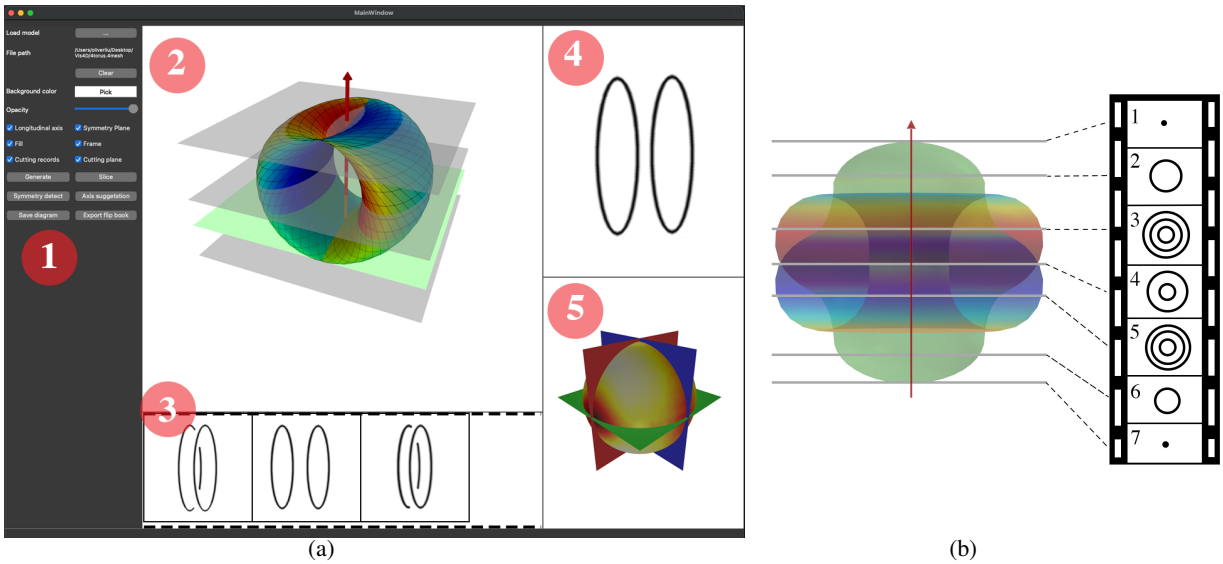
DOI: 10.1111/cgf.14545
Eurographics Conference on Visualization (EuroVis) 2022
R. Borgo, G. E. Marai, and T. Schreck
(Guest Editors)

Volume 41 (2022), Number 3

A Flip-book of Knot Diagrams for Visualizing Surfaces in 4-Space

Huan Liu and Hui Zhang

Computer Science and Engineering Department, University of Louisville, USA



(a) (b)

Figure 1: *System screen and major interface elements of our 4D visualization tool. (a) Our interface can suggest the optimal longitudinal axis for the viewer to explore 4D surfaces by extracting a flip-book of topologically meaningful slices.① — the toolbox to configure the slicing-based visualization tool, user interface elements including e.g., model file dialogue, slider to set opacity level, buttons to generate flip book (automatically or interactively). ② — central visualization panel for one to position longitudinal axis, cutting planes, and to view the slicing results. ③ — flip-book outcome that contains the optimal cross-sections to represent the surface's interior structure. ④ — cross-section viewer where one can interactively slice a 4D surface and view the intersection. ⑤ — entropy map viewer where one can observe the symmetry detection result when doing automatic longitudinal axis suggestion. (b) A flip-book of 7 cross-sectional diagrams extracted to represent a 4D spun knotted sphere.*

Abstract

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 itself. They twist, turn, and fold back on themselves, leaving important but hidden features behind the surface sheets. In this paper, we propose a smart slicing tool that can read the 4D surface in its entropy map and suggest the optimal way to generate cross-sectional images — or "slices" — of the surface to visualize its underlying 4D structure. Our visualization thinks of a 4D-embedded surface as a collection of 3D curves stacked in time, very much like a flip-book animation, where successive terms in the sequence differ at most by a critical change. This novel method can generate topologically meaningful visualization to depict complex and unfamiliar 4D surfaces, with the minimum number of cross-sectional diagrams. Our approach has been successfully used to create flip-books of diagrams to visualize a range of known 4D surfaces. In this preliminary study, our results show that the new visualization and slicing tool can help the viewers to understand and describe the complex spatial relationships and overall structures of 4D surfaces.

CCS Concepts

• ***Human-centered computing*** → *Scientific visualization; Visualization design and evaluation methods;*

1. Introduction

Many interesting mathematical objects require four dimensions to be appreciated fully. Among them are topological objects such as the 3-torus and the real projective plane which can only be embedded without self-intersection in 4D or higher, "knotted surfaces" (closed 2D surfaces embedded in 4D) [Mar05], and the quaternions, which are useful for representing 3D rotations [Sho85]. These 4D entities have a 4D "eye coordinate," or depth w, in addition to the coordinates (x, y, z) of their 3D graphics projections. Challenges arise when we visually communicate these 4D surfaces. 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, leaving important features behind the surface sheets. Most existing 4D visualization efforts rely on surface rending techniques, and exploit visual or haptic cues (see, e.g., [HIM99, HZ05]) to help the viewer to identify salient global features of the four-dimensional object, while often ignoring the important features or structures behind the surface sheets of their 3D graphics projections.

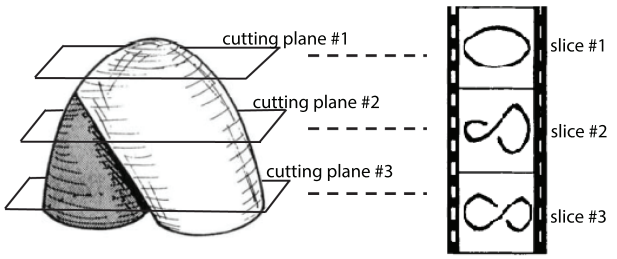


Figure 2: *Slicing a simple 4D surface in our dimensions — the "breaks" in the cross-sectional diagrams indicate which sheet is above (or under) another from the point of view of the projection into 3-space (modified from Carter's book [Car95]).*

Our goal in this paper is to exploit computer graphics and automatic algorithms to generate topologically meaningful illustrations — or "slices" — to depict these unfamiliar surfaces in space beyond 3D. Our basic idea is to think 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 [Car95]. If a set of 3-dimensional cutting planes are placed *appropriately* to slice a surface in 4-dimensions, the intersection will be a *reasonably* small number of cross-sectional diagrams that can describe the true 4D structure behind the surface sheets of their 3D graphics projections. For example, in Figure 2 a smooth (and simple) 4D surface folds back on itself, twist and turn in its 3D graphics projection. Three representative cutting planes are placed to obtain the topologically meaningful cross-sectional diagrams of this surface. The "breaks" in the cross-sectional diagrams indicate which sheet is above (or under) another from the point of view of the projection into 3-space — the 3D intersection between the surface sheets is only an artifact of projection. These cross-sectional diagrams in the progressing forms of movement, and the original 3D surface plotting, can combine to describe the underlying structure of the surface in 4 dimensions.

