Relaxing topological surfaces in four dimensions

Hui Zhang & Huan Liu

The Visual Computer
International Journal of Computer Graphics

ISSN 0178-2789

Vis Comput
DOI 10.1007/s00371-020-01895-5

Springer

The Visual Computer
https://doi.org/10.1007/s00371-020-01895-5

ORIGINAL ARTICLE

Relaxing topological surfaces in four dimensions

Hui Zhang[1] · Huan Liu[1]

Abstract
In this paper, we show the use of visualization and topological relaxation methods to analyze and understand the underlying structure of mathematical surfaces embedded in 4D. When projected from 4D to 3D space, mathematical surfaces often twist, turn, and fold back on themselves, leaving their underlying structures behind their 3D figures. Our approach combines computer graphics, relaxation algorithm, and simulation to facilitate the modeling and depiction of 4D surfaces, and their deformation toward the simplified representations. For our principal test case of surfaces in 4D, this for the first time permits us to visualize a set of well-known topological phenomena beyond 3D that otherwise could only exist in the mathematician's mind. Understanding a fairly long mathematical deformation sequence can be aided by visual analysis and comparison over the identified "key moments" where only critical changes occur in the sequence. Our interface is designed to summarize the deformation sequence with a significantly reduced number of visual frames. All these combine to allow a much cleaner exploratory interface for us to analyze and study mathematical surfaces and their deformation in topological space.

Keywords Visual mathematics · Relaxation · Optimization · Knot theory · Four dimensions

1 Introduction

We often use figures to help communication about geometric objects [21], and our perception and understanding of geometric properties can be strongly facilitated by drawing and viewing these figures. However, it can be a far more challenging task to communicate about topology [15,31], which studies geometrical objects under the equivalence relation of homeomorphism, i.e., the geometric properties and spatial relations unaffected by the continuous change of shape or size of figures. Our paper is mainly concerned with the illustration of the topology of mathematical surfaces (2-dimensional objects) embedded in 4-space. The main properties of these surfaces to be studied and visualized in our work are related to their topology. In topology, a very small circle is the "same" as a huge one, and a small sheet of surface is the "same" as a big one, because we can stretch the small ones to make them exactly like the big ones. More generally, two sheets are going to be considered the "same" if it is possible to deform one into the other without cutting. Therefore, we think of *surface* as rubber sheet geometry—we can stretch or shrink them as much as we want, without tearing or losing the "thinness."

Generating illustrative drawings to communicate such topological properties are not trivial, and unique challenges for this class of math visualization problems include:

- **Length and area are of no concern in topology problems** When relating surfaces under the equivalence relation of homeomorphism, their geometric shapes are relatively unimportant. We are going to think that surfaces are made of extremely flexible rubber materials, able to shrink and expand at will. The length of the side or the area of surface becomes no concern in their computerized models and in their deformations.
- **Long and compute-intensive deformation sequence is often needed** To study and visualize the unaffected geometric properties and spatial relations between different-looking objects, we need to model, reconstruct, and inspect their topological evolution through the continuous change of shape or size of figures. The evolution is often a long deformation sequence, and every single change in the sequence requires computation in order to propose the change of *shape* (or *size*) while preserving the underlying structures.
- **Communicating the changes beyond 3D** An important special case of surfaces is the subject of knotted surfaces studied in 4-space. While closed curves in knot theory

✉ Hui Zhang
h0zhan22@louisville.edu

[1] University of Louisville, Louisville, USA

Published online: 04 July 2020

are studied in 3-space, knotted surfaces are studied in 4-space [27]. Some surfaces in 4-space appear to be knotted but are really unknotted. They can be "untied" in principle by a series of deformations developed by Dennis Roseman [29] during which the surface does not develop self-intersections; such deformations are examples of isotopies, and visualizing the deformation sequence in high-dimensional space with a series of impression in our dimension is a fundamental problem of interest to mathematical visualization applications.

Our task in this paper is to show the use of computer graphics, computational algorithm, and simulation to fully illustrate the topological evolution of mathematical surface in 4-space. We propose a family of visual and computational methods to depict and analyze mathematical surfaces in 4-space under the equivalence relation of homeomorphism. This paper starts from a familiar task of deforming and relating different-looking 3D strings, which we model as very "thin and flexible" rubber strings that can preserve the underlying topology while being relaxed. Having established the mechanisms and intuition of this artifice, we proceed to showing how the methods can be extended to surfaces and their deformations that occur beyond 3D—we think of *surface* as a fabric woven from those rubber strings, and extend the rubber string's dynamic model to move surfaces under topological constraints in 4-space. We showcase interesting mathematical phenomena and their visual proofs that can be generated from our visualization methods and interfaces, including the topological refinement of mathematical curves, visual understanding of spatial relationship of surfaces in 4-space, the study of knotted surfaces beyond 3D, and the transformation between the Klein bottle's different maps. By exploiting such visualization interfaces and methods, we feel that we can make a novel contribution to building intuition about classes of geometric and topological problems beyond our dimensions, speculated on by ancient Greek philosophers two millennia ago [1,2].

2 Background

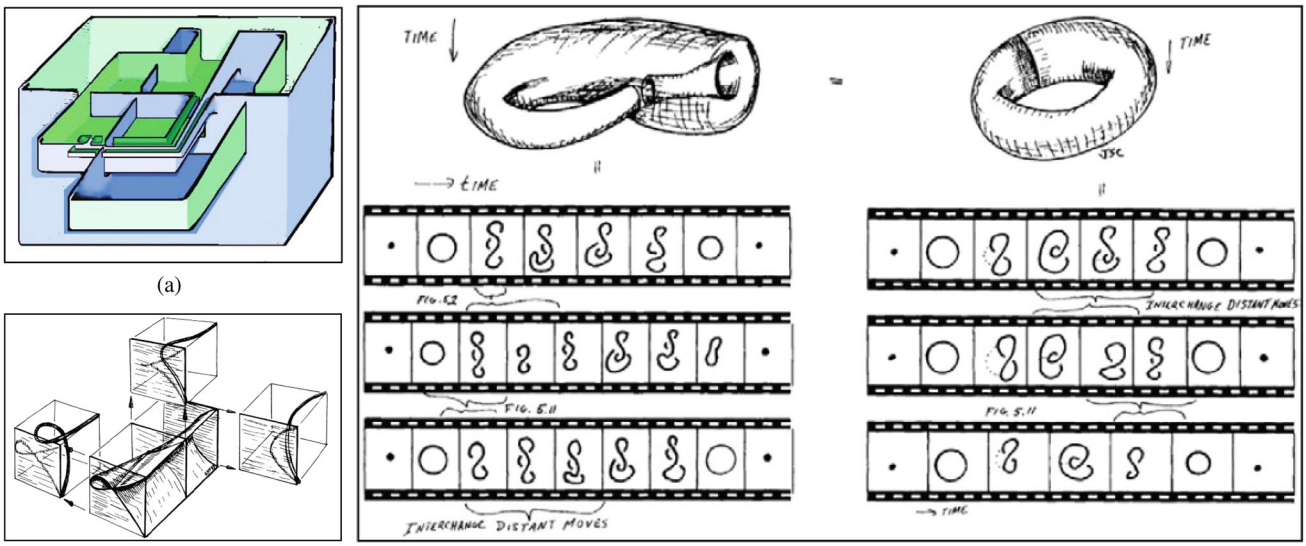
Relating two fundamentally identical entities that appear to very different has long been a fascinating challenge problem. For example, the idea of knot equivalence [19,22,32] is to give a precise definition of when two knots should be considered the same even when positioned quite differently in space. More challenging is to *move, deform, and relate* surfaces embedded, e.g., in 4D, and to respect a family of problem-specific constraints about location, collisions, etc. There is a fundamental need for methods in higher dimensions to "play with a manifold" as though it were a lump of stretchy clay being wrapped around other fixed objects in the environment [8,20].

