter and M. Saito. *Knotted Surfaces and Their Diagrams*. Mathematical Surveys and Monographs. American Mathematical Society, 1998.

[11] J. S. Carter. Reidemeister/roseman-type moves to embedded foams in 4-dimensional space. 2012.

[12] J. Cohen, L.Markosian, R. Zeleznik, J. Hughes, and R.Barzel. An interface for sketching 3d curves. In *SI3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 17–21, New York, NY, USA, 1999. ACM Press.

[13] R. Egli, C. Petit, and N. F. Stewart. Moving coordinate frames for representation and visualization in four dimensions. *Computers and Graphics*, 20(6):905–919, November–December 1996.

[14] G. Francis. *A topological picturebook*. Springer-Verlag, 1987.

[15] A. J. Hanson and H. Zhang. Multimodal exploration of the fourth dimension. In *Proceedings of IEEE Visualization*, pages 263–270, 2005.

[16] D. Hilbert and S. Cohn-Vossen. *Geometry and the Imagination*. AMS Chelsea Publishing Series. AMS Chelsea Pub., 1999.

[17] O. Karpenko, W. Li, N. Mitra, and M. Agrawala. Exploded view diagrams of mathematical surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1311–1318, Nov. 2010.
[18] S. Klimenkol, I. Nikitin, M. Göbel, and H. Tramberend. *Visualization in topology: assembling the projective plane*. Springer, 1997.
[19] B. Li, X. Zhao, and H. Qin. 4-dimensional geometry lens: A novel volumetric magnification approach. *arXiv preprint arXiv:1307.0147*, 2013.
[20] W. Li, M. Agrawala, B. Curless, and D. Salesin. Automated generation of interactive 3d exploded view diagrams. *ACM Trans. Graph.*, 27(3):101:1–101:7, Aug. 2008.
[21] N. G. Lipari and C. W. Borst. Evaluation of stereoscopy and lit shading for a counting task in knot visualization. In H. R. Arabnia, editor, *CGVR*, pages 23–29. CSREA Press, 2007.
[22] C. Livingston. *Knot Theory*, volume 24 of *The Carus Mathematical Monographs*. Mathematical Association of America, Washington, U.S., 1993.
[23] E. Moncls, P.-P. Vzquez, and I. Navazo. Efficient selection of representative views and navigation paths for volume data exploration. In L. Linsen, H. Hagen, B. Hamann, and H.-C. Hege, editors, *Visualization in Medicine and Life Sciences II*, Mathematics and Visualization, pages 133–151. Springer Berlin Heidelberg, 2012.
[24] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. Fibermesh: designing freeform surfaces with 3d curves. *ACM Trans. Graph.*, 26(3), July 2007.
[25] J. Phillips, A. M. Ladd, and L. E. Kavraki. Simulated knot tying. In *ICRA*, pages 841–846. IEEE, 2002.
[26] D. Roseman. Twisting and turning in four dimensions, 1993. Video animation, Department of Mathematics, University of Iowa, and the Geometry Center.
[27] R. G. Scharein. *Interactive Topological Drawing*. PhD thesis, Department of Computer Science, The University of British Columbia, 1998.
[28] S. Schmidt, M. A. Nacenta, R. Dachselt, and S. Carpendale. A set of multi-touch graph interaction techniques. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 113–116, New York, NY, USA, 2010. ACM.
[29] O. Sorkine and D. Cohen-Or. Least-squares meshes. In *Proceedings of Shape Modeling International*, pages 191–199. IEEE Computer Society Press, 2004.
[30] J. Spillmann and M. Teschner. An adaptive contact model for the robust simulation of knots. *Comput. Graph. Forum*, 27(2):497–506, 2008.
[31] J. van Wijk and A. Cohen. Visualization of the genus of knots. In *Visualization, 2005. VIS 05. IEEE*, pages 567 – 574, oct. 2005.
[32] J. Weeks. Snappea: a computer program for creating and studying hyperbolic 3-manifolds, 2001.
[33] H. Zhang and A. Hanson. Shadow-driven 4d haptic visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1688–1695, 2007.
[34] H. Zhang and A. J. Hanson. Physically interacting with four dimensions. In G. Bebis, R. Boyle, B. Parvin, D. Koracin, P. Remagnino, A. V. Nefian, M. Gopi, V. Pascucci, J. Zara, J. Molineros, H. Theisel, and T. Malzbender, editors, *ISVC (1)*, volume 4291 of *Lecture Notes in Computer Science*, pages 232–242. Springer, 2006.
[35] H. Zhang, S. Thakur, and A. J. Hanson. Haptic exploration of mathematical knots. In *ISVC (1)*, pages 745–756, 2007.
[36] H. Zhang, J. Weng, and A. Hanson. A pseudo-haptic knot diagram interface. In *Proc. SPIE 7868*, volume 786807, pages 1–14, 2011.
[37] H. Zhang, J. Weng, L. Jing, and Y. Zhong. Knotpad: Visualizing and exploring knot theory with fluid reidemeister moves. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2051 –2060, dec. 2012.
[38] X. Zhao, C.-C. Tang, Y.-L. Yang, H. Pottmann, and N. Mitra. Intuitive design exploration of constrained meshes. In L. Hesselgren, S. Sharma, J. Wallner, N. Baldassini, P. Bompas, and J. Raynaud, editors, *Advances in Architectural Geometry 2012*, pages 305–318. Springer Vienna, 2013.