In this paper we take the first steps towards visualizing much more complex continuous topological surfaces in 4D, by designing a smart visual interface capable of helping us to slice the surface and extract flip-books of topologically meaningful diagrams to fully represent these 4D surfaces, which otherwise can only exist in mathematicians' mind.

2. Related Work

Traditional techniques for visualizing surfaces in 4D typically involve creating 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., [HZ05, HIM99, ZH07]). Other representative efforts include a variety of ways to render 4D objects (see e.g., Banks' interactive manipulation and display of surface in 4D [Ban92], Chu's use of 4D light sources to render 4D surfaces [CFHH09], Noll's method of rotating hyper-objects in four-dimensional space [Nol67], and Zhang's cloth-like modeling and rendering of 4D surfaces [ZL20]). Figure 3 shows some of the typical 4D visualization techniques. Figure 3(a) shows the 3D graphics projection of a 4D spun trefoil knot [Fri05], a knotted sphere embedded in 4D. An additional visual cue was added by assigning a surface color keyed to 4D depth relative to the projection center. Transparent surfaces in Figure 3(b) are created to help the viewer to perceive the internal structure of the spun trefoil — a trefoil knot spun about a plane in 4D. Other alternatives are to use cutaways and banded windows (e.g., see Figure 3(c)(d)) to remove portions of the surface to help the viewer see through the surface sheet of the 3D graphics projection [Ban90, HGH*10].

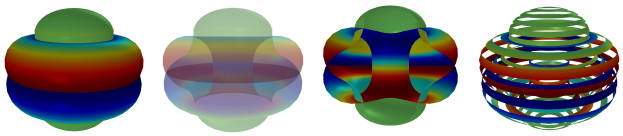


Figure 3: *Various approaches to visualizing a 4D spun trefoil knotted surface. (a) An additional visual cue was added by assigning a surface color keyed to 4D depth relative to the projection center. (b) Applying semi-transparency in addition to a surface color. (c)-(d) Exposing the structure behind the surface sheet with cutaways and banded windows.*

The renderings in Figure 3 are undoubtedly important for understanding the complex spatial relationships and overall structures of 4D surfaces, 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 serve to confuse the 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 behind the surface sheet, however it is nearly impossible for viewers to distinguish and interpret the structure in regions where multiple semi-transparent sheets intersect with each other. Similarly, cutaways and banded windows are both limited when applied to surfaces having more complex spatial relationships.

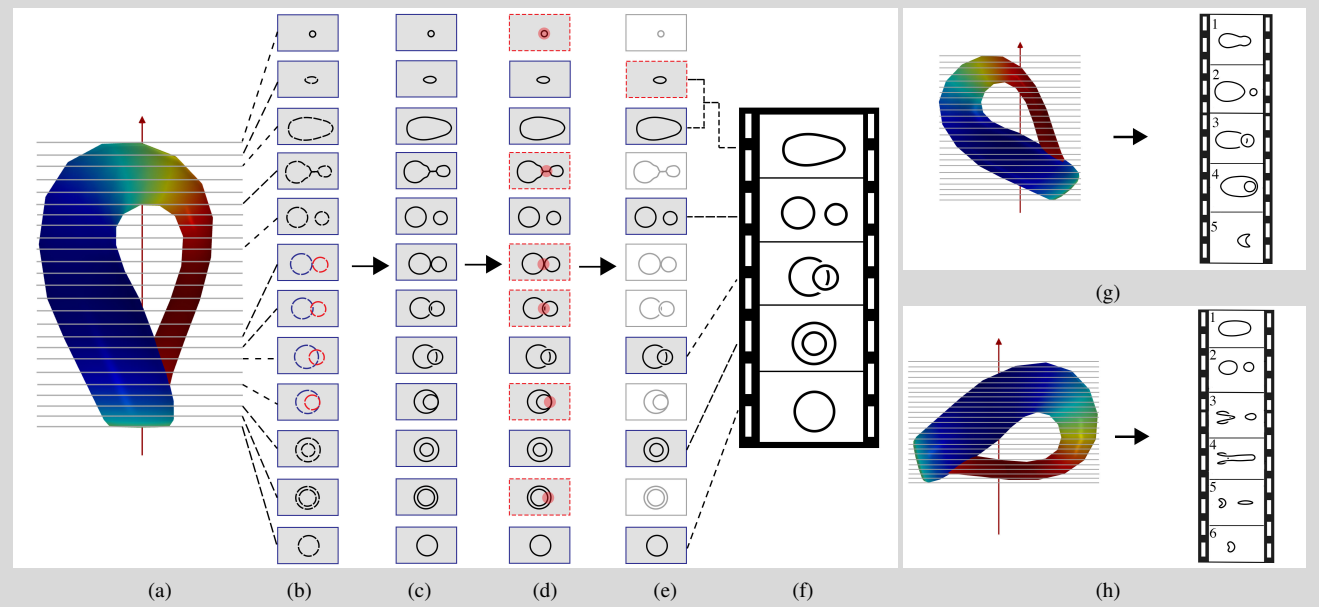


Figure 4: *Logical steps involved in the process of rendering a movie for the Klein Bottle. (a) Place longitudinal axis and cutting planes. (b) Calculate intersections on slices. (c) Reconstruct knot diagrams from intersections. (d) Identify critical changes and associated frames (highlighted with red dotted line box). (e) Selecting representative frames between critical changes to form the flip-book of diagrams. (f) The resultant visualization. (g)-(h) Placing the longitudinal axis differently results in different slices, intersections, and eventually different visualizations.*

3. Motivation

We are thus motivated to design a new tool that can read a 4D surface and generate topologically meaningful illustrations — or "slices" — to visualize these unfamiliar surfaces in space beyond our dimensions. The basic idea, as illustrated in Figure 2, is to slice the 4D surface appropriately and extract only the topologically meaningful intersection. Imagine we are interested in extracting horizontal slices of the 4D spun, and is empowered to place an infinite number of 3-dimensional cutting planes (see e.g., Figure 1(b)). The resultant intersection are (usually) a closed curve (or curves), except on the two cutting planes that intersect the spun knot at a point (i.e., the north/south pole). If we examine the resultant intersections over all the slices to extract the representative cross-sections, we seem to only need the seven cross-sectional diagrams in Figure 1(b) to fully describe the spun knot's underlying structure — successive terms in the sequence differ by a critical change. With the advent of interactive graphics technology and automatic algorithms we can begin to appreciate the challenge of depicting such surfaces embedded in high dimensions by slicing them and generating a flip-book of diagrams, which had only existed as hand-drawn diagrams in those beautifully written topology books (see e.g., [Car95] and [CS98]).