Surfaces in 4D play many roles analogues to those of curves we are familiar with in 3D. For example, spheres (2-spheres) in 4-space are the analogs of closed curves (1-sphere) in 3-space, and 3D knots can be generalized to 4D "knotted surfaces" (more generally, an $N-$sphere can be tied in knots in $N+2$ dimensions [4].) When examining a particular geometric feature in 4D, one often relates it to an equivalent geometric feature in our dimension and uses strategies such as drawing analogous figures to model, illustrate, and analyze the problems. Furthermore, many spatially extended shapes can be initially drawn and communicated on paper and blackboard using curve analogies. In the past decade, several articles and books have been published that are such examples. One of these is Ralph Fox's *A Quick Trip Through Knot Theory* [14], where Fox uses hand-drawn intersecting curves to reveal the underlying structures and spatial relationships of 4D knotted surfaces (see, e.g., Fig. 1a). George Francis is another expert at popularizing 4D mathematical concepts. Francis' book *A Topological Picturebook* [15] includes a beautiful chapter that contains numerous high-quality illustrations of 4D surfaces, considered as disks interpolated by the motion of boundary curves (see, e.g., Fig. 1b). Equally beautiful is Scott Carter's *Knotted Surfaces and Their Diagrams* [20], a marvelous book of pictures to establish the fundamental concepts of 4-dimensional geometric topology with *movies*—sequences of diagrams to depict and analyze topological equivalence between 4D surfaces (see, e.g., Fig. 1c).

We are thus inspired to consider how to progress from the model and manipulation of a familiar but deformable 3D curve to a powerful but still simple paradigm for modeling and visualizing surfaces in 4-space. Typical geometric problems of interest in 4-space involve both static structures, and changing structures requiring deformation in four dimensions. Motivated by the understanding of how to deform and relax a mathematical knot in our dimension, we in this work bring the expressive power of curve analogies to the modeling and illustration of unfamiliar 4D surfaces that are otherwise beyond the experience accessible to us.

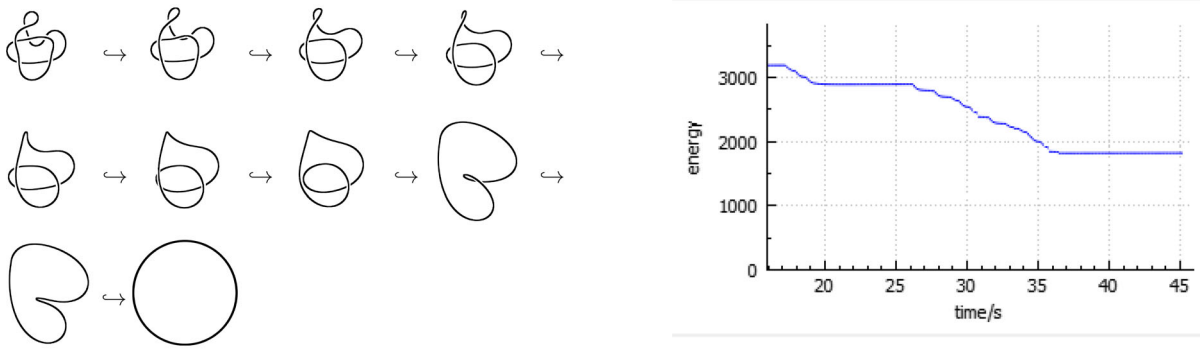
3 Relaxing a familiar 3D knot

In this section, we introduce the basic elements of the relaxation algorithm with a familiar 3D knot example, with the assumption that the algorithm will be extended to treat surfaces in 4D in a later section. We start with the creation and relaxation of a familiar curve piece in 3-space. We can shrink, expand, slide curves along one another. We are not allowed to cut them or slide them through others. Two curves are equivalent (or, the "same" from the topological point of view) if



(a) (b) (c)

Fig. 1 Examples of mathematical drawings to communicate geometry and topology. **a** Fox's hand-drawn equatorial cross sections to visualize a knotted sphere in 4-space. **b** Francis' book uses slice and shadow to represent 4D surfaces. The crossings on the boundary curves depict how the surfaces intersect in 4 dimensions. **c** Carter uses *movies* to analyze the topological equivalence between two Klein bottles, which appear very different in our dimension but represent the same embedding in 4-space



(a) (b)

Fig. 2 **a** A sequence of screen images showing a seemly knotted curve relaxed into a trivial unknotted piece. **b** The curve's energy reaches the minimum value when fully untangled

one can be deformed into the other. Note that we will stick with 2D diagrams to communicate the curve's structure and deformation in 3-space (see Fig. 2a), much like how you read about knots in the printed books; there is a hope that this experience of understanding 3-space with 2-dimensional impression will help us to communicate the surface's structure and deformation in 4-space using their 3D figures (with color and shading) in our next section.

We first need to create a mathematical 3D knot (a closed curve in 3-dimensional Euclidean space). This task can be facilitated with a 2D drawing interface. One can construct an initial configuration for an object while neglecting most issues of geometric placement (see Fig. 2a). This is possible, for example, when knot diagrams are drawn, only bare projections with relative depth ordering indicated at crossings while precise 3D depth information is unimportant at this step [30,39,40]. The next task is to refine the created initial embedding, ideally in a way that most of the work can be done automatically. Our method is largely based on force-directed algorithms, also known as spring embedders, that calculate and relax the layout of a graph (in our case, the linked nodes embedded in 3-space) using only information contained within the structure itself, rather than relying on domain-specific knowledge (see, e.g., the 1984 algorithm of Eades [10], and the use of spring embedders in 2D graph drawing [17,23] and 3D graph visualization [5,24]).

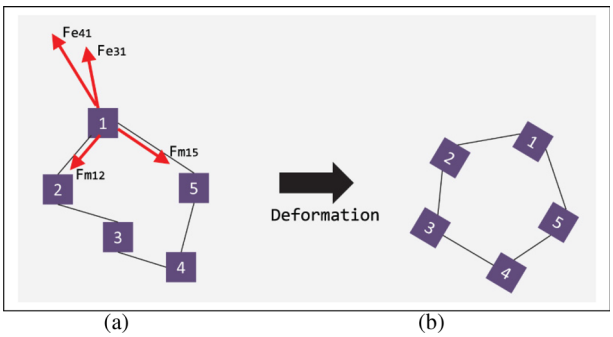


Fig. 3 Optimizing graph embedding through deformation. **a**, **b** The vertices are replaced with electro-statically charged masses that move the initial embedding to a relaxed and smooth configuration, ending at a lower energy state

3.1 The basic force model

To optimize the embedding of a graph, we replace the vertices with electro-statically charged masses and replace each edge with a spring to form a mechanical system. The vertices are placed in some initial layout and let go so that the spring systems and electrical forces on the masses move the system toward a reduced energy state.

As illustrated in Fig. 3, we use two basic forces in the mechanical system, an attractive *mechanical* force applied between adjacent masses on the same spring and a repulsive *electrical* force applied between all other pairs of masses. The mechanical force connects the masses and maintains the mechanical system's structure, and the electrical force tends to untangle the shape and deform the system into simplified shapes.

The mechanical force is a generalization of Hooke's law, allowing for an arbitrary power of the distance r between masses,

$$F_m = Hr^{\beta}, \tag{1}$$

where H is a constant. The mechanical force tends to attract adjacent masses toward each other.

The electrical force also allows for a general power of the distance,

$$F_e = Kr^{-\alpha}, \tag{2}$$

where r again is the distance between the two masses and K is a constant. The electrical force is applied to all pairs of masses excluding those consisting of adjacent masses on the same link. The electrical force repels all non-adjacent masses away from each other. In most of the examples shown in this paper, we use $\beta = 1$ and $\alpha = 2$.

