Reprinted from
Journal of Imaging Science and Technology® 65(6): 060410-1–060410-11, 2021.

Visualizing and Slicing Topological Surfaces in Four Dimensions

Huan Liu and Hui Zhang
University of Louisville; Louisville, KY 40292
E-mail: hui.zhang@louisville.edu

Abstract. *Smooth topological surfaces embedded in 4D create complex internal structures in their projected 3D figures. Often these 3D figures twist, turn, and fold back on themselves, leaving important properties behind the surface sheets. Triangle meshes are not well suited for illustrating such internal structures and their topological features. In this paper, we propose a new approach to visualize these internal structures by slicing the 4D surfaces in our dimensions and revealing the underlying 4D structures using their cross-sectional diagrams. We think of a 4D-embedded surface as a collection of 3D curves stacked and evolved in time, very much like a 3D movie in a time-elapse form; and our new approach is to translate a surface in 4-space into such a movie — a sequence of time-lapse frames where successive terms in the sequence differ at most by a critical change. The visualization interface presented in this paper allows us to interactively define the longitudinal axis, and the automatic algorithms can partition the 4D surface into parallel slices and expose its internal structure by generating a time-lapse movie consisting of topologically meaningful cross-sectional diagrams from the representative slices. We have extracted movies from a range of known 4D mathematical surfaces with our approach. The results of the usability study show that the proposed slicing interface allows a mathematically true user experience with surfaces in four dimensions.* © *2021 Society for Imaging Science and Technology.*
[DOI: 10.2352/J.ImagingSci.Technol.2021.65.6.060410]

1. INTRODUCTION

Our work concerns 2-dimensional surfaces smoothly embedded in 4-space, a basic class of fundamental interest in descriptive topology. The essential difference of these 4D entities from their 3D counterparts is that each vertex of the surfaces has a 4D "eye coordinate," or depth w, in addition to the coordinates (x, y, z) in their 4D projection to 3D. Surfaces embedded in 4D play many roles in analogs to those of our familiar curves in 3D [1]; for example, spheres are the analogs of closed curves, and knots can be generalized to "knotted surfaces" (closed 2D surfaces embedded in 4D) [2].

Challenges arise when we try to visualize these 4D surfaces (when triangulated). Just as 2D shadows of 3D curves lose structure where lines cross, 3D graphics projections of smooth 4D topological surfaces are interrupted where one surface intersects another. Furthermore, many 4D surfaces are constructed by 3D curves spun around a plane in 4D, and their 3D figures often leave important properties behind the surface sheets. Most existing 4D visualization efforts employ a projection to 3D as a fundamental step, and exploit visual or haptic cues (see, e.g., [3, 4]) to help the viewer to identify salient global features of the four-dimensional object.

We, in this paper approach the problem from mathematicians' point of view, by utilizing computer graphics and automatic algorithms to generate topological illustrations that can potentially help depict these unfamiliar surfaces embedded in space beyond 3D. Our work is motivated by the *movie* description adopted in Carter's book *How Surfaces Intersect in Space* [5]. In this book, Carter thinks of 4-dimensional space as a pile of 3-dimensional spaces, stacked in time; and a surface in 4-dimensional space as a collection of curves in 3-dimensional space. That is, each 3-dimensional cutting plane in 4-space cuts a surface in 4-dimensions, and the intersection is a collection of closed curves that can reveal the internal structures behind the surface in their 3D projections. For example, in Figure 1 seven representative cutting planes were shown to create the topologically meaningful cross-sectional diagrams of a Klein bottle [6]. These seven diagrams in the progressing form (very much like a flip-book animation), describe the underlying structure of the Klein bottle, a one-sided surface which, if traveled upon, could be followed back to the point of origin while flipping the traveler upside down.

In this paper we take the first steps towards this goal by designing a slicing interface that is able to automatically compute the longitudinal axis and cutting planes to generate movies for us to analyze these surfaces in 4D Euclidean space.

2. RELATED WORK

Traditional techniques for visualizing surfaces in 4D typically involve creating visual pictures of 4D entities intersecting in a 3D projection and associating the fourth dimension (i.e., the w "eye coordinate") with visual cues such as 4D depth color, texture density, etc (see e.g., [3, 4, 7]). Other representative efforts include a variety of ways to render 4D objects (see e.g., Banks' interactive manipulation and display of surface in 4D [8], Chu's use of 4D light sources to render 4D surfaces [9], Noll's parallel and perspective projections of four-dimensional hyper-objects rotating in four-dimensional space [10], and Zhang's cloth-like modeling and rendering of 4D surfaces [11]).

Figure 2 shows some of the typical 4D visualization techniques. Fig. 2(a) is the 3D graphics projection of a 4D spun trefoil knot [12], with surface color keyed to 4D depth

Received Aug. 7, 2021; accepted for publication Nov. 19, 2021; published online Dec. 7, 2021. Associate Editor: Thomas Wischgoll.
1062-3701/2021/65(6)/060410/11/$25.00

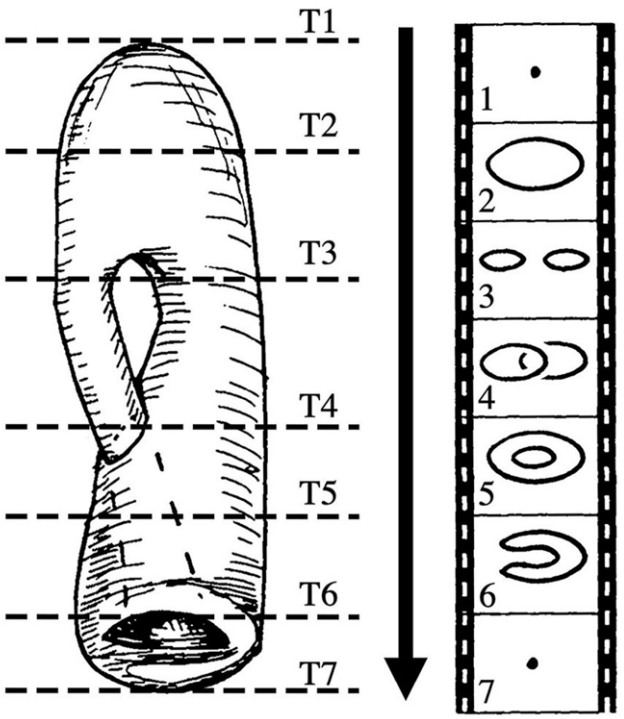


Figure 1. A movie description to describe the underlying structure of a Klein bottle (modified from Carter's book [5]). Successive frames in the movie differ at most by one critical change.

relative to the projection center. The 3D figure contains massive intersections in 3D, especially behind the boundary surface. Transparency is applied in Fig. 2(b) to help the viewer to perceive the internal structure — a trefoil knot spun about a plane in 4D. In prior visualization efforts, the most common techniques for exposing the internal structure of 4D surfaces projected to 3D also include the use of cutaway view (e.g., see Fig. 2(d)) and "banded view" (see e.g., Fig. 2(e)), a different cutaway view by removing just alternating bands from the surface to help the viewer to see through the surface [13]. These images are undoubtedly important for understanding the complex spatial relationships and overall structures, but they provide limited value of helping us understand the underlying structures and important features behind the surface sheet in a 3D projection. It may also result in confusion of user when the visual evidence is difficult to interpret. For instance, making the entire surface semi-transparent is arguably the simplest technique for (partially) exposing occluded portions of the surface. While the transparency view can sometimes convey the internal structure behind the surface sheet, it is nearly impossible for viewers to distinguish and interpret the structure in regions where multiple semi-transparent layers overlap. Similarly, cutaway and banded views are both limited when applied to surfaces having very complex internal structures (e.g., the 4D knotted spheres).