From the user's viewpoint, our visual interface is a smart "*slicing and imaging*" tool. The interface consists of a control panel and three major display areas. The user using the tool can load, transform, and view the 3D graphics projections of 4D surfaces with various desired settings and parameters. For example, the user can choose different 3D sub-spaces for projecting and viewing a 4D surface, rendered using the desired parameters. The user can toggle among various options to use transparency, cutaways, and banded windows when viewing the 3D graphics of 4D surfaces. In the central display area, the user can interactively define a longitudinal axis and place a cutting plane to extract slices of the 4D surface (see Figure 1). More importantly, our tool is capable of suggesting the optimal way to slice the surface and capture the representative diagrams from the slices. It reads the 4D surface in its entropy map, and can suggest the optimal longitudinal axis for placing slices. Out of a massive number of slices, the tool can analyze and reduce them to a minimal set of intersections that can still describe the 4D surface — very much like a flip-book where each page in the sequence shows just a representative slice and and these slices over pages differ at most by a critical change. Our tool can recognize and then "remember" each slice in the flip-book and present them in the interface, therefor one can take advantage of such a flip-book to trace and explore different portions of the surface being studied.

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:

- *Create a 3D graphics projection of a smoothly embedded object.* Examples are knotted curves embedded in 3D and knotted surfaces embedded in 4D. When projected to 3D, various parts of the smooth (non-intersecting) original shape appear to touch each other. What is seen is essentially the $(N-1)$-dimensional "shadow" of the original N-dimensional object.
- *Place the longitudinal axis and cutting planes.* To slice 4-dimensional space as a pile of 3-dimensional spaces stacked in time, we need to define a longitudinal axis (i.e., the *time*) and place cutting planes (i.e., the $cross-sections$) densely perpen-

dicular to the longitudinal axis, to ensure the inclusion of all topologically meaningful cross-sections (see e.g., Figure 4(a)).

- *Extract knot diagrams from slices.* The intersections over slices is the key to our understanding of the underlying 4D structure (see Figure 4(b)). Intersections in the original format of points and line segments will be converted into structured knot diagrams (see Figure 4(c)).
- *Select representative cross-sectional diagrams.* Cross-sectional diagrams extracted from all the slices are in progressing forms of movement. Most of the changes between successive frames are small and trivial changes from a topological perspective. Occasionally, the frames undergo significant changes such as the appearance or disappearance of a new closed loop, the change in the number of crossings in the diagrams, or a Reidemeister move (see Figure 4(d)). In this step, we will identify the representative diagrams leading to each critical change, and these representative diagrams forms the flip-book of diagrams for visualizing the 4D surface (see Figure 4(e) and (f)).

4. Implementation Methods

In this section we describe the families of models used to implement the interaction procedures, visual elements, slicing interface, and the automatic algorithm to compute the longitudinal axis and to extract representative cross-sectional images for generating the flip-book. Our fundamental techniques are based on a wide variety of prior art, including the use of exploded view in surface, flow, and volume visualization [WT10, KLMA10, VG07, ZWR14], algorithms for 3D triangle mesh slicing [MVS*17, Kos03], and other variants on computer graphics and visual interfaces for mathematical visualization including, e.g., the work of [LZ21] and [Sch98].

4.1. Slicing and Imaging the 4D Surface

Compute the intersections. Computing intersections of the surface and the parallel cutting planes is an essential part of our resolution of the problems. The cutting planes are placed densely along the longitudinal axis to slice the surface. Raw intersections are computed and derived in the format of points and line segments. To improve the computational efficiency, our implementation adopted several methods introduced in [MVS*17], 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.

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. For our principal test case of closed surfaces embedded in 4D, the line segments in the raw intersection should form one or more closed loops. For example, in Figure 5 the cutting plane slices through the Klein Bottle, and the raw intersections are shown in Figure 5(a). The collection of the line segments in the intersections are stringed into two closed loops since they exhibit different w eye coordinates (Figure 5(b)).

Identify critical changes. The densely positioned cutting planes slice the surface into a large number of cross-sectional frames, and

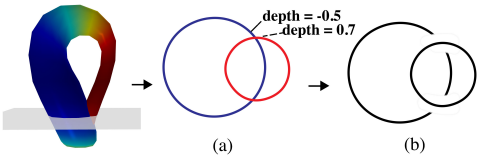


Figure 5: *Reconstruct the closed loops from line segments in the intersections. (a) A cutting plane slices a Klein Bottle, and the resultant intersections have two collection of line segments that appear to overlap each other, with different w coordinates. (b) We string line segment with each other. Based on their w coordinates, these line segments are stringed into two closed loops.*

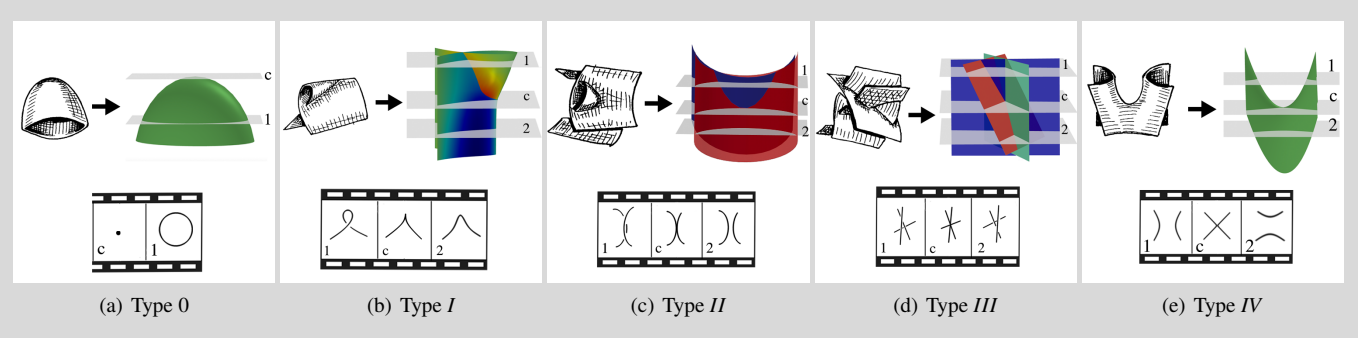
the changes between successive frames are often small and trivial changes from a topological perspective. Occasionally, the frames undergo significant changes such as the appearance or disappearance of a new closed loop, the change in the number of crossings in the diagrams, or a Reidemeister type of move. Carter summarizes five critical changes we might encounter over the cross-sectional frames sliced from the surfaces [Car95].