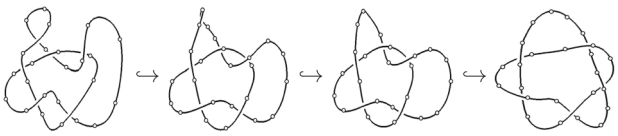


Fig. 4 Relaxation of a simple closed curve (a "Double Overhand Knot," also termed knot 5^1 in the Rolfsen Knot Table) with the proposed force laws and collision avoidance mechanism

3.2 Relaxing mathematical knots

For this force-directed algorithms to be applicable to mathematical curves positioned in $\mathbb{R}^3$, it is imperative that any proposed evolution should respect topological constrains: it does not involve cutting the curve or passing the curve through itself. Parallel to the force laws previously specified, the self-intersection problem is solved in our approach by requiring that the position of each mass be updated one at a time (no simultaneous updates), and collision avoidance is strictly performed to determine if one is heading toward one of the following two potential collisions:

- *point–segment collision*—a vertex of a 3D curve is going toward a link of the curve and the distance is less than the threshold
- *segment–segment collision*—a link of a 3D curve is going toward another link and their distance is less than the threshold

If either of these two states exists, the pair of closest points on the colliding components are identified to define a 3D vector passing through them. An equal (but opposite) displacement along the 3D vector is then applied to each component to take the component out of collision range.

Relaxing curves with dynamic model

We use a simple dynamical model that emulates a pseudo-physical situation of curve evolution [30]. During the simulation, each mass simply moves in the direction it is compelled to by the forces that are applied to it. The links or edges in the curves are not directly involved in the dynamics, although they provide constraints on the movement of the masses.

The collision avoidance process modifies masses' positions whenever necessary to ensure the entire evolution is under topological constraint. Figure 4 shows the dynamic model works to prevent unwanted intersections that might change the closed curve's topological features. Figure 2a shows an example where a seemly knotted curve can be relaxed into a trivial unknotted piece with our force model. Figure 2b plots the embedded energy of the curve during the entire deformation; the energy being calculated is the *minimum distance* or MD energy model defined in [32,36]: if a polygonal knot K consists of several edges $e_1, \ldots, e_n$, the

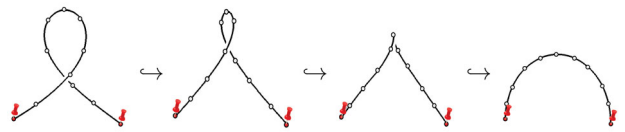


Fig. 5 Screen images of the topological relaxation of a curve, with masses turned off for the two end-points (much like we had two push pins down on the end-points)

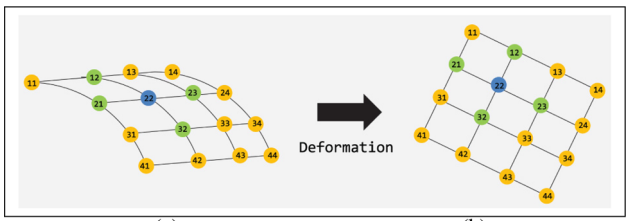


Fig. 6 We apply the attractive *mechanical* force between adjacent masses on the same spring and the repulsive *electrical* force between all other pairs of masses. For example, the node 22 on the fabric is attracted by nodes (12, 21, 23, 32), and repelled by all others, i.e., nodes (11, 13, 14, 24, 31, 33, 34, 41, 42, 43, 44). **a**, **b** Optimizing the surface's embedding through relaxation: the vertices move to the new locations and result in a smoother configuration

MD energy of K is defined as $E(K) = \sum \frac{L_i L_j}{D_{ij}^2}$, where L_i is the length of e_i, D_{ij} is the minimum distance between e_i and e_j, and the sum is taken over all non-adjacent edges.

Deforming with constrained motion In the current dynamic model, we assign identical masses to all the vertices. With a given force, each mass will respond in the same way (ignoring any topological constraints upon the movement). We can also lift this restriction of identical masses, and allow masses to be turned on and off. In our model, the mass indeed provides a scale factor on the amount of movement on each time step—when a mass is turned off, no movement can be produced for the corresponding vertex. In Fig. 5), a curve is relaxed while the two end-points stay fixed, and the whole relaxation appears much like a untwist Reidemeister move [26].

4 Relaxing surfaces in 4-space

In this section, our objects of interest have progressed from curves (1-dimensional objects) to surfaces (2-dimensional objects). Much like curve, surface is also infinitely malleable and infinitesimally thin. It can be hammered into any shape as desired; once shaped, it will retain its shape. The surface's size is of no concern—sometimes we need acres, and sometimes we need square inches. It has no thickness, and can be shrunk and stretched at will; however, the surface does not tear, nor does it puncture.

We think of surface as a fabric woven of those rubber strings that we have used to model and deform mathematical

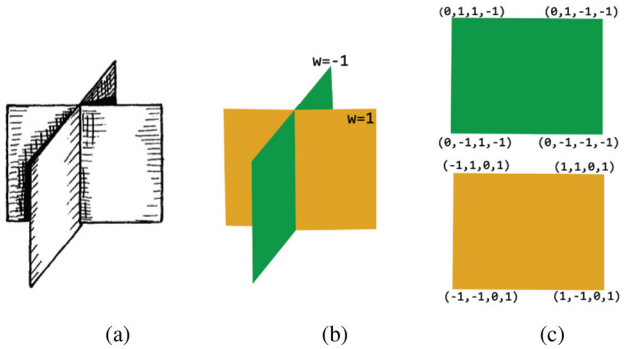


Fig. 7 Smooth 4D surfaces appear to intersect with each other in our dimensions. **a** A hand-drawn illustration of two surface patches embedded in 4D (by Carter in [6]). **b**, **c** A computer-generated 3D representation of the two 4D surface patches, 4D depth color-coded, corresponding to the diagram in **a**

curves. To define the structure and dynamics of the fabric, we have again used two basic forces—the attractive *mechanical* force applied between adjacent masses on the same spring and the repulsive *electrical* force applied between all other pairs of masses. The mechanical and electrical forces are still the generalization of Hooke's law as described in Eqs. 1 and 2. As illustrated in Fig. 6, each node on the fabric is attracted by up to four adjacent masses, and repelled by all the other masses in the fabric.

Our interest in this paper is *surface embedded in 4-space* [18]—a class of entities that are of interest to many mathematical problems. An important special case of surfaces in 4D is the subject of knotted surfaces. While closed curves are knottable in 3D, smooth curves (whether or not they are thickened) can always be untied without self-intersection in 4D. However, surfaces can be knotted in 4D. Some surfaces in 4D appear to be knotted but are really unknotted. Next we investigate methods to create, deform, and visualize surfaces in four dimensions. Our earlier example of self-deformable 3D mathematical strings has set the stage for our new problem: a surface to be relaxed in four dimensions.

Let us first understand Fig. 7—a pair of 4D-embedded surfaces drawn in Carter's book [6]. We have used 2D diagrams earlier to represent curves in 3-space; here we also use the similar scheme to describe a class of surfaces embedded in 4D with just their 3D figures. When a surface is embedded in 4-space, each vertex has a 4D "eye-coordinate," or depth w, in addition to the coordinates (x,y,z) that we are already familiar with.

Just as smooth 3D curves lose their structures when projected to 2D space, smooth 4D surfaces might appear to intersect with each other in our dimension (see, e.g., Fig. 7a). We have used "cutaways" in 2D knot diagram to visually indicate the string pieces' relative 3D depth order, we now use surface color coded for its 4D "eye-coordinate" to communicate their 4D depth. In Fig. 7, the two surface patches indeed

have no intersections of any kind in the fourth dimension, since one has a negative 4D "eye-coordinate" ($w = -1$, and colored in green) and the other one has a positive 4D "eye-coordinate" ($w = 1$, in yellow.)

4.1 Constructing surfaces in 4-space

In our work, we have constructed concrete examples of surfaces embedded in 4-space. 3D figures of 4D mathematical entities often look far more complicated than Fig. 7. They often twist, turn, and fold back on themselves, creating many important properties behind the surface sheet. Such self-intersecting surfaces are difficult to be created to represent the underlying 4D structures. Our methods to create concrete 4D embeddings are based on the following mathematicians' expositions:

Stacking curves method In Carter's book [6], a 4D embedded surface can be thought of a collection of curves in 3-space, just as if we use a plane to cut the surface in 4-space, and we observe a resultant curve on every single cutting plane (see Fig. 8a). The collection of the 3D curves becomes the bare projective skeleton of the 4D-embedded surface, and the relative depth order in the fourth dimension for patches containing self-intersections are indicated by the "cutaways" of the corresponding curves.