To visualize the internal structures behind a closed surface, Karpenko et al. proposed the use of exploded views by partitioning the surface into parallel slices, along a linear explosion axis [14]. This approach is limited to 3D surfaces only. Carr et al. introduced a method to compute the contour tree of a surface in arbitrary dimensions [15]. The contour tree is a graph that tracks contours at the levels as they split, join, appear, and disappear. While this work focuses on only the topological properties of a surface, it is difficult to visually map between the surface's geometric shape and its topological structure. Edelsbrunner et al. suggest that *Reeb graph* [16] can be used to indicate the topological structure of a surface by locating all saddle points along a longitudinal axis. Similar to Carr's work, this work mainly focuses on how surfaces split, join, appear, and disappear, and does not address topological deformations.

We are thus motivated to design and implement a new visualization paradigm that can summarize and visualize the interior structure behind the 4D surface sheet in its 3D projection. The basic idea is to slice the 4D surfaces, in dimensions that we define. Imagine we have an infinite number of 3-dimensional cutting planes in 4-space that cut the knotted sphere in Fig. 2(e) in parallel. The intersection of these cutting planes and the knotted sphere will usually be a closed curve (or curves), except for the two cutting planes that intersect the spun knot at a point, one on the north pole and the other on the south. When the cutting plane intersects a 4D surface at one single point (like on the north or south pole), the surface has a critical point — often occur when the curve of intersection have points of tangency, or sometimes three sheets of the surface meet at a triple point. Now if we examine the cutting planes between successive critical points and the resultant intersections on these planes, and from them we extract the most representative cross-sections, we will get the seven cross-sections in Fig. 2(f) that help to describe the spun knot's underlying structure. With the advent of modern interactive graphics technology and automated algorithms we can begin to appreciate the challenge of depicting such surfaces embedded in high dimensions by slicing and visualizing their cross-sections, that had only existed in the hand-drawn diagrams in Refs. [5] and [17].

3. OVERVIEW OF SLICING-BASED VISUALIZATION TOOL

From the user's point, our interface is a slicing-based visualization tool, and it consists of a control panel and a visualization area. With this tool, the user can load, transform, and view the 3D figures of 4D surfaces with various desired settings and parameters. For example, the user can choose the sub-space and desired parameters to generate the 3D picture of the 4D surface, and can toggle between transparency view, cutaway view, and "banded view" when creating the movie description.

More importantly, the tool provides user interface elements for users to slice the surface and look into the structures behind the surface sheet. One can interactively place a longitudinal axis for the system to automatically place the slices and render the movie to describe the underlying structure of the 4D surface (see Figure 3). In

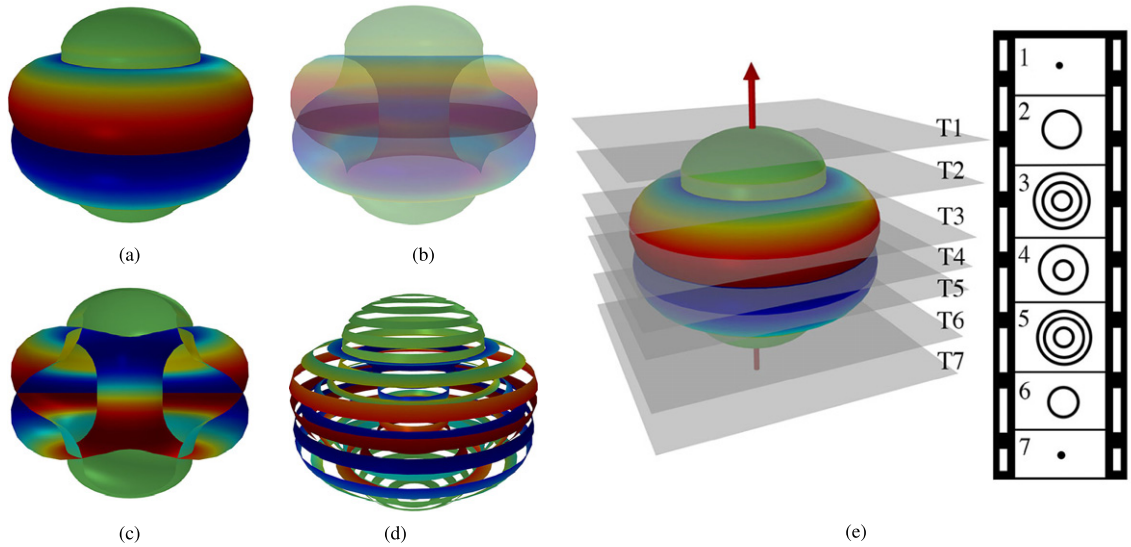


Figure 2. Various approaches to visualizing a 4D spun trefoil knotted surface. (a) A 4D depth colored 3D figure. (b) Applying semi-transparency to the 4D depth colored 3D figure of the 4D spun. (c) A cutaway view to expose the structure and intersections behind surface sheet. (d) A banded view. (e) A movie description of the knotted sphere — a collection of 7 cross-sectional diagrams that describes the spun knot's underlying structure.