- **Type** 0: The birth/death of a simple closed curve. The change starts with frame c when the cutting plane is tangent to the surface, and transitions to the birth of the closed loop (Figure 6(a)).
- **Type** *I*: Introduced by a type *I* Reidemeister move, the preceding and succeeding frames of this change frame c undergo a type *I* Reidemeister move, which led to the addition/reduction of one crossing (Figure 6(b)).
- **Type** *II*: This critical change is introduced by a type *II* Reidemeister move. For example, the preceding and succeeding frames of the critical change frame c undergo a type *II* Reidemeister move, leading to addition/reduction of two crossings (Fig 6(c)).
- **Type** *III*: The preceding and succeeding frames of the critical change frame c undergone a type *III* Reidemeister move (Figure 6(d)).
- **Type** *IV*: As shown in Figure 6(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 two different pairs of non-intersecting curves.

Our next task is to identify the transitioning frames in these five critical changes illustrated in Figure 6. These frames can be identified with geometric computing — they are the transitioning (or, *unsafe*) frames between other frames whose diagrams are in *safe position* [Sch98]. A diagram is in a safe position when it fulfills the following conditions:

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

As summarized in Figure 6, when the five types of critical changes occur, the diagrams on the transitioning frames are turning into unsafe position, until the critical change is accomplished. Our system scans all the cross-sectional diagrams to identify diagrams in an unsafe position due to undergoing a critical change. The time periods when a series of consecutive unsafe frames have occurred



(a) Type 0 (b) Type *I* (c) Type *II* (d) Type *III* (e) Type *IV*

Figure 6: *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.*

are the critical time points. Our system locates all the critical time points (see Figure 4(d)), and removes unsafe frames occurring during these critical time points. The remaining candidate frames are divided into sections by these (removed) critical time points.

Selecting representative diagram from each section. We now identify the most representative frame from those in each section (separated by the critical changes). We will use one such representative diagram from each section to form the flip-book, as the difference across diagrams within each section is not a critical change, and therefore trivial from a topological perspective. The following metrics are used for the selection:

1. *Crossing Distance.* The sum of minimal crossing distance is used to determine which frame is more relaxed (preferred). For a given frame, with crossings, $C_i(i = 1,2,...,n)$. Let $D(i, j)$ be the distance between C_i and C_j. We calculate the minimum value of the distances between each crossing to all others $M(i) = min(D(i,1),...,D(1,i-1),D(i,i+1),...,D(i,n))$. The sum, $S = \sum_{i=1}^{n} M(i)$, is used to compare frames within the section. Our ranking prefers frames with greater crossing distance.
2. *Length.* When two frames have the same crossing distance, or they have less than two crossings, the system computes the total length of all polygons for each frame, and favors one with greater total length.
3. *Convexity.* Lastly our ranking method favors the frame with the more "convex" diagram (see e.g., [ZR04] for polygon convexity calculation). In this work, we use 0.7 weight for *length* and 0.3 for *convexity* when they are used.

As shown in Figure 4(e) and (f), our visualization interface now renders the resultant sequence of representative diagram into flip-book like visualization, that contains the minimum number of frames, that are topologically meaningful to describe the interior structure of the 4D surface perceived in our dimensions.

4.2. The Optimal Longitudinal Axis for Flip-book Creation

The methods we discuss so far in principle is sufficient to allow us to create a flip-book of diagrams to visualize a 4D surface. However, in practice, choosing the right longitudinal axis is not a trivial task before we can start to generate the cross-sectional diagrams and create the flip-book. A differently placed longitudinal axis will possibly result in a very different, even an unnecessarily complicated flip-book of diagrams (see e.g., Figure 4(g)(h)). The flip-books can differ in the number of frames, or the complexity of the frames extracted. In this section, we discuss an algorithm that can auto-compute the optimal longitudinal axis for us to create the "simplest" flip-book that will use a minimal number of most representative diagrams.

The overall idea of this algorithm is described in Figure 7. We start by detecting the presence of symmetry in the 3D graphics of 4D surfaces, using an entropy based approach proposed in Li's work [LJYL16]. By analyzing the extracted symmetry information, our algorithm suggests all candidates of the longitudinal axis for flip-book creation, and the final recommendation will be made by ranking the resultant flip-books created from all the candidate axes.

4.2.1. Symmetry Detection using Viewpoint Entropy Map

Unlike some common symmetry detection approaches based on geometry features (see e.g., [MGP06, BBW*09]), our approach exploit information theory based measurement, the *viewpoint entropy*, to detect symmetry. Viewpoint entropy was first introduced by Váquez et al. [VFSH01]. It is based on the Shannon entropy, and represents the visual entropy of a visible object when observed from a specific position, i.e., the viewpoint. In our approach, the viewpoint entropy is defined in Equation 1, essentially an orthogonal projection version of what was developed by Takahashi et al. in [TFTN05].

$$E = \frac{1}{log_2(m+1)} \sum_{j=0}^{m} \frac{A_j}{S} log_2 \frac{A_j}{S}, \tag{1}$$

where m is the number of visible triangles, A_j is the projected area of triangle $j(j = 1,2,...,m)$, S is the total area of the projection plane, and A_0 is the area of the blank background portion of the projection plane. The equation can be further simplified as Equation 2, as when deriving viewpoint entropy, we project the surface into a normalized space, contained in a normalized bounding sphere. We also translate the normalized surface by centering its bounding sphere at the origin, to facilitate efficient detection of the surface's symmetry planes at later steps.