We continue to use Fig. 8 to show our proposed method for depicting a 4D surface patch embedded in four dimensions. The 4D surface, drawn on the book, appears to be a piece of 3D surface that intersects with itself (see Fig. 7a). However, each vertex on the surface has a 4D "eye coordinate" or depth w, and there are no real collisions in 4-space. The crossing-diagram on the surface's parametric domain, which can be thought of as an unfolded view of the 4D surface (see Fig. 7c), reveals the secret: we assign each vertex a 4D "eye-coordinate," or depth w: $w = 1$ for vertices on patches that are "in front" in the fourth dimension, $w = -1$ for vertices that are "behind." (*blue* is "in front," or nearer the projection point; *red* is "behind," or farther from the projection point; and *grey* means $w = 0$.) Therefore, the "north" half of the intersection line is "in front" or, nearer the 4D projection point, and the south part of the intersection line is "behind"; the two parts have no intersections of any kind in 4-space. Figure 7d shows the resultant 4D-embedded surface corresponding to Fig. 7a, with initial 4D depth values assigned to the vertices on the curve stack.

Spinning knot method
More complicated and interesting is a class of knotted surfaces in 4-space—surfaces can be knotted in 4D; some surfaces in 4D appear to be knotted but are actually unknotted. The earliest construction of knotted surfaces in 4-space was due to E. Artin in 1926 [3]. Friedman's exposition in [16] provides several more generalizations that introduce

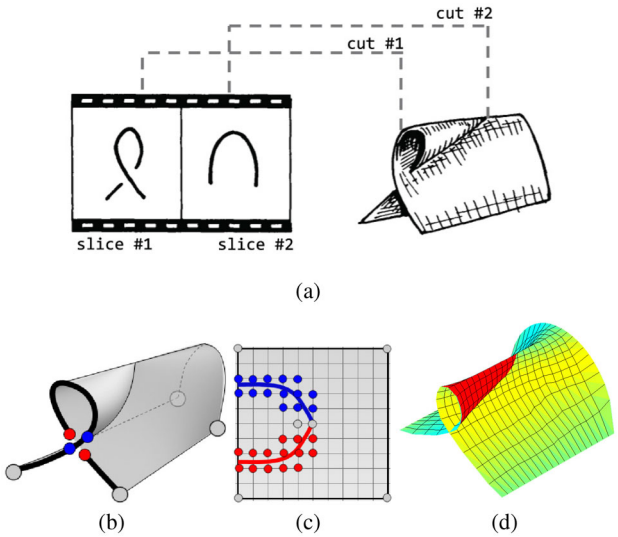


Fig. 8 Creating 4D embedding using Carter's slicing method. **a** Surface embedded in 4-dimensional space can be thought of as a sequence of 3D curves stacked together like the pages of a book. **b**–**d** The resultant 4D embedding we created

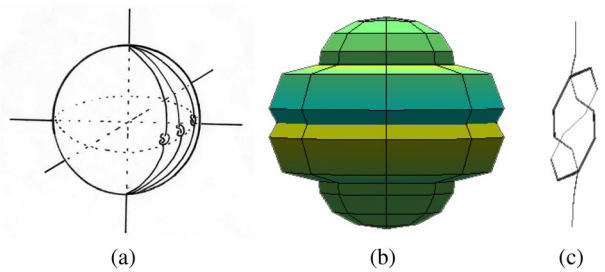


Fig. 9 **a** Schematic of knot spinning. **b** The 3D image of resultant 4D spun trefoil knot, projected from 4 dimensions to xyz space. **c** The 4D spun trefoil knot, projected from 4-space to yzw space, reveals the trefoil knot structure from which the knotted surface was spun

higher-dimensional knotted surfaces—embeddings of S^{n-2} in S^n—through spinning constructions. The method of creating 4D spun surfaces is "simple"—our shoe laces, those archetypal hand tools of knot theory in 3 dimensions, can be spun and turned into knotted spheres in the higher dimension.

As shown in Fig. 9a, a 3D trefoil curve swings around the globe with the north and south poles remaining fixed, and the knotted arc gets spun into the image of a 2-sphere S^2 embedded in 4-space. Although the image of the 2-sphere (see Fig. 9b) appears to have massive self-intersections in our dimensions, it has no collisions at all in the fourth dimension, and the relative 4D depth order of the resultant surface patches is exactly the same as the knotted arc from which the surface gets spun (see Fig. 9c).

4.2 Relaxing surfaces in 4-space

When progressing to self-deformable surfaces (2-dimensional objects) in 4-space, we implement two types of extensions from the mechanism we just established for deforming curves (1-dimensional objects) in 3-space:

- **string → fabric**—we extend our force-directed algorithms that calculate the layout of a 3D curve to a topological surface, i.e., from rubber strings to rubber fabric. The layout of our previous rubber string example is indeed a special type of graph, i.e., 1-dimensional linked nodes (see Fig. 3), and a topological surface is now 2-dimensional graph structure (see Fig. 6).
- **3-vector → 4-vector**—the second extension concerns the dimensions the masses are embedded in. We will still calculate and use the two basic forces, i.e., an attractive mechanical force applied between adjacent masses on the same spring and a repulsive electrical force applied between all other pairs of masses. The real difference is that now these masses are embedded in dimensions above three and the vector operations are performed in dimensions above three. For the most part, vector operations in high dimensions are simple extensions of their 3D counterparts [38]. For example, computing the addition of two four-vectors is a matter of forming a resultant vector whose components are the sum of the pairwise coordinates of the two operand vectors. In the same fashion, subtraction, scaling, and dot-products are all simple extensions of their more common three-vector counterparts. In addition, operations between four-space points and vectors are also simple extensions of the more common three-space points and vectors. Computing the four-vector difference of four-space points is a simple matter of subtracting pairwise coordinates of the two points to yield the four coordinates of the resulting four-vector.

Similar to 3D collision handling mechanisms, the space-extended idea is to prevent 4D collisions before they happen [37]. Let t_0 be an instant when there is no inter-penetration between the two polygons embedded in 4D. The basic idea is to consider a time interval $[t_0, t_0 + \Delta t]$, and knowing the positions and velocities of each node of the 4D surface model at time t_0, we can compute its positions at time $t_0 + \Delta t$ (we predefined a threshold value δ for the thickness of the 4D surface and any motion between two frames is clamped to be no greater than δ.) Collision avoidance then consists of determining if one is heading toward a potential collision. Two cases are considered to have to be avoided:

1. *point–triangle collision*—a vertex of a triangle on a 4D fabric is going toward another triangle and the distance is less than δ.
2. *edge–edge collision*—the edges of a triangle are going toward those of another triangle and their distance is less than δ.

We next detail our math formulas to calculate the two types of distances in our consideration above.

Distance between line segments in 4-space We first consider two infinite lines $\mathbf{L}_1$: $\mathbf{P}(s) = \mathbf{P}_0 + s(\mathbf{P}_1 - \mathbf{P}_0) = \mathbf{P}_0 + s\mathbf{u}$ and $\mathbf{L}_2$: $\mathbf{Q}(t) = \mathbf{Q}_0 + t(\mathbf{Q}_1 - \mathbf{Q}_0) = \mathbf{Q}_0 + t\mathbf{v}$. Let $\mathbf{w}(s,t) = \mathbf{P}(s) - \mathbf{Q}(t)$ be a vector between points on the two lines. The goal here is to find the $\mathbf{w}(s,t)$ that has a minimum length for all s and t. In any N-dimensional space, the two lines $\mathbf{L}_1$ and $\mathbf{L}_2$ are closest at unique points $\mathbf{P}(s_c)$ and $\mathbf{Q}(t_c)$ for which $\mathbf{w}(s_c, t_c)$ attains its minimum length. Also, if $\mathbf{L}_1$ and $\mathbf{L}_2$ are not parallel, then the line segment $\mathbf{P}(s_c) \leftrightarrow \mathbf{Q}(t_c)$ joining the closest points is uniquely perpendicular to both lines at the same time. No other segment between $\mathbf{L}_1$ and $\mathbf{L}_2$ has this property. That is, the vector $\mathbf{w}_c = \mathbf{w}(s_c, t_c)$ is uniquely perpendicular to the line direction vectors $\mathbf{u}$ and $\mathbf{v}$, and thus it satisfies the equations:

$$\begin{aligned} \mathbf{u} \cdot \mathbf{w}_c &= 0 \\ \mathbf{v} \cdot \mathbf{w}_c &= 0. \end{aligned} \tag{3}$$