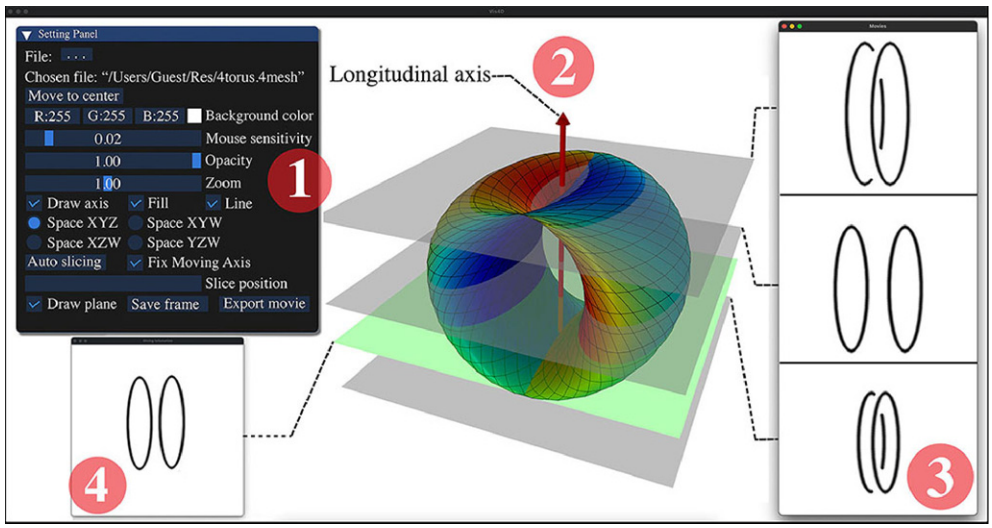


Figure 3. System screen and major interface elements of our slicing-based visualization tool. ① — the toolbox to configure the slicing-based visualization tool, user interface elements including e.g., model file dialogue, slider to set opacity level, radio button control to choose slicing mode (automatically or manually), and checkbox control to choose 3D projection (i.e., *(x,y,z)*, *(x,y,w)*, *(y,z,w)*, or *(x,z,w)*). ② — central visualization panel for one to position longitudinal axis, cutting planes, and to view the slicing results. ③ — movie outcome that contains the optimal cross-sections to represent the surface's interior structure. ④ — cross-section viewer where one can define an arbitrary slicing window and observe the intersection.

addition, the user can freely position a cutting plane in our interface and examine the resultant intersection — the cross-sectional diagrams to illustrate the surface's topological features behind the sheet.

The task of choosing an appropriate longitudinal axis is significant. A slightly different longitudinal axis may result in very different movie descriptions from the same surface sheet. A movie might lead to user confusion, when it is too lengthy and the intersection is difficult to make sense of. Our interface can suggest the optimal longitudinal axis which requires the least number of cross sectional pieces to render the movie description.

4. IMPLEMENTATION DETAILS

In this section, we describe the families of models used to implement the interaction procedures, visual elements, slicing interface, and the automated algorithm to compute the longitudinal axis and to extract and render the resultant movie. Our fundamental techniques are based on a wide variety of prior art, including the use of exploded view in surface and volume visualization [14, 18, 19], algorithms for 3D triangle mesh slicing [20, 21], and other variants on computer graphics and visual interfaces for mathematical visualization including, e.g., the work of [22] and [23].

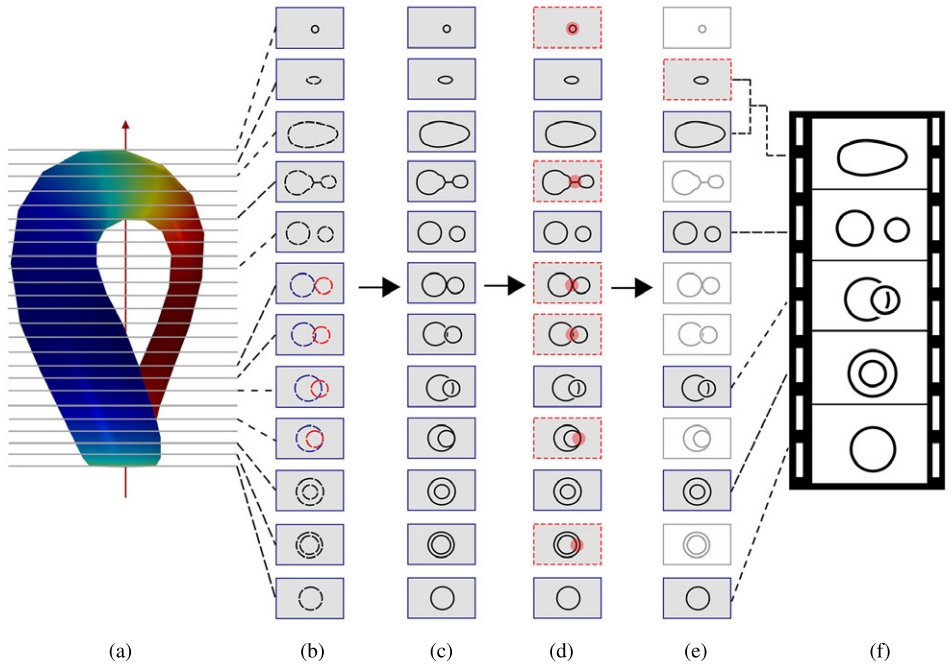


Figure 4. Logical steps involved in the process of rendering a movie for the Klein Bottle. (a) Place longitudinal axis and slicing planes. (b) Obtain resultant intersections. (c) Reconstruct the closed loops. (d) Identify critical changes and associated frames. (e) Selecting key frames for rendering the movie. (f) The resulting movie.

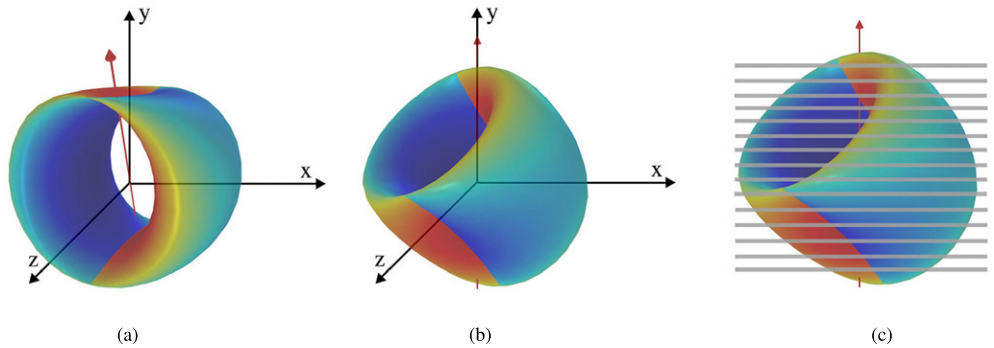


Figure 5. Place longitudinal axis and position cutting planes. (a) An arbitrary longitudinal axis placed around the mass center of the surface's 3D figure. (b) Rotate the surface until the chosen longitudinal axis coincides with *y*-axis. (c) Place cutting planes densely along the longitudinal axis.

Our goal in this section is to interpret a surface in 4-dimensional space as a collection of diagrams in 3-dimensional space, stacked in time. Before we lay out details of our interaction procedures and algorithms, some definitions are in order.