$$E = \frac{1}{log_2(m+1)} \sum_{j=0}^{m} A_j log_2 A_j. \tag{2}$$

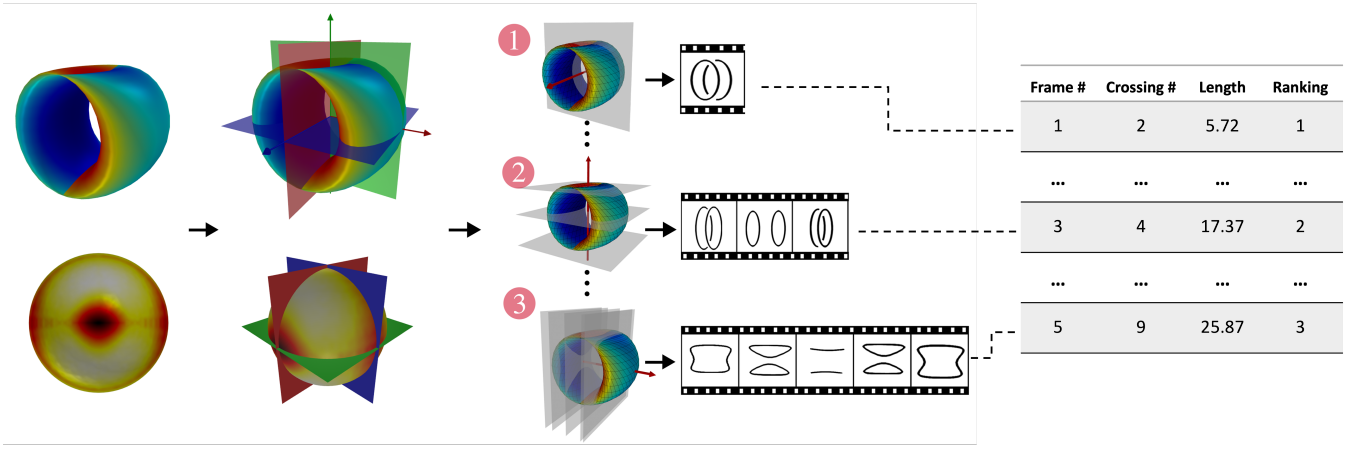


Figure 7: *Auto-computing the optimal longitudinal axis. (a) Generating entropy maps of 4D surfaces. (b) Detecting symmetry and finding longitudinal axis candidates. (c)→(d) Creating flip-books from each longitudinal axis candidate. (e) Ranking and recommending the optimal flip-book from all outcomes.*

Vertices of the bounding sphere are our viewpoints. From each viewpoint **P**, our method calculates the viewpoint entropy value of the surface. We next outline the key steps towards deriving the *viewpoint entropy map* that visualizes the distribution of entropy values calculated from all viewpoints. Let **U** be **[0.0, 1.0, 0.0]**, and **R** be **(-P)** × **U**. We define the 3D projection matrix **T** in Equation 3,

$$\mathbf{T} = \begin{bmatrix} R_x & R_y & R_z & 0 \\ U_x & U_y & U_z & 0 \\ -P_x & -P_y & -P_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -P_x \\ 0 & 1 & 0 & -P_y \\ 0 & 0 & 1 & -p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

With transformation matrix **T**, Equation 4 will project a 3D vertex V to 2D (V') on a projective plane,

$$\begin{bmatrix} V'_x & V'_y & V'_z & w \end{bmatrix}^{\mathsf{T}} = T \cdot \begin{bmatrix} V_x & V_y & V_z & 1 \end{bmatrix}^{\mathsf{T}}. \tag{4}$$

After the surface is projected from 3D to 2D (using Equation 4), viewpoint entropy value is calculated using all the visible triangles in the projection. Each viewpoint (i.e., each vertex on the bounding sphere) will be paired up with an entropy value. In Figure 8-*II* the viewpoint icosahedrons on bounding spheres are rendered with surface color keyed to their associated viewpoint entropy values — these are the viewpoint entropy map we will leverage to detect symmetry in our next step.

We next describe our process to detect the potential symmetry planes from an entropy map we have just derived. The main idea is to iteratively select a matching pair of viewpoints to generate a symmetry plane, and verify all the rest matching pairs to see whether they are symmetric as well with regard to the symmetry plane or at least in the symmetry plane. The logic steps are listed in Algorithm 1: it first locates a pair of entropy of equal viewpoints (thus the difference is lower than the threshold δ), and creates a symmetry plane exactly halfway between the two viewpoints and perpendicular to the line connecting them. The next step is to verify if there exists at least $\frac{N-2^{l-2}}{2}$ pairs of viewpoints that are symmetric w.r.t. this plane N is the total number of viewpoints, l is the iterating level building the entropy map). If the condition is satisfied, one symmetry plane is found. Note that we do not require a very strict symmetry determination condition, so the value of entropy difference variation(δ) is 0.075 by default (about five times greater than the value used in the original algorithm reported in [LJYL16]).

4.2.2. From Symmetry Planes to Candidate Longitudinal Axes

The symmetry detection results can be categorized in three scenarios. We next identify candidate longitudinal axes in each of these three scenarios:

1. **Two or more symmetry planes.** Examples are 3 symmetric planes found for a 4D torus (Figure 8 *III*-(c)), and similarly the 4 symmetric planes for a 4D spun trefoil knotted sphere (Figure 8 *III*-(k)). In this case we compute the intersection lines between the symmetry planes as the candidate longitudinal axis (axes).
2. **Only one symmetry planes.** For example, in Figure 8 *III*(g), only one symmetry plane has been found for the Klein bottle. We then compute all the intersection points between this symmetry plane and the surface, and select the line connecting the two farthest points of all focal points as the longitudinal axis.
3. **No symmetry planes.** The surface is completely asymmetric. For example, no symmetric planes are found for the 1-twisted spun knotted sphere in Figure 8 *III*(o). In this case we compute the axis of longest extent as the longitudinal axis.

4.2.3. Recommending the Optimal Longitudinal Axis

When there are multiple candidate longitudinal axes identified, we will compute all the optimal flip-books, corresponding to each of the candidates. We use the following ranking metrics to recommend the optimal longitudinal axis with the highest score:

1. **Flip-book Length.** Fewer diagrams in the flip-book is preferred. In Figure 7, different candidate axes of the 4D torus were evaluated by generating the flip-books. In Figure 7①, a one-diagram

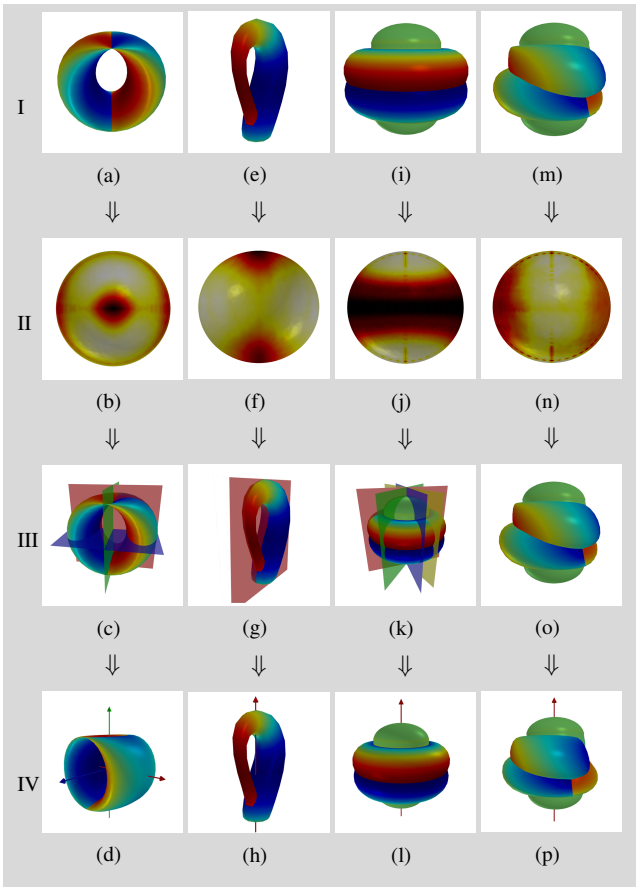


Figure 8: *Surface (I) → Entropy maps (II) → Symmetry planes (III) → Longitudinal axes (IV). Viewpoint entropy maps are derived for surfaces, including a 4D torus (a), a Klein bottle (e), a 4D spun (i), and a 1-twisted spun (m). Symmetry planes are built based on their entropy maps, and the longitudinal axes are finally identified for each of the surfaces.*

flip-book is generated, which perfectly describes the underlying $S^1 \times S^1$ structure of the 4D torus. The candidate axis in Figure 7② results in a flip-book of three diagrams, and in Figure 7③ the flip-book contains 5 diagrams. Our algorithm ranks the flip-book in Figure 7① as the best among all.