We can solve these two equations by substituting $\mathbf{w}_c = \mathbf{P}(s_c) - \mathbf{Q}(t_c) = \mathbf{w}_0 + s_c\mathbf{u} - t_c\mathbf{v}$, where $\mathbf{w}_0 = \mathbf{P}_0 - \mathbf{Q}_0$, into each one to get two simultaneous linear equations. Then, letting $a = \mathbf{u} \cdot \mathbf{u}$, $b = \mathbf{u} \cdot \mathbf{v}$, $d = \mathbf{u} \cdot \mathbf{w}_0$, and $e = \mathbf{v} \cdot \mathbf{w}_0$, we solve for s_c and t_c as:

$$s_c = \frac{be - cd}{ac - b^2}, \quad t_c = \frac{ae - bd}{ac - b^2}. \tag{4}$$

Having solved for s_c and t_c, we have the points $\mathbf{P}(s_c)$ and $\mathbf{Q}(t_c)$ where the two lines $\mathbf{L}_1$ and $\mathbf{L}_2$ are closest. Then the distance between them is given by:

$$d(\mathbf{L}_1, \mathbf{L}_2) = \left| (\mathbf{P}_0 - \mathbf{Q}_0) + \frac{(be - cd)\mathbf{u} - (ae - bd)\mathbf{v}}{ac - b^2} \right|. \tag{5}$$

Now we represent a segment $\mathbf{S}_1$ (between endpoints $\mathbf{P}_0$ and $\mathbf{P}_1$) as the points on $\mathbf{L}_1$: $\mathbf{P}(s) = \mathbf{P}_0 + s(\mathbf{P}_1 - \mathbf{P}_0) = \mathbf{P}_0 + s\mathbf{u}$ with $0 \le s \le 1$. Similarly, the segment $\mathbf{S}_2$ on $\mathbf{L}_2$ from $\mathbf{Q}_0$ to $\mathbf{Q}_1$ is given by the points $\mathbf{Q}(t)$ with $0 \le t \le 1$. The distance between segment $\mathbf{S}_1$ and $\mathbf{S}_2$ may not be the same as the distance between their extended lines $\mathbf{L}_1$ and $\mathbf{L}_2$. The first step in computing a distance involving segments is to get the closest points for the lines they lie on. So, we first compute s_c and t_c for $\mathbf{L}_1$ and $\mathbf{L}_2$, and if these are in the range of the involved segment, then they are also the closest points for them. If they lie outside the range, only a few more boundary

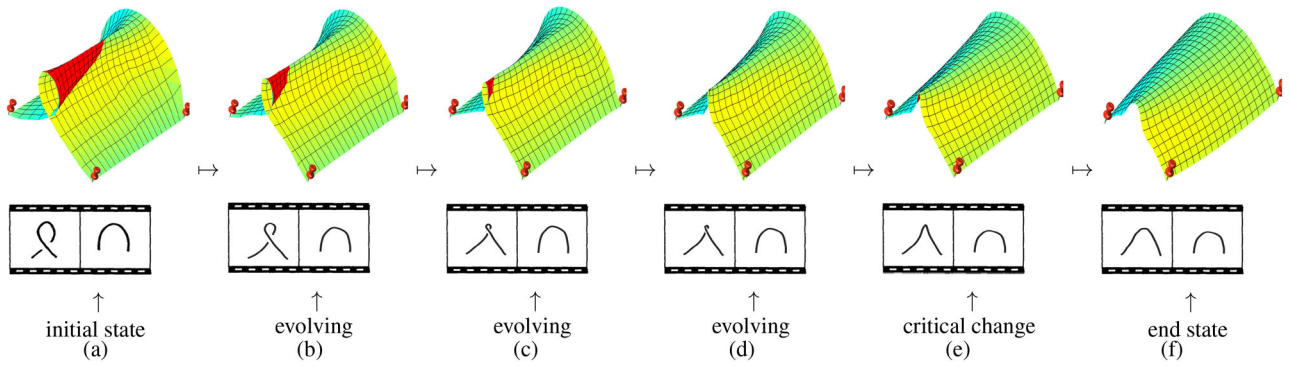


Fig. 10 **a**–**e** Applying our force-directed algorithms to the piece of 4D surface patch we were creating in Fig. 7. Our result shows that the smooth 4D-embedded surface patch evolves with the two basic forces and eliminates its interruptions in our dimension, which were just an artifact of projection from 4D to 3D. Among the 6 moves here, **a** and **e** are two "key moments" in that the surface's frontal boundary curve goes through a Type-I Reidemeister move. (note: we turned off the masses on the four corners as we did in Fig. 5)

tests are needed to compute the closest points between the line segments as suggested by [12].

Distance between point and triangle in 4-space The problem now is to compute the minimum distance between a point $\mathbf{P}$ and a triangle $\mathbf{T}(s,t) = \mathbf{B} + s\mathbf{E}_0 + t\mathbf{E}_1$ for $(s,t) \in D = (s,t) : s \in [0,1], t \in [0,1], s+t \leq 1$. The minimum distance is computed by locating the value $(\bar{s},\bar{t}) \in D$ corresponding to the point on the triangle closest to $\mathbf{P}$. Please note our algorithm should be working for triangle and point embedded in arbitrary dimensions.

The squared-distance function for any point on the triangle to $\mathbf{P}$ is $Q(s,t) = |\mathbf{T}(s,t) - \mathbf{P}|^2$ for $(s,t) \in D$. The function is quadratic in s and t,

$$Q(s,t) = as^2 + 2bst + ct^2 + 2ds + 2et + f, \tag{6}$$

where $a = \mathbf{E}_0 \cdot \mathbf{E}_0, b = \mathbf{E}_0 \cdot \mathbf{E}_1, c = \mathbf{E}_1 \cdot \mathbf{E}_1, d = \mathbf{E}_0 \cdot (\mathbf{B} - \mathbf{P})$, and $f = (\mathbf{B} - \mathbf{P}) \cdot (\mathbf{B} - \mathbf{P})$. Quadratics are classified by the sign of $ac - b^2$. For function Q,

$$\begin{aligned} &ac - b^2 = (\mathbf{E}_0 \cdot \mathbf{E}_0)(\mathbf{E}_1 \cdot \mathbf{E}_1) \\ &-(\mathbf{E}_0 \cdot \mathbf{E}_1)^2 = |\mathbf{E}_0 \times \mathbf{E}_1|^2 > 0. \end{aligned} \tag{7}$$

The positivity is based on the assumption that the two edges $\mathbf{E}_0$ and $\mathbf{E}_1$ of the triangle are linearly independent, so their cross product is a nonzero vector. In calculus terms, the goal is to minimize $Q(s,t)$ over D. Since Q is a continuously differentiable function, the minimum occurs either at an interior point of D where the gradient $\nabla Q = 2(as + bt + d, bs + ct + e) = (0,0)$ or at a point on the boundary of D.

The distance calculation is performed during each iteration to deform the 4D fabric. If either point–triangle or edge–edge collision is to occur, the pair of closest points on the colliding components are identified and l is defined as the 4D vector passing through them. Then an equal (but opposite) displacement along l is applied to each component along l, the displacement is just enough to take the component out of collision range.

Figure 10 shows the extended force models and dynamics can work to refine the piece of 4D surface patch we were creating in Fig. 7. The smooth 4D surface patch evolves with the two basic forces and collision avoidance, and eliminates its self-intersections in our dimension, which were just an artifact of projection from 4D to 3D. This relaxation can be thought of as 4D generalization of the 3D Reidemeister move in Fig. 5. In fact, Carter in [6,7] draws surfaces in 4-space as disks bounded by curves, and uses the boundary curves and their changes to define the surface's evolution in 4-space. Such changes can also be visualized in Fig. 7—(a) and (e) are two "key moments" identified in the entire 4D evolution since the surface's frontal boundary curve undergoes a Type-I Reidemeister move.