- **Height function and 3D Projection.** When slicing and exposing the interior structures of a 4D surface with "flat" cross-sectional images, we need a generic height function $R^4 \rightarrow R$ to define the spatial relationship between points on the projection. In our implementation, the height function also determines how the projection from 4D to 3D is defined. We use the hyper-plane perpendicular to the direction of the chosen height function to create the 3D projection. While arbitrary height function can be defined and used, in our solution we fix the height function to align with the w eye coordinate and thus 4D entities are projected orthogonally into the xyz sub-space. The user can still "change" the projection direction (the height function), with an intuitive rolling-ball interface to apply 4D rotation to transform the 4D entities in their full dimensional space [3].
- **Longitudinal Axis and Cutting Plane.** When a surface is projected and displayed in the 3D space, each vertex in the 3D figure has a w coordinate in addition to the three-dimensional coordinates (i.e., x,y, and z). In our slicing-based visualization approach, we think of the surface in 4D as a collection of cross-sectional diagrams, stacked in time. To extract the topologically meaningful cross-

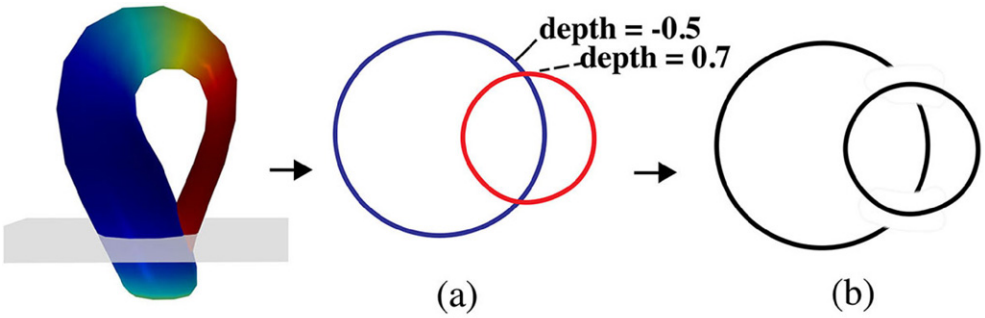


Figure 6. Reconstruct the close loops from line segments in the intersections. (a) A cutting plane slices a Klein Bottle, and the resultant intersections have two collection of line segments that appear to overlap each other, with different w coordinates. (b) We string line segment with each other. Based on their w coordinates, these line segments are stringed into two close loops.

sections, we need to define a longitudinal axis time (i.e., the *time*) and the cutting planes (i.e., the *cross — sections*). The longitudinal axis and the cutting plane are perpendicular to each other. For example, the red arrow in the central area of Fig. 3 is the longitudinal axis, and the cutting planes are perpendicularly positioned along the longitudinal axis. The longitudinal axis defines the direction of the densely positioned parallel cutting planes, from which the algorithm proposed in this work will extract the most meaningful slices and their diagrams as the 4D surface's topological illustration.

4.1 *Creating the Movie — Slicing based Visualization*

The key ideas of the overall scenario should now be clear. The logical series of modeling steps, the problems they induce, and the ultimate resolution of the problems are as follows:

- *Place the longitudinal axis and position the cutting planes.* To ensure the inclusion of all cross-sections from which we will extract the movie, cutting planes are position densely perpendicular to the longitudinal axis (see e.g., Figure 4(a)).
- *Compute the intersections.* Each cutting plane will slice the 3D figure of the 4D entity, leaving intersections on the plane. The intersections is the key to our understanding of the underlying 4D structure (see Fig. 4(b)).
- *Reconstruct the closed loops.* We think of a 4D surface as a collection of evolving closed loops, stacked in time. Intersections in the format of points and line segments will be recognized and closed loop(s) will be reconstructed in this step (see Fig. 4(c)).
- *Identify critical changes and removing associated transitioning frames.* The third step is to locate and remove critical time points. These time points often reveal critical changes in movie content. These time points are removed from the candidate elements and the candidate frames are divided into different sections according to the critical time points (see Fig. 4(d)).
- *Choose the optimal frames between critical changes and render the movie.* Frames between successive critical changes undergo minor changes. In the last step, we identify the best frame from each partition divided by successive critical changes, and the remaining frames now are rendered into the movie (see Fig. 4(e) and (f)).

4.1.1 *Compute the Intersections*

Computing intersections of the surface and the parallel cutting planes is an essential part of our resolution of the problems. Figure 5 shows the basic idea — when the longitudinal axis is determined, we first rotate the surface so that its longitudinal axis coincides with the y-axis in the world coordinate system (see Fig. 5(a) and (b)). This transformation facilitates computation as the cutting planes are now parallel to the xy plane, and we can simply check on each triangle's y coordinate range to determine the collection of the cutting planes it intersects with. The cutting planes are placed densely along the longitudinal axis (see Fig. 5(c)). Intersections of cutting planes with the surface are in the format of points and line segments. To improve the computational efficiency, our implementation adopted several methods introduced in Ref. [20], including the sorting and grouping of the triangle meshes, and the binary search method to quickly identify triangles for intersection computing. It is worth noting that the line segments are disordered and their endpoints still carry the w coordinate.

4.1.2 *Reconstruct the Closed Loop(s)*

The next step is to string the disordered line segments from the raw intersections into one or more closed polygons. Since our principle test cases are closed surfaces embedded in 4D, the line segments in the raw intersection should form one or more closed loops. For example, in Figure 6 the cutting plane slices through the Klein Bottle, and the raw intersections are shown in Fig. 6(a), the collection of the line segments in the intersections can be stringed into two close loops since they exhibit different w eye coordinates (Fig. 6(b)).

4.1.3 *Identify Critical Changes*

The densely positioned cutting planes slice the surface, and that results in a large number of cross-sectional frames. Most of the changes between successive frames are small from

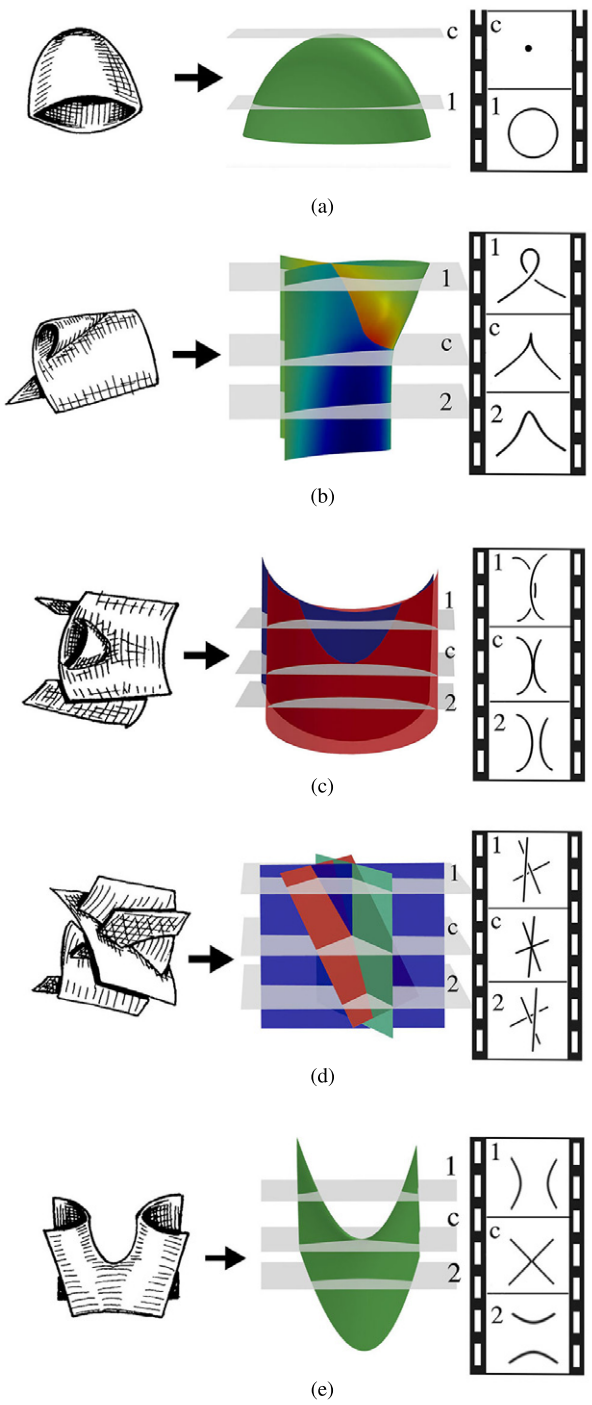


Figure 7. The five types of critical changes. (a) The birth/death of a component. (b)-(d) Changes corresponding to the three Reidemeister moves. (e) The fusing/fissuring change.