2. **Number of Crossings.** In the events of two flip-books having equal length, our algorithm counts the number of crossings from all diagrams in each flip-book, and favours the flip-book with the fewer crossings.
3. **Total Length of Closed Loop(s).** When two flip-books are of equal length and crossing number, the total length of the closed loop(s) in each flip-book will be computed and compared. We favour flip-books with greater total length of closed loop(s).

Rotational Axis. We typically create and render flip-books of diagram by making slices perpendicular to the longitudinal axis. In some other scenarios, we have found that we can obtain a better flip-book of diagrams by slicing the surface around the longitudinal axis, much like how we make the pizza slices.

In Figure 3(e), we sliced the 4D spun with perpendicular cutting planes along the longitudinal axis, and obtained a flip-book of 7 unique diagrams extracted from representative slices. This flip-book of 7 diagrams may not be the best way to describe the underlying structure for the 4D spun trefoil knotted sphere — a trefoil knot spun about a plane in 4D, whose 3D graphics projection is indeed a completely symmetric structure. In Figure 9, we rotate the cutting plane about the longitudinal axis. While the rotation and the slicing continue, we can conclude that all the diagrams are identical and just one single diagram should be sufficient to represent the internal structure of the 3D graphics of a 4D spun knotted sphere.

Inspired by this example, we choose to complement our visualization tool to perform both parallel and rotational slices around each candidate longitudinal axis, to search and recommend the right longitudinal axis and the right flip-book of diagrams to fully describe the 4D surface.

Algorithm 1: Symmetry detection by pairing entropy values

Input : N: number pf viewpoints;
$Pos[N]$: Positions of N viewpoints;
$Ent[N]$: Entropy values of N viewpoints
l: Icosahedron subdivision level;
δ: Entropy difference variation;
ε: Minimal difference in two float values
Output: Planes: All detected symmetry planes

```
for u ← 0 to N − 2 do
    P_u ← Pos[u]
    for v ← u + 1 to N − 1 do
        if |Ent[u] − Ent[v]| > δ ∗ min(Ent[u], Ent[v]) then
            continue
        matches ← 2, P_v ← Pos[v]
        T_1 ← normalize(P_u − P_v)
        for i ← 0 to N − 2 do
            if i == u or i == v then
                continue
            P_i ← Pos[i] for j ← i + 1 to N − 1 do
                if j == u or j == v then
                    continue
                if |Ent[i] − Ent[j]| > δ ∗ min(Ent[i], Ent[j])
                 then
                    continue
                P_j ← Pos[j], P_m ← (P_i + P_j)/2
                T_2 ← normalize(P_i − P_j)
                CT ← T_1 × T_2, DT ← T_1 · T_2
                if ||CT|| > ε and |DT| ≠ 0 then
                    continue
                if |T_1 · P_m| > ε then
                    continue
                matches ← matches + 2
                break
        if matches ≥ (N − 2^(l−2)) then
            plane ← T_1[0]x + T_1[1]y + T_1[2]z = 0
            if plane not in Planes then
                Append plane to Planes
```

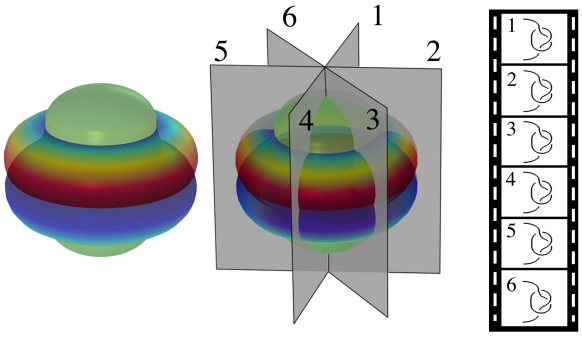


Figure 9: *Apply rotational slices to explore the 4D spun knotted surface. The resultant diagrams across slices are identical.*

5. More Examples

We have been able to utilize our slicing tool to render flip-books of diagrams from an array of known surfaces in 4-space. In Figure 10, we show a series of movies rendered for a Klein bottle being relaxed from the traditional bottle shape into a pinched torus shape [ZL20]. The Klein bottle is a closed non-orientable surface that has no inside or outside, first described by Felix Klein [Wei03]. Our tool starts with the standard shape of Klein bottle, and renders movies across difference phases while the Klein bottle was being relaxed under an energy model [ZL20]. The Klein bottle reaches its minimum energy state and appears to be a pinched torus in our dimensions (see the flip-book rendered in Figure 10(d)).

Our interface is highly interactive and capable of rendering the flip-book of diagrams in real time for the viewer, with a user-defined longitudinal axis. This allows the viewer to explore the 4D surface on his/her own, and can examine and compare the different cross-sections by placing different longitudinal axes. The interface also allows one to extract the cross-section by placing an arbitrary cutting plane in the scene. For example, one can use this tool to slice into the 4D spun and the 1-twisted spun (see Figure 9 and Figure 11), to compare the different shape and structure of the trefoil knot that is spun into the 4D surfaces.

In Figure 12 we visualize a *Boy's surface*, a surface embedded in three-dimensional space, first studied by Boy [Boy01]. The algorithm and interface presented in this paper can also be exploited to extract the representative diagrams to help us understand the internal structure of the Boy's surface. Figure 12(c) shows the complex internal structure of this surface using our slicing interface.