4.3 A few motivating 4D use cases

Many mathematical phenomena in 4-space are documented without formal visual representations. In this section, we will generate a few more pictures as experimental "visual proofs" to accompany the process of non-visual thinking in the math books and articles. Another value of generating these visualization is to validate our proposed models and algorithms in 4-space by applying them to these well-known and documented mathematical entities and phenomena.

Knots in 3-spaces and knotted surfaces in 4-space While closed curves are knottable in 3D, smooth curves (whether or not they are thickened) can always be untied without self intersection in 4D [18,28]. In four dimensions, any closed loop of one-dimensional string is equivalent to an unknot. We can achieve the necessary deformation in two steps: the first step is to "push" the loop into a three-dimensional subspace,

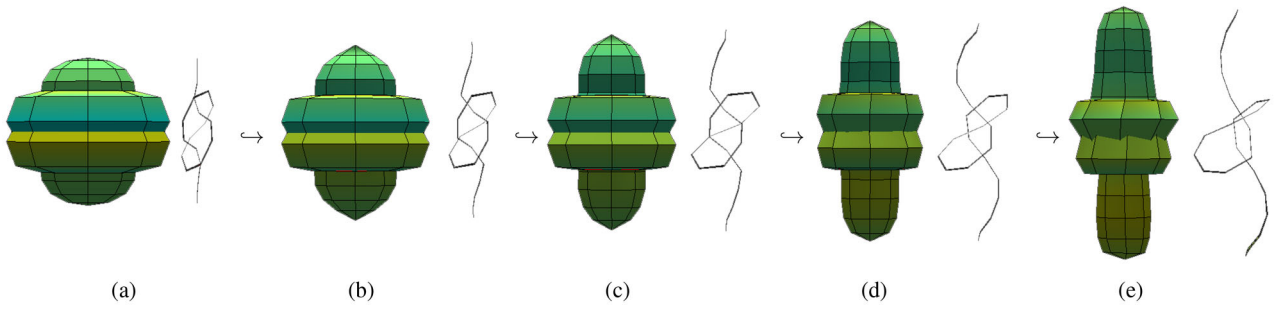


Fig. 11 Deformation of a 4D spun trefoil knot. **a**–**e** *left:* The knotted sphere projected from 4-space to xyz space, 4D depth colored. **a**–**e** *right:* Projection in yzw space, showing the trefoil knot structure from which the knotted surface was spun from

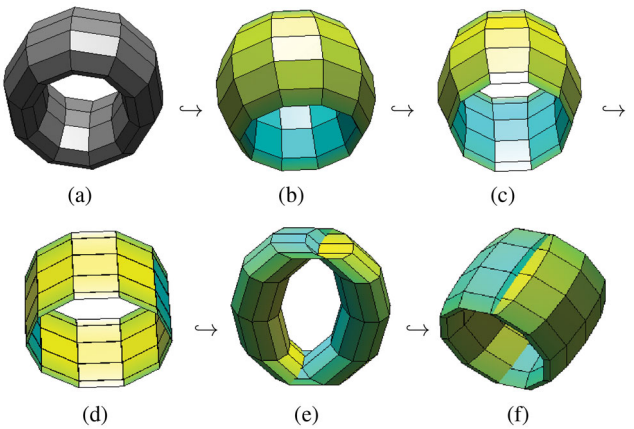


Fig. 12 Energy-based self-deformation turns the toroidal shape 3D torus (a) into a 4D embedded torus (e), with pinch-points appearing where the X-shaped fold-over occurs in the 3D projective image

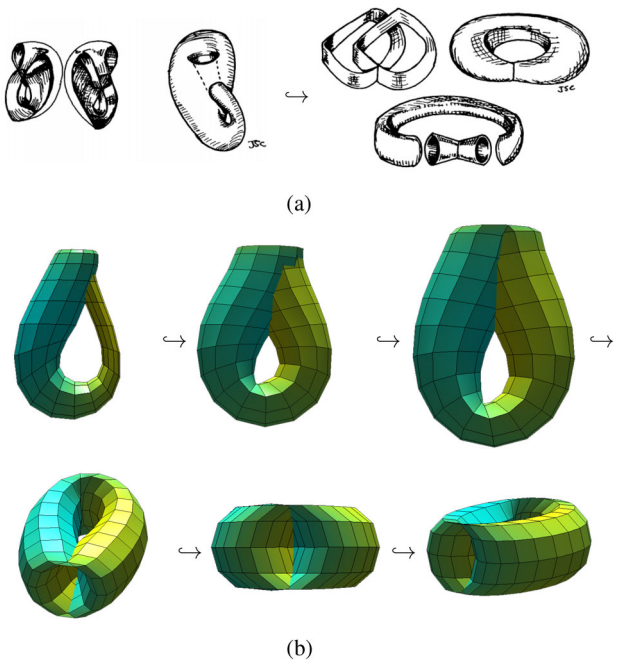


Fig. 13 **a** Cater's hand-drawn images to related the two different-looking Klein bottles. **b** Energy-based self-deformation turns the standard map of the Klein bottle into the pinched torus map

which is always possible, though technical to explain; the second step is changing crossings.

However, surfaces, e.g., the trefoil spun knotted surface that we created in Fig. 9, can be knotted in 4D. As illustrated in Fig. 11, when equipped with our 4D force model, the spun knotted sphere is tightened by the repulsive electrical forces between its north and south poles; however, it can't be untied at all—in 4-space, this 2-dimensional object is a truly knotted one and cannot be untied by repelling the two poles apart.

Transforming 4D torus Figure 12 shows an interesting result we have: the 4D forces and the dynamic model turn a 3D doughnut-like torus into a 4D embedded torus, an object of fundamental interest, with a standard model given by $X(u,v) = (\cos u, \sin u, \cos v, \sin v)$.

The deformation starts with a regular (doughnut-like) torus that we are familiar with (see Fig. 12a), except that we are now embedding the torus in 4-space with $w = 0$ (thus it is colored in gray, indicating it is initially flat in the fourth dimension). With a little jitter in the fourth dimension, the torus starts to deform to its optimal shape in the full-dimensional space. The 3D image of the deforming 4D-embedded torus starts be thin and flat in Fig. 12c–e contains self-intersections. The final stabled 4D torus shows an X-shaped fold-over in its 3D image, which is color-coded for 4D depth. Although the orthogonal 3D graphics projection of the 4D embedded torus has two lines of self-intersection, the true surface, just as illustrated in this topology-preserving evolution, is actually a smooth topological manifold in four dimensions.

Transforming Klein bottle into pinched torus The example in Fig. 13 is concerned with Klein bottle, first described in 1882 by the German mathematician Felix Klein. The Klein bottle is an example of a non-orientable surface — a two-dimensional manifold against which a system for determining a normal vector cannot be consistently defined. Informally, it 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 13a is from the hand-drawn diagrams by Carter [6] showing how the surface transforms from a regular map of Klein bottle (*left*) to a pinched torus map (*right*). Figure 13b shows a few representative frames extracted from our deformation sequence where the standard Klein bottle is shown first, followed a series of relaxation leading to a more symmetric version, and finally the "pinched torus" Klein bottle.

5 Extracting representative frames from relaxation

In this section, we introduce two different computations that can further improve our experiences with curve deformation in $\mathbb{R}^3$ and surface deformation in $\mathbb{R}^4$. The non-rigid deformation of shapes into one another produces a long sequence of intermediate shapes to allow a smooth visual effect. However, one interesting problem is that the deforming shapes often scale and rotate in the full dimensional space, which makes our observation and visual comparison of successive changes difficult in our own three dimensions. Therefore, our work here is particularly concerned with topology, while the deformation sequence consists of a large number of intermediate changes, what is mostly desired is a collection of "key moments" extracted from the deformation where only critical changes occurred.