a topological perspective. Occasionally, the frames undergo significant changes such as the appearance or disappearance of a new close loop, the change in the number of crossings in the diagrams, or a Reidemeister type of move. In Ref. [5] Carter summarizes five critical changes we might encounter over the cross-sectional frames sliced from the surfaces.

- Type 0: The birth/death of a simple closed curve. As depicted in Figure 7(a), frame c is introduced when the cutting plane is tangent to the surface, and frame c is the transitioning frame leading to the birth of the closed loop in the next frame.
- Type *I*: The critical change introduced by a type *I* Reidemeister move. As shown in Fig. 7(b), the preceding and succeeding frames of the critical change frame c appear to have undergone a type *I* Reidemeister move. This critical change also has led to the addition/reduction of one crossing.
- Type *II*: The critical change introduced by a type *II* Reidemeister move. For example, the preceding and succeeding frames of the critical change frame c in Fig. 7(c) appear to have undergone a type *II* Reidemeister move, which also has led to addition/reduction by two crossings.
- Type *III*: The critical change introduced by a type *III* Reidemeister move. For example, in Fig. 7(d), the preceding and succeeding frames of the critical change frame c appear to have undergone a type *III* Reidemeister move.
- Type *IV*: The operation of fusing two components into one, or fissuring one component into two. As depicted in Fig. 7(e), frame c is introduced when the cutting plane is tangent to the surface. The preceding and succeeding frames of the critical change frame c appear to have undergone a fusing/fissuring operation.

Our next task is to identify the transitioning frames that correspond to the critical changes illustrated in Fig. 7. These frames can be geometrically identified — they are the transitioning frames between other frames where the diagrams are in *safe position* [23]. A diagram is in a safe position when it fulfills the following conditions:

- No two non-adjacent edges in the components are closer than a threshold distance d_{close};
- No two crossings in the components are closer than a threshold distance d_{close};

As indicated in Fig. 7, when these five types of critical changes occur, the diagrams on the transitioning frames are turning into unsafe position (see e.g., Figure 8), until the critical change is accomplished. The unsafe positions corresponding to the five types of critical changes are summarized in Fig. 8, where we highlight the the unsafe regions which our system detects and recognizes as critical changes.

The system scans all cross-sectional diagrams and checks whether they are in a safe or unsafe position corresponding to a critical change. A series of consecutive unsafe frames are considered by the system as critical time points. The system locates all the critical time points (see Fig. 4(d)), and removes these unsafe frames representing the critical time points from the candidate frames. The remaining candidate frames are divided into sections separated by the (removed) critical time points.

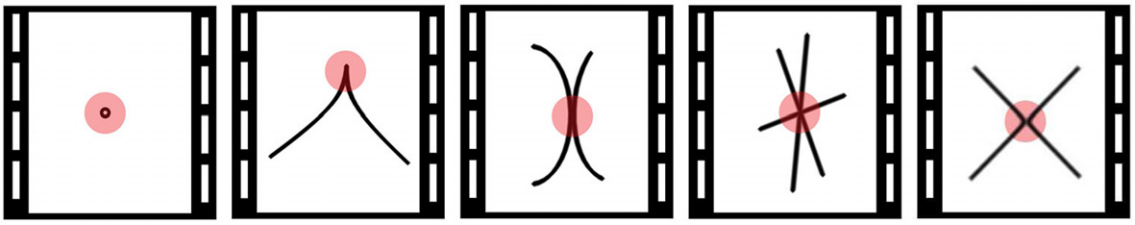


Figure 8. Frames containing unsafe diagrams correspond to critical change moments. (a) Corresponds to critical change 0 in (Fig. 7(a)). (b) Corresponds to critical change *I* in (Fig. 7(b)). (c) Corresponds to critical change *II* (Fig. 7(c)). (d) Corresponds to critical change IV (Fig. 7(d)) and (e) is corresponding to critical change V (Fig. 7(e)).

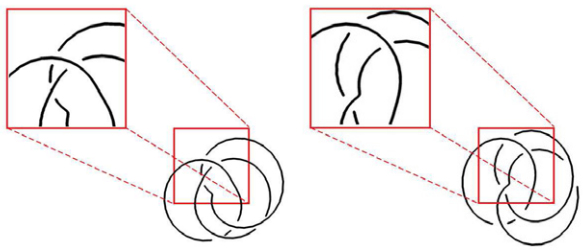


Figure 9. Select optimal frame by comparing the sum of minimal distance between candidate frames.

4.1.4 *Best Frame Identification in One Section*

The final step is to identify the best-shaped frame in each section (separated by the critical changes) to represent underlying structure of the surface sheet corresponding to the section. The system selects the optimal frame from each section to generate the final movie. In our implementation, the comparison metric considers the following factors:

1. **Crossing Distance.** The sum of minimal crossing distance is used to determine which frame is more relaxed (preferred). For a given frame, we find all the crossings, $C_i (i = 1, 2, \ldots, n)$, the distance of C_i and C_j is defined as $D(i, j)$, then calculate the minimum value of the distance from each crossing to all other crossings $M(i) = \min(D(i, 1), \ldots, D(1, i-1), D(i, i+1), \ldots, D(i, n))$, and then sum up the minimum distance of each crossing to get the sum of minimal distance $S = \sum_{i=1}^{n} M(i)$. The system prefers the frame with greater crossing distance. For example, in Figure 9, the sum of minimal crossing distance of frame in Fig. 9(a) is lesser than that of Fig. 9(b), which means the frame in Fig. 9(b) is more "relaxed" than the other, so it is preferred.
2. **Length.** Suppose two frames have the same sum of minimal crossing distance or have fewer than two crossings, the system counts the length of all polygons in two frames, and the frame with greater length is preferred.
3. **Convexity.** In the case where the sum of the minimum crossings is equal or the number of crossings is less than 2, the frame with the more "convex" polygon is preferred. The convexity of a polygon can be calculated as in Algorithm 1. For example in Figure 10, convexity of the cross-sectional diagram in Fig. 10(a) is greater than Fig. 10(b), so the system would prefer the cross-section in (a). In this paper, we compare both the perimeter and convexity to determine the frame with the best shape if the sum of the minimum intersection distances is the same or the number of intersections is less than 2. By default, the weight of length is 0.7 and the weight of convexity is 0.3.