6. Evaluation

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

Our smart slicing tool has been used to generate visualizations to describe an array of 4D surfaces. In Table 1 we summarize the total processing time for the optimal flip-books to be auto-computed and

Table 1: *Total processing time needed to identify the optimal longitudinal axis, and render the optimal flip-book visualization*

Surface	Triangle #	Slicing #	Key frame#	Time
4D torus	800	60	1	191*s*
Klein bottle	800	60	5	171*s*
4D spun trefoil	5600	210	7	452*s*
1-twisted spun	5600	210	7	421*s*
Boy's surface	3200	120	4	225*s*

Table 2: *Further breakdowns of processing time*

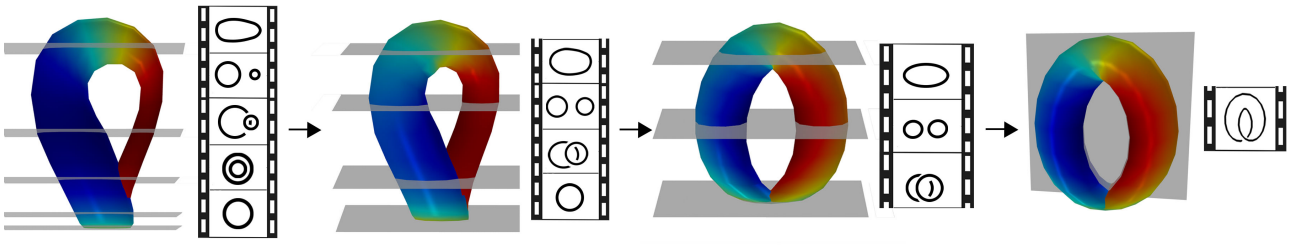
Surface	per slice	Symmetry detection	Rendering Flip-book	Number of flip-books
4D torus	0.3*s*	82*s*	27*s*	4
Klein bottle	0.3*s*	104*s*	33*s*	2
4D spun trefoil	0.5*s*	158*s*	98*s*	3
1-twisted spun	0.5*s*	207*s*	107*s*	2
Boy's surface	0.4s	97*s*	64*s*	2

derived for each 4D surface. The total time listed include several compute-intensive steps leading to the visualization: the symmetry detection and the identification of all candidate axes, and the evaluation of each candidate axis by generating flip-book of diagrams from them, etc. The further breakdowns of total processing time is listed in Table 2. Three parameters can impact the performance significantly: the triangle amount of the model, the slicing interval distance and the *icosahedron subdivision level*, which was mentioned in Algorithm 1. Its worth noting that our visualization tool is highly interactive with user-defined longitudinal axis. The relatively long processing time is only needed for the auto-computed process, that includes the search for the optimal axis.

Table 3: *Completion rates from the two sessions*

Surface	View-only	Slicing-enabled
4D torus	75%	100%
Klein bottle	83%	100%
4D spun trefoil	50%	100%
1-twisted spun	33%	100%
Boy's surface	17%	100%

We also performed a preliminary usability study in the UofL *VCL* (Visual Computing Lab) to evaluate the smart slicing interface. We invited a group of 12 non-expert participants to a game we designed by utilizing an interface adapted from the slicing tool presented in this paper. Before the game, an introduction to surfaces embedded in four dimensions were given to the participants. In the introduction, we used a flip-book of diagrams to describe a very simple 4D surface, to help their understanding of 4D surfaces in our dimensions. The participants were told that they will be using visualization software to view and interact with five mathematical surfaces in two sessions of this study, 30 minutes each session. After each session, the participants were asked to select one flip-book



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

Figure 10: *A series of flip-books rendered when the Klein bottle [Kau98] was evolving from the classic shape to the pinched torus shape. Klein bottle is a one-sided surface which, if traveled upon, could be followed back to the point of origin while flipping the traveler upside down*

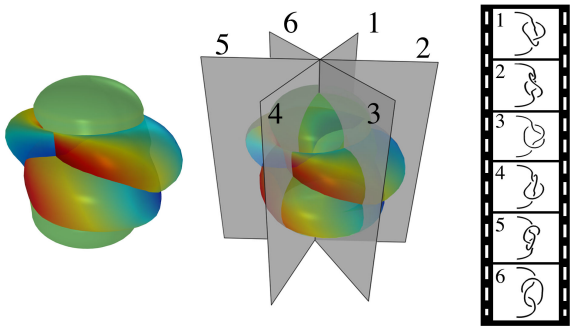
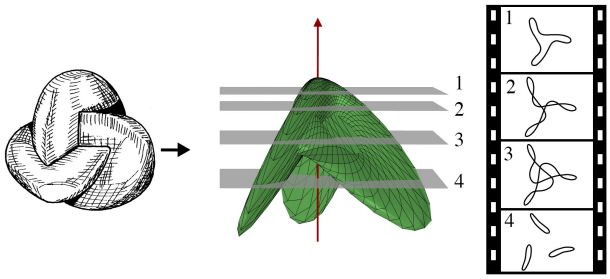


Figure 11: *Apply rotational slices to explore the 4D 1-twist spun knotted surface.*



(a) (b) (c)

Figure 12: *Generating flip-book of diagrams for the Boy's surface. (a) A hand-drawn figure of the Boy's surface in Carter's book [Car95]. (b) The longitudinal axis and cutting planes used in our interface. (c) A flip-book of 4 diagrams is generated to describe the interior structure of the Boy's surface projected to 3D.*

from a set of three to match each of the five mathematical surfaces they explored. The 5 surfaces used in the study included those from Figure 7, Figure 10, Figure 9, Figure 11 and Figure 12.

1. *With a View-only Interface* — In the first session, the participants were given the adapted (view-only) interface which is capable of rendering these five surfaces and allows the users to rotate, scale, and explore the surface with their desired view options (e.g., the opacity level to be used).

Table 4: *Recorded interaction times from the two sessions*

Surface	View-only	Slicing-enabled
4D torus	2.75 *min*	3.2 *min*
Klein bottle	3.67 *min*	4.5 *min*
4D spun trefoil	4.33 *min*	7.5 *min*
1-twisted spun	2.65 *min*	4.3 *min*
Boy's surface	2.75 *min*	6.6 *min*

2. *With a Slicing-enabled Interface* — In this second session, the participants were presented the smart slicing interface. They were able to place desired longitudinal axes in the scene, and view the resultant flip-book of diagrams recommend by the tool. They were also able to use the slicing tool to obtain any arbitrary cross-sections they desired in the exploration.

Table 3 shows the completion rates from participants in the two sessions. The participants by just viewing the 3D graphics projections were not able to describe the structure of the surfaces, especially for those complex ones, such as 1-twisted spun and the Boy's surface. They eventually were able to complete the tasks after the second session when they were exposed to the right tool — one that they can use to slice the surface and examine the cross-sections. Table 4 lists the average times that participants spent interacting with the five surfaces in the two sessions of this study. We have two observations: first, the time spent with each of the surfaces in the first (view-only) session is significantly lower than in the second (slicing-enabled) session; second, in the first session, the time spent on more complex surfaces are not necessary longer than on those simpler ones, while in the second session, participants spent observably longer time to interact with more complex surfaces such as the Boy's surface. The completion rates and the recorded time here suggest that the slicing tool can engage the users by allowing them to explore the surfaces in a meaningful way. At the end of the study, the participants all agreed that our slicing interface helped them to visualize the surfaces embedded in 4D by "seeing" their shadow images in our dimensions in this way. All these results suggest that the new visualization tool we are creating can enable one's mathematical experience with mathematical surfaces, particularly those embedded in high dimensional surfaces.