5.1 Aligning intermediate shapes during relaxation

Under our proposed force model, deforming curves in $\mathbb{R}^3$ and surfaces in $\mathbb{R}^4$ are translation and orientation independent. Our improved deformation method takes advantage of this property by computing the best-fitting rigid transformation [?] [9,13,33] to provide alignment across intermediate shapes in the least squares sense. Let $\mathbf{p} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ and $\mathbf{q} = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n\}$ be two intermediate geometric shapes in $\mathbb{R}^d$. We can compute the optimal translation $\mathbf{t}$ and rotation R that align successive shapes with the following steps.

1. Compute the centroids of points of both shapes

$$\overline{\mathbf{p}} = \frac{\sum \mathbf{p}_i}{n}, \overline{\mathbf{q}} = \frac{\sum \mathbf{q}_i}{n}.$$

2. Compute two centered vectors, $\mathbf{x}$ and $\mathbf{y}$

$$\mathbf{x}_i := \mathbf{p}_i - \overline{\mathbf{p}}, \mathbf{y}_i := \mathbf{q}_i - \overline{\mathbf{q}}, i = 1, 2, \ldots, n.$$

3. Compute the $d \times d$ covariance matrix

$$S = XWY^T,$$

where X and Y are the $d \times n$ matrices that have $\mathbf{x}_i$ and $\mathbf{y}_i$ as their columns, respectively, and $W = diag(w_1, w_2, \ldots, w_n)$.

4. Compute the SVD (singular value decomposition): $S = U \sum V^T$. The desired rotation matrix, R, can be defined as follows,

$$R = V \begin{pmatrix} 1\,0\ldots & 0 \\ 0\,1\ldots & 0 \\ \vdots\,\vdots\,\ddots & \vdots \\ 0\,0\ldots & det(VU^T) \end{pmatrix} U^T.$$

5. Compute the optimal translation, $\mathbf{t}$, as follows:

$$\mathbf{t} = \overline{\mathbf{q}} - R\overline{\mathbf{p}}.$$

With the computed $\mathbf{t}$ and R, we now can align intermediate shapes in curve deformation sequence (embedded in $\mathbb{R}^3$), as well as in surface deformation sequence (embedded in $\mathbb{R}^4$). This method allows us to perceive the topological deformation by viewing and inspecting a series of smoothly transitioning images.

5.2 Reducing relaxation sequences to "key moments"

If we can select the most interesting viewpoints of the intermediate shapes before we align them, we can further improve the visual outcomes. This is possible, for example, by using the so-called *viewpoint entropy* method (see, e.g., [11,25,35]). In the best views of mathematical knots, their projected images will expose the least possible number of intersections and the largest possible projected area/length in the 2D view, and a fairly long deformation sequence can thus be reduced to just a set of frames that can be thought of as the "key moments" recorded when a critical change occurs. If we can align the "key moments" in the least squares sense, we then have a set of 2D impressions created to fully represent the deformation sequence. Given a viewpoint, the goodness is evaluated by the following three properties:

1. the projected length/area in the view $I(S, p) = -\sum p_i log(p_i)$, where the logarithms are taken in base two, and p_i is the relative projected area A_i/A_t of face i (A_i is the projected area of face i and A_t is the total projected area). Here larger length/area will contribute to a better viewpoint.

2. the number of intersections in the view, less intersections will contribute to a better viewpoint.
3. the square of the separation distance between the point sets in current view and the previous "key moment," less separation distance will contribute to a better viewpoint.

In our method, we place a set of viewpoints around deforming knots to select the best view for each moment, and the collection of views presenting the entire deformation can be further reduced to "key moments" with critical changes identified across successive terms (see, e.g., Figs. 2 and 12).

6 System environment and limitations

Our user interfaces (see Figs. 14, 15) are based on OpenGL and QT. The software runs on Dell PC desktop with an Intel *i*7 CPU and a high-performance graphics card supporting OpenGL. Although our system creates effective visualizations for a class of mathematical curves and surfaces, our approach does have the following limitations:

- This work leverages the energy-driven force model to relax surfaces in 4-space to help us visualize and analyze their underlying structures. The relaxation algorithm can only ensure that the energy converges to a local minimum. While our visual interface allows one to *pull and drag* the knots (and surfaces) to help the system better escape local minimum, full untangling a mathematical knot is not the objective of this work.
- The topological relaxation algorithm behind the deformation is energy-driven, and it moves the curves and surfaces from higher energy state toward lower energy state only. Choosing the right starting point is important to trigger and understand the deformation. For example, in Fig. 12 the deformation starts with the doughnut-like torus, which we are familiar with but actually has a higher energy than its X-shaped equivalent (many of us had thought otherwise.)
- Our method to create concrete 4D-embedded surfaces is mostly applicable to symmetric objects as illustrated in this paper. More comprehensive sketching and computer-aided geometric design interfaces will be needed to support more complicated surfaces in 4-space.
- Finally, relaxing surfaces embedded in 4-space is very compute-intensive, and currently we are only able to achieve real-time visualization with relatively lower resolution meshes. Similarly the selection of best views in our work is using the brute-force algorithm [34], and production of "key moments" is not done in a real-time fashion.

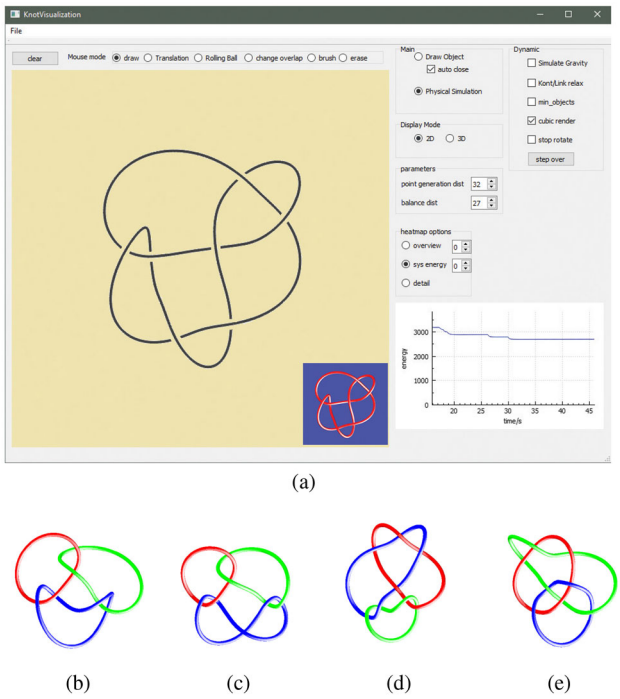


Fig. 14 **a** Typical screen image of relaxing mathematical knots. **b–e** A few sketched and smooth mathematical links (links 6_3^1, 7_3^1, 8_3^2, and 8_3^5)

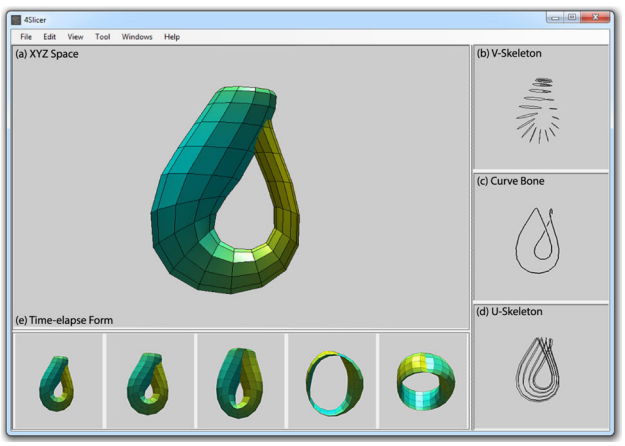


Fig. 15 Evolving a Klein bottle into a 4D pinched torus with topological relaxation. **a** The Klein bottle's projected figure immersed in three-dimensional (XYZ) space. **b–d** The Klein bottle represented in decomposed views: a curve skeleton described by v parametric space and geometrically shaped to a stack of cylinder rings in **b**, a curve bone that depicts the Klein bottle's structure in **c**, and a curve skeleton described by u parametric space and geometrically shaped to the connected cylinder strips in **d**. **e** Critical steps traced in a time-elapse form to visualize the entire topological relaxation process

7 Conclusion and future work

In this work, we adopt for the most part a computer scientist's perspective on the techniques and prospects of graph, algorithm, and computing in geometric topology, emphasizing mathematical curve and surface deformation where we

believe thinking in terms of figures could be valued as a means of facilitating grasp of linguistic text and hand-drawn analogies. Toward this goal, we have introduced a family of models and algorithms to move these mathematical entities embedded in 3- and 4-space, and user interfaces to visualize and track the deformation sequence. We have showcased the application of our methods and interfaces to a set of well-known and documented mathematical phenomena in high-dimensional space that otherwise is not accessible in our dimension.