Algorithm 1: Polygon Convexity Calculation

```
Input  : Vertex list vertices[] of length n
Output: Convexity
i = 0
Convexity = 0
while i < n do
    v1 = vector(vertices[i], vertices[(i+1) mod n])
    v2 = vector(vertices[i], vertices[(i+2) mod n])
    if v1 × v2 > 0 then
        Convexity = Convexity + 1
    end
    if v1 × v2 < 0 then
        Convexity = Convexity − 1
    end
    i = i + 1;
end
Convexity = |Convexity| / n
```

After the system identifies the best frame from each partition of the original movies and renders them into a movie, as Fig. 4(e) and (f) shows, the movie contains the minimum number of frames that are topologically meaningful to describe the underlying structure of the 4D surface.

4.2 *Automatically Computed Longitudinal Axis*

Given a specific longitudinal axis, our system can generate a movie to describe the surface's structure, with a minimum number of frames that are topologically representative. Choosing an optimal longitudinal axis is a significant task. A slightly different longitudinal axis can possibly result in a very different or an unnecessarily complicated movie. The movies can differ in the number of frames, the contents of the frames in the movie. In this section, we discuss an algorithm to automatically compute the optimized longitudinal axis

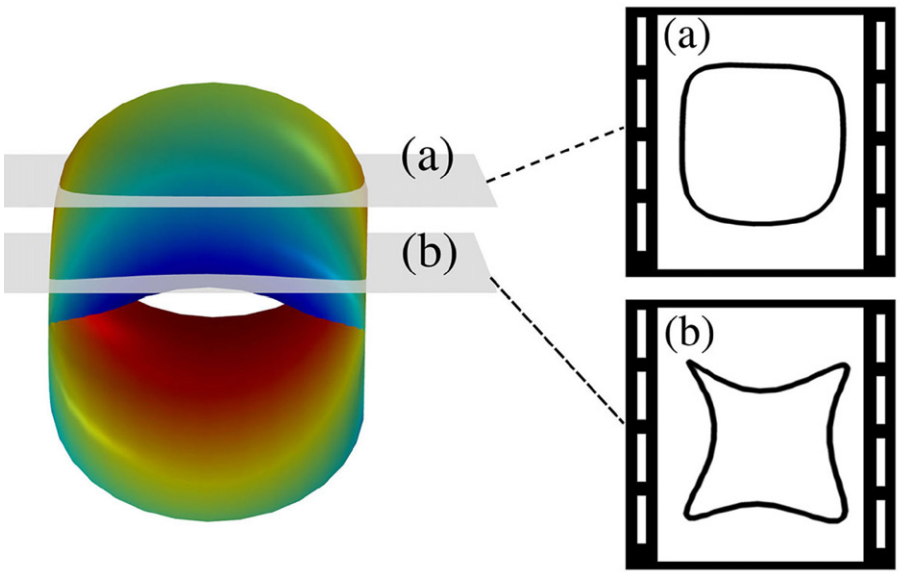


Figure 10. Select optimal frame with better convexity. (a) Convexity = 1.0. (b) Convexity = 0.5.

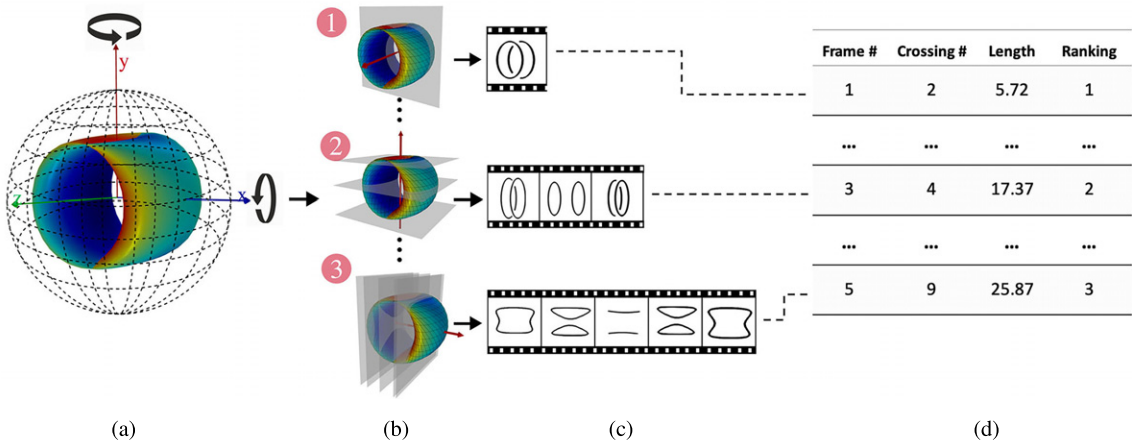


Figure 11. Auto-computed longitudinal axis. (a) Generating candidate longitudinal axes by rotating the surface around x-axis and y-axis incrementally, 180 in total. (b) → (c) generate movies from each candidate longitudinal axis. (d) Ranking and recommending the best movie from all movie outcomes.

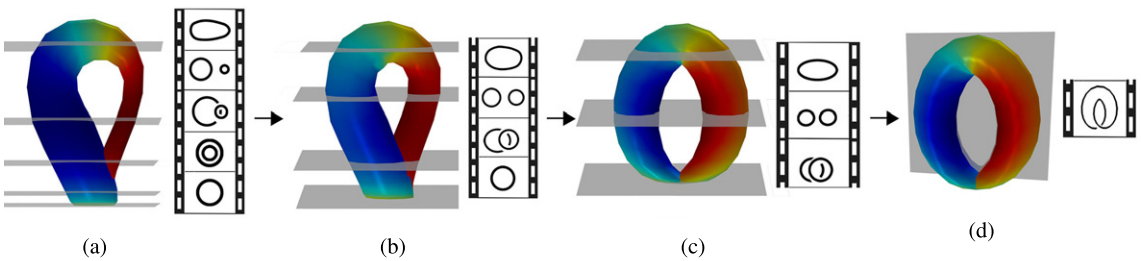


Figure 12. A series of movies generated for the Klein bottle, evolving from the classic shape to the pinched torus shape.

with which our system will render the shortest movie containing the most relaxed and topologically meaningful cross-sections.