7. Conclusions and Future Work

In this paper, we discuss a novel visualization method to slice surfaces embedded in 4 dimensions and use a flip-book of the resultant cross-sections to visualize the surfaces. Through this new visualization, we can begin to appreciate the underlying structures of these 4D surfaces. We further provide an automated method to recommend the right longitudinal axis in order to create the most meaningful flip-book of diagrams. Case studies have shown that our method scales well to many classical and known surfaces, and the new automated approach can be applied to the study of knotted surfaces in more complex and general 4-dimensional spaces when placing an optimal longitudinal axis is challenging. Starting from this basic slicing and flip-book rendering framework, we plan to proceed to visualizing the evolution (deformation) of 4D surfaces with a series of flip-books. Improvement of rendering and visualization of 4D surfaces, e.g., the use of 4D lighting model, is also among our future research directions. In order to allow an interactive interface for even more complex surfaces, we also consider the possibility of using parallel computing to improve computational efficiency, both in the slicing process and in the recommendation of longitudinal axes.

8. Acknowledgments

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

References

[Ban90] BANCHOFF T. F.: *Beyond the third dimension*. Scientific American Library New York, 1990. 2

[Ban92] BANKS D.: Interactive manipulation and display of surfaces in four dimensions. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics* (New York, NY, USA, 1992), I3D '92, Association for Computing Machinery, p. 197–207. 2

[BBW*09] BOKELOH M., BERNER A., WAND M., SEIDEL H.-P., SCHILLING A.: Symmetry detection using feature lines. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 697–706. 5

[Boy01] BOY W.: *Über die Curvatura integra ud Topologie geschlossener Flächen*. Dieterich, 1901. 8

[Car95] CARTER J. S.: *How surfaces intersect in space: an introduction to topology*, vol. 2. World Scientific, 1995. 2, 3, 4, 9

[CFHH09] CHU A., FU C.-W., HANSON A., HENG P.-A.: Gl4d: A gpu-based architecture for interactive 4d visualization. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1587–1594. doi:10.1109/TVCG.2009.147. 2

[CS98] CARTER J. S., SAITO M.: *Knotted surfaces and their diagrams*. American Mathematical Soc., 1998. 3

[Fri05] FRIEDMAN G.: Knot spinning, handbook of knot theory, 187–208, 2005. 2

[HGH*10] HUMMEL M., GARTH C., HAMANN B., HAGEN H., JOY K. I.: Iris: Illustrative rendering for integral surfaces. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1319–1328. 2

[HIM99] HANSON A. J., ISHKOV K. I., MA J. H.: Meshview: Visualizing the fourth dimension. *Overview of the MeshView 4D geometry viewer* (1999). 2

[HZ05] HANSON A., ZHANG H.: Multimodal exploration of the fourth dimension. In *VIS 05. IEEE Visualization, 2005.* (2005), pp. 263–270. doi:10.1109/VISUAL.2005.1532804. 2

[Kau98] KAUFFMAN L. H.: Virtual knot theory. *arXiv preprint math/9811028* (1998). 9

[KLMA10] KARPENKO O., LI W., MITRA N., AGRAWALA M.: Exploded view diagrams of mathematical surfaces. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1311–1318. 4

[Kos03] KOSCHAN A.: Perception-based 3d triangle mesh segmentation using fast marching watersheds. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* (2003), vol. 2, IEEE, pp. II–II. 4

[LJYL16] LI B., JOHAN H., YE Y., LU Y.: Efficient 3d reflection symmetry detection: A view-based approach. *Graphical Models 83* (2016), 2–14. 5, 6

[LZ21] LIU H., ZHANG H.: A suggestive interface for untangling mathematical knots. *IEEE Transactions on Visualization and Computer Graphics 27*, 2 (2021), 593–602. doi:10.1109/TVCG.2020.3028893. 4

[Mar05] MARTINS J. F.: Categorical groups, knots and knotted surfaces. *arXiv preprint math/0502562* (2005). 2

[MGP06] MITRA N. J., GUIBAS L. J., PAULY M.: Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics (TOG) 25*, 3 (2006), 560–568. 5

[MVS*17] MINETTO R., VOLPATO N., STOLFI J., GREGORI R., DA SILVA M.: An optimal algorithm for 3d triangle mesh slicing. *Computer-Aided Design 92* (07 2017). doi:10.1016/j.cad.2017.07.001. 4

[Nol67] NOLL A. M.: A computer technique for displaying n-dimensional hyperobjects. *Commun. ACM 10*, 8 (Aug. 1967), 469–473. 2

[Sch98] SCHAREIN R. G.: *Interactive topological drawing*. PhD thesis, University of British Columbia, 1998. 4

[Sho85] SHOEMAKE K.: Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques* (1985), pp. 245–254. 2

[TFTN05] TAKAHASHI S., FUJISHIRO I., TAKESHIMA Y., NISHITA T.: A feature-driven approach to locating optimal viewpoints for volume visualization. In *VIS 05. IEEE Visualization, 2005.* (2005), pp. 495–502. doi:10.1109/VISUAL.2005.1532834. 5

[VFSH01] VÁZQUEZ P.-P., FEIXAS M., SBERT M., HEIDRICH W.: Viewpoint selection using viewpoint entropy. In *VMV* (2001), vol. 1, Citeseer, pp. 273–280. 5

[VG07] VIOLA I., GRÖLLER E.: On the role of topology in focus+context visualization. In *Topology-based Methods in Visualization* (Berlin, Heidelberg, 2007), Hauser H., Hagen H., Theisel H., (Eds.), Springer Berlin Heidelberg, pp. 171–181. 4

[Wei03] WEISSTEIN E. W.: Klein bottle. *https://mathworld. wolfram. com/* (2003). 8

[WT10] WEINKAUF T., THEISEL H.: Streak lines as tangent curves of a derived vector field. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1225–1234. doi:10.1109/TVCG.2010.198. 4

[ZH07] ZHANG H., HANSON A.: Shadow-driven 4d haptic visualization. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1688–1695. doi:10.1109/TVCG.2007.70593. 2

[ZL20] ZHANG H., LIU H.: Relaxing topological surfaces in four dimensions. *The Visual Computer 36*, 10 (2020), 2341–2353. 2, 8

[ZR04] ZUNIC J., ROSIN P. L.: A new convexity measure for polygons. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 7 (2004), 923–934. 5

[ZWR14] ZHANG H., WENG J., RUAN G.: Visualizing 2-dimensional manifolds with curve handles in 4d. *IEEE transactions on visualization and computer graphics 20*, 12 (2014), 2575–2584. 4