Starting from this basic framework, we plan to proceed to attacking more geometric topology problems such as the interactive manipulation of apparently knotted, but actually unknotted spheres in 4D. Other planned future work will optimize the current algorithms and computations to support the real-time interactive manipulation of more complicated mathematical entities in space.

Acknowledgements This work was supported in part by National Science Foundation Grant #1651581 and the 2016 ORAU's Ralph E. Powe Junior Faculty Enhancement grant.

References

1. Abbott, E.A.: Flatland. Dover Publications, Inc., Mineola (1952)
2. Annas, J.: An Introduction to Plato's Republic. Clarendon Press, Oxford (1981)
3. Artin, E.: Zur isotopie zweidimensionaler flächen im r 4. In: Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg, vol. 4, pp. 174–177. Springer (1925)
4. Brown, R.: Mathematics and knots. In: Visual Representations and Interpretations, pp. 32–42. Springer (1999)
5. Bruß, I., Frick, A.: Fast interactive 3-d graph visualization. In: International Symposium on Graph Drawing, pp. 99–110. Springer (1995)
6. Carter, J.: How Surfaces Intersect in Space: An Introduction to Topology. K & E Series on Knots and Everything. World Scientific, Singapore (1995)
7. Carter, J.S.: Reidemeister/roseman-type moves to embedded foams in 4-dimensional space (2012)
8. Carter, J.S., Saito, M.: Knot diagrams and braid theories in dimension 4. Pitman Research Notes in Mathematics Series, pp. 112–112 (1995)
9. Chetverikov, D., Stepanov, D., Krsek, P.: Robust euclidean alignment of 3d point sets: the trimmed iterative closest point algorithm. Image Vis. Comput. **23**(3), 299–309 (2005)
10. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Algorithms for drawing graphs: an annotated bibliography. Comput. Geom. Theory Appl. **4**(5), 235–282 (1994)
11. Dutagaci, H., Cheung, C.P., Godil, A.: A benchmark for best view selection of 3d objects. In: Proceedings of the ACM Workshop on 3D Object Retrieval, 3DOR '10, pp. 45–50. ACM, New York, NY, USA (2010)
12. Eberly, D.H.: 3D game engine design: a practical approach to real-time computer graphics. CRC Press, Boca Raton (2006)
13. Fitzgibbon, A.W.: Robust registration of 2d and 3d point sets. Image Vis. Comput. **21**(13–14), 1145–1153 (2003)
14. Fox, R.H.: A quick trip through knot theory. Topology **3**, 120–167 (1962)
15. Francis, G.K.: A Topological Picturebook. Springer, Berlin (1987)
16. Friedman, G.: Knot spinning, pp. 187–208. Handbook of knot theory. Elsevier Science, Amsterdam (2005)
17. Fruchterman, T.M., Reingold, E.M.: Graph drawing by force-directed placement. Softw.: Pract. Exp. **21**(11), 1129–1164 (1991)
18. Hanson, A.J., Munzner, T., Francis, G.: Interactive methods for visualizable geometry. Computer **27**(7), 73–83 (1994)
19. Hass, J., Lagarias, J.C., Pippenger, N.: The computational complexity of knot and link problems. J. ACM (JACM) **46**(2), 185–211 (1999)
20. J. Scott, Carter, M.: Knotted Surfaces and Their Diagrams. American Mathematical Soc, Providence (1998)
21. Jordan, K., Miller, L.E., Moore, E., Peters, T., Russell, A.: Modeling time and topology for animation and visualization with examples on parametric geometry. Theor. Comput. Sci. **405**(1), 41–49 (2008)
22. Kawauchi, A.: A survey of knot theory. Birkhäuser, Basel (2012)
23. Kobourov, S.G.: Spring embedders and force directed graph drawing algorithms. arXiv preprint arXiv:1201.3011 (2012)
24. Kumar, A., Fowler, R.H.: A spring modeling algorithm to position nodes of an undirected graph in three dimensions. Tech. rep., Technical Report CS-94-7 (1994)
25. Li, C., Sun, Z., Song, M., Zhang, Y.: Best view selection of 3d models based on unsupervised feature learning and discrimination ability. In: Proceedings of the 6th International Symposium on Visual Information Communication and Interaction, VINCI '13, pp. 107–108. ACM, New York, NY, USA (2013)
26. Livingston, C.: Knot Theory, The Carus Mathematical Monographs, vol. 24. Mathematical Association of America, Washington (1993)
27. Paton, R., Neilsen, I.: Visual representations and interpretations. Springer, Berlin (2012)
28. Ranicki, A.: High-Dimensional Knot Theory: Algebraic Surgery in Codimension 2. Springer, Berlin (2013)
29. Roseman, D.: Twisting and turning in four dimensions. Video animation, Department of Mathematics, University of Iowa, and the Geometry Center (1993)
30. Scharein, R.G.: Interactive topological drawing. Ph.D. thesis, Department of Computer Science, The University of British Columbia (1998)
31. Simmons, G.F., Hammitt, J.K.: Introduction to Topology and Modern Analysis. McGraw-Hill, New York (1963)
32. SIMON, J.K.: Energy functions for polygonal knots. J. Knot Theory Ramific. **03**(03), 299–320 (1994)
33. Sorkine, O.: Least-squares rigid motion using SVD. Tech. Not. **120**(3), 52 (2009)
34. Vázquez, P.P., Feixas, M., Sbert, M., Heidrich, W.: Viewpoint selection using viewpoint entropy. In: VMV '01: Proceedings of the Vision Modeling and Visualization Conference 2001, pp. 273–280. Aka GmbH (2001)
35. Vázquez, P.P., Sbert, M.: Fast adaptive selection of best views. In: Kumar, V., Gavrilova, M.L., Tan, C.J.K., L'Ecuyer, P. (eds.) Computational Science and Its Applications – ICCSA 2003, pp. 295–305. Springer, Berlin (2003)
36. Wu, Y.: An md knot energy minimizing program. Department of Mathematics, University of Iowa
37. Zhang, H., Hanson, A.: Shadow-driven 4d haptic visualization. IEEE Trans. Vis. Comput. Gr. **13**(6), 1688–1695 (2007)
38. Zhang, H., Hanson, A.J.: Physically interacting with four dimensions. In: G. Bebis, R. Boyle, B. Parvin, D. Koracin, P. Remagnino, A.V. Nefian, M. Gopi, V. Pascucci, J. Zara, J. Molineros, H. Theisel, T. Malzbender (eds.) ISVC (1), *Lecture Notes in Computer Science*, vol. 4291, pp. 232–242. Springer (2006)
39. Zhang, H., Weng, J., Hanson, A.: A pseudo-haptic knot diagram interface. In: Proc. SPIE 7868, vol. 786807, pp. 1–14 (2011)

40. Zhang, H., Weng, J., Jing, L., Zhong, Y.: Knotpad: visualizing and exploring knot theory with fluid reidemeister moves. IEEE Trans. Vis. Comput. Gr. **18**(12), 2051–2060 (2012)

Hui Zhang is assistant professor at the Computer Science and Engineering Department, University of Louisville, USA. Zhang received his Ph.D. from the Computer Science Department at Indiana University, USA. Zhang's research interests include data visualization, data mining, and computer graphics with applications in mathematical visualization, medical and dental computing, and other big data problems.

Huan Liu received B.S. degree in Information Systems and Management from Communication University of Zhejiang, China, in 2018. Liu is currently pursuing Ph.D. degree in University of Louisville, USA. Liu's research interests include mathematical visualization and computer graphics.