The basic components of the algorithm is described in Figure 11. The algorithm rotates the surface so that its longitudinal axis coincides with z-axis as the initial step. The algorithm subsequently rotates the surface around the x-axis and y-axis, respectively, s degree each time for 180° in total. In this way, we get a total of $\left(\frac{180}{s}\right)^2$ longitudinal axes differently positioned to search for the best movie outcome (see e.g., the three representative longitudinal axes

in Fig. 11(b)). For each longitudinal axis, a movie outcome is generated (see Fig. 11(c)). Then the system compares all the candidate movies and ranks the movies for the viewer (see Fig. 11(d) and (f)). The metric used to compare the candidate movies follows the following rules:

1. **Movie Length.** Compare the length of the two movies, and the shorter movie is preferred. For example, in Fig. 11, different longitudinal axes were tested for generating the best movie to describe a 4D torus. In Fig. 11 ①, the candidate longitudinal axes results in a one-frame movie, which perfectly describes underlying $S^1 \times S^1$ structure behind the 3D figure of the 4D torus. The candidate longitudinal axis in Fig. 11 ② results in a movie of three frames, and in Fig. 11 ③ the movie contains 5 frames. Our algorithm ranks the movie outcome in Fig. 11 ① the best.
2. **Crossings.** In the events of two movies having the same length, our algorithm calculates the number of crossings from each frame, and prefer the movie outcome with the lesser total crossing number.
3. **Total Length of Close Loop(s).** When two movies are of same length and same crossing number, the total length of the close loop(s) in the two movies are compared. The algorithm recommends the movie with greater total length of close loop(s).

5. MORE EXAMPLES

We implemented the presented algorithms in *C++*. Our core rending capability, including the planar knot diagram and the 3D rendering for surface is based on *OpenGL*. The software currently runs on a *MacBook Pro* with 2.2GHz 6-Core *Intel Core i7* Processor and *Radeon Pro* 555X graphic processor.

We utilized the interface to render movies for a family of surfaces in 4-space. In Figure 12, we show a series of movies rendered for a Klein bottle being relaxed from the traditional bottle shape into a pinched torus shape [11]. The Klein bottle is a closed non-orientable surface that has no inside or outside, first described by Felix Klein [24]. Our tool starts with the standard shape of Klein bottle, and renders movies across difference phases while the Klein bottle was being relaxed with a energy based relaxation model [11]. The Klein bottle reaches its minimum energy state and appears to be a pinched torus in our dimensions (see e.g., the movie in Fig. 12 at the end).

In addition to the automatic method, our interface also provides a manual way to generate a movie of a surface. The manual method has its own advantages, one of which is that the direction of motion of the cutting plane can be defined by the user. In the examples depicted in Figure 13(a) and (b), the user places cutting planes around the 4D spun trefoil knotted sphere and the 1-twist spun knot to explore the underlying structure. The 6 user-defined cutting planes positioned in Fig. 13(a) generate identical cross-sectional diagrams, revealing the underlying spinning structure of the 4D spun the 4D surface is constructed by a three-dimensional knot spun about a plane in four dimensions. The resultant

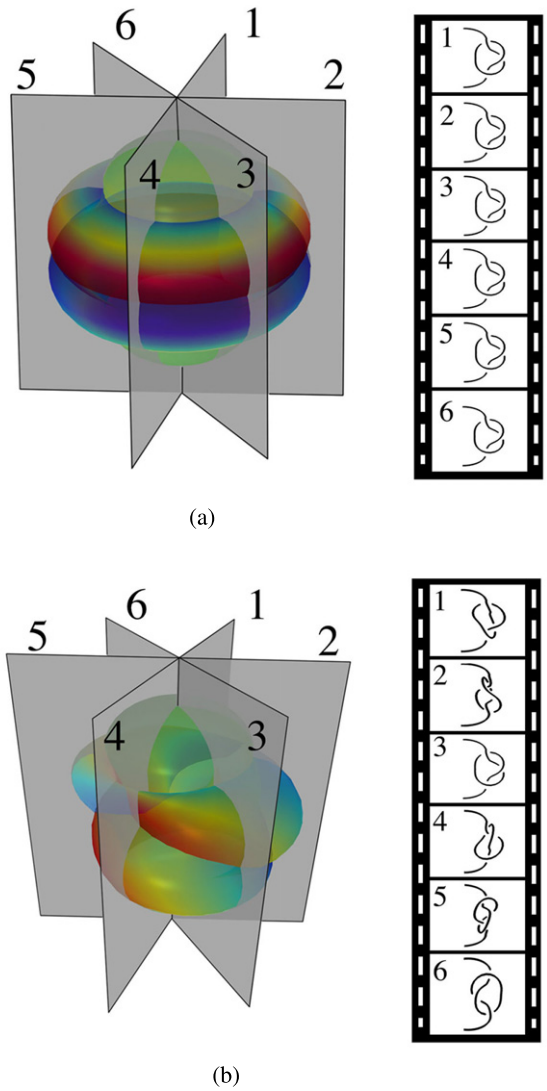


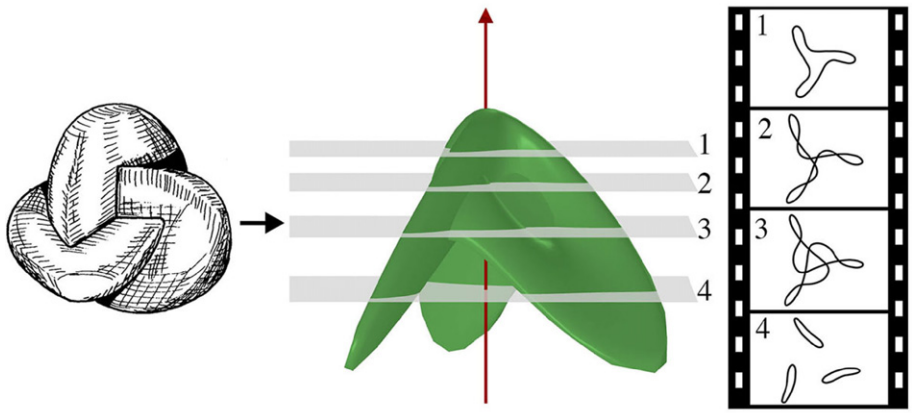
Figure 13. Apply user-defined slices to explore the 4D spun knotted sphere in (a), and the 1-twist spun knotted surface in (b).

cross-sectional diagrams in Fig. 13(b) reveal a different spinning process — the 1-twist spun knot was constructed by a twist spinning trefoil knot, which rotates 360° itself while spun about a plane in 4D.

In Figure 14 is a *Boy's surface*, a surface embedded in three-dimensional space, first studied by Werner in 1901 [25]. The algorithm and interface presented in this paper can also be exploited to generate the movie description to help us understand the underlying structure of the Boy's surface. Fig. 14(c) reveals the complex internal structure of the surface through our slicing interface.

6. PRELIMINARY USABILITY EVALUATION

We have performed a preliminary usability study in the UofL VCL (Visual Computing Lab) to evaluate our interface. The study invited a group of 12 non-expert participants to play two games we designed and adapted from our visualization tool. In both games, the participants were asked to interact



(a) (b) (c)

Figure 14. Generating movie for the Boy's surface. (a) A hand-drawn figure of the Boy's surface in Carter's book [5]. (b) The longitudinal axis and cutting positions in our interface. (c) A movie of 4 frames generated to describe the interior structure of the Boy's surface.

with the five mathematical surfaces, shown in Fig. 11, Fig. 12, Fig. 13(a), Fig. 13(b) and Fig. 14 respectively. Participants were asked to perform two tasks:

1. Drawing game — participants were given an interface capable of rendering a surface and allowing the user to rotate, scale, cut, and adjust the rendering of the surface (such as the opacity level). After interacting with the mathematical surface, participants were asked to describe the structure of the surface by drawing the contours of the surface they experimented with.
2. Matching game — participants were presented a collection of 3D static images of the surfaces, as well as flip-books of cross-sectional diagrams. Participants were asked to match the surface plots with the equivalent flip-books of diagrams.

Table I summarizes the completion rates of the participants from the two games. All the participants were able to complete the tasks of matching the flip-books and the image of the surfaces. However with the drawing game, results showed that the more complicated surfaces had a lower completion rate. Interviews with participants reveals that it was nearly impossible for them to understand the internal structure of the complicated surfaces. All participants agreed that our slicing interface can help them visualize how surfaces intersect in 4-space by "seeing" the key frames of the internal structures.

The study results suggest this new visualization tool can enable and enrich one's mathematical experience with mathematical surfaces, particular those embedded in high dimensional surfaces.

Table I. Completion rate of the two games in usability study.

Model	Drawing game completion rate	Matching game completion rate
4D torus	75%	100%
Klein bottle	83%	100%
4D spun trefoil	50%	100%
1-twisted spun	33%	100%
Boy's surface	17%	100%
Average	**53%**	**100%**

7. CONCLUSION

In this paper, we discuss a novel visualization method to slice 2-manifold embedded in 4 dimensions and explore their underlying topological structures. Through this novel visualization method, we can begin to appreciate the underlying mathematics and topological features behind the surface sheets of 4D surfaces of 3D figures. We further provide an automated interface of recommended slice directions to discover the most suitable slice sequence possible for the study. Several case studies have shown that our method works well in extracting useful geometric or topological properties on many classical surfaces, and this new automated approach could be applied to the study of knotted surfaces in more complex and general 4-dimensional spaces.

Starting from this basic slicing and movie-rendering framework, we plan to proceed to solve more 4D visualization problems such as the interactive manipulation of apparently knotted, but actually unknotted, spheres in 4D. Planned future work will also extend the range of objects for which we can support the interactive visualization of the smooth deformation between Boy and Roman surface, and the evolution of various 3D figures of the Klein bottle that have given the same surface in 4-space. In order to implement an interactive interface, improving computational efficiency is an essential part, both in the slicing process and in the recommendation of longitudinal axes, with the possibility of using parallel computing to improve the computing efficiency.

ACKNOWLEDGMENT

This work was supported in part by National Science Foundation grant IIS-1651581 and DUE-1726532.

REFERENCES

[1] S. Klimenkol, I. Nikitin, M. Göbel, and H. Tramberend, "Visualization in topology: assembling the projective plane," *Visualization in Scientific Computing '97* (Springer, Berlin/Heidelberg, Germany, 1997), pp. 95–104.

[2] J. F. Martins, "Categorical groups, knots and knotted surfaces," J. Knot Theory and its Ramificaions **16**, 1181–1217 (2007).

[3] A. J. Hanson, K. I. Ishkov, and J. H. Ma, "Meshview: Visualizing the fourth dimension," *Overview of the MeshView 4D Geometry Viewer* (Indiana University, USA, 1999).

[4] A. Hanson and H. Zhang, "Multimodal exploration of the fourth dimension," *VIS 05. IEEE Visualization, 2005* (IEEE, Piscataway, NJ, 2005), pp. 263–270.

[5] J. S. Carter, *How Surfaces Intersect in Space: An Introduction to Topology* (World Scientific, Singapore, 1995), Vol. 2.

[6] L. H. Kauffman, "Virtual knot theory," European J. Combinatorics **20**, 663–691 (1999).

[7] H. Zhang and A. Hanson, "Shadow-driven 4d haptic visualization," IEEE Trans. Vis. Comput. Graphics **13**, 1688–1695 (2007).

[8] D. Banks, "Interactive manipulation and display of surfaces in four dimensions," *Proc. 1992 Symposium on Interactive 3D Graphics. I3D '92* (Association for Computing Machinery, New York, NY, USA, 1992), pp. 197–207.

[9] A. Chu, C.-W. Fu, A. Hanson, and P.-A. Heng, "Gl4d: A gpu-based architecture for interactive 4d visualization," IEEE Trans. Vis. Comput. Graphics **15**, 1587–1594 (2009).

[10] A. M. Noll, "A computer technique for displaying n-dimensional hyper-objects," Commun. ACM **10**, 469–473 (1967).

[11] H. Zhang and H. Liu, "Relaxing topological surfaces in four dimensions," Vis. Comput. **36**, 2341–2353 (2020).

[12] G. Friedman, "Knot spinning," *Handbook of Knot Theory* (Elsevier Science, Netherlands, 2005), pp. 187–208.

[13] T. F. Banchoff, *Beyond the Third Dimension* (Scientific American Library, New York, 1990).

[14] O. Karpenko, W. Li, N. Mitra, and M. Agrawala, "Exploded view diagrams of mathematical surfaces," IEEE Trans. Vis. Comput. Graphics **16**, 1311–1318 (2010).

[15] H. Carr, J. Snoeyink, and U. Axen, "Computing contour trees in all dimensions," Comput. Geom. **24**, 75–94 (2003).

[16] H. Edelsbrunner and J. Harer, *Computational Topology: An Introduction* (American Mathematical Society, Providence, RI, 2010).

[17] J. S. Carter and M. Saito, *Knotted Surfaces and Their Diagrams* (American Mathematical Society, Providence, RI, 1998).

[18] I. Viola and E. Gröller, "On the role of topology in focus+context visualization," in *Topology-based Methods in Visualization*, edited by H. Hauser, H. Hagen, and H. Theisel (Springer, Berlin, Heidelberg, 2007), pp. 171–181.

[19] H. Zhang, J. Weng, and G. Ruan, "Visualizing 2-dimensional manifolds with curve handles in 4d," IEEE Trans. Vis. Comput. Graphics **20**, 2575–2584 (2014).

[20] R. Minetto, N. Volpato, J. Stolfi, R. Gregori, and M. da Silva, "An optimal algorithm for 3d triangle mesh slicing," Comput.-Aided Des. **92**, 07 (2017).

[21] A. Koschan, "Perception-based 3d triangle mesh segmentation using fast marching watersheds," *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, 2003* (IEEE, Pistcataway, NJ, 2003), Vol. 2, pp. II–II.

[22] H. Liu and H. Zhang, "A suggestive interface for untangling mathematical knots," IEEE Trans. Vis. Comput. Graphics **27**, 593–602 (2021).

[23] R. G. Scharein, "*Interactive topological drawing*." Ph.D. thesis (University of British Columbia, 1998).

[24] E. W. Weisstein, Klein bottle, (2003).

[25] W. Boy, *Über die Curvatura integra und die Topologie geschlossener Flächen. (Mit 24 Figuren im Text)*, Mathematische Annalen (Druck von Dieterich, 1903), Vol. 57, pp. 151–184.