Accelerating visual communication of mathematical knot deformation

Juan Lin & Hui Zhang

Journal of Visualization

ISSN 1343-8875

J Vis
DOI 10.1007/s12650-020-00663-w

Springer

J Vis
https://doi.org/10.1007/s12650-020-00663-w

REGULAR PAPER

Juan Lin · Hui Zhang

Accelerating visual communication of mathematical knot deformation

Received: 29 January 2020 / Revised: 8 April 2020 / Accepted: 15 May 2020

Abstract Mathematical knots are different from everyday ropes, in that they are infinitely stretchy and flexible when being deformed into their ambient isotopic. For this reason, challenges of visualization and computation arise when communicating mathematical knot's static and changing structures during its topological deformation. In this paper, we focus on visual and computational methods to facilitate the communication of mathematical knot' dynamics by simulating the topological deformation and capturing the critical changes during the entire simulation. To improve our visual experience, we design and exploit parallel functional units to accelerate both topological refinements in simulation phase and view selection in presentation phase. To further allow a real-time keyframe-based communication of knot deformation, we propose a fast and adaptive method to extract key moments where only critical changes occur to represent and summarize the long deformation sequence in real-time fashion. We conduct performance study and present the efficacy and efficiency of our methods.

Keywords Knot untanglement · View selection · Least squares fitting · Parallelization

1 Introduction

One of the fundamental problems in knot theory (Adams 2004) is to determine whether a closed mathematical curve can be deformed into a ring (or "unknot") without cutting or passing through the curve itself. The objects being studied, i.e., mathematical knots, are familiar and appear similar to the 3D ropes in our everyday life, except that the mathematical ones are infinitely stretchy and flexible during deformation to their topological equivalence.

The mathematical way of untangling a knot is to perform Reidemeister moves (Trace 1983), which reduce all knot deformations to a sequence of three types of "moves" called the *twist* move, *poke* move, and *slide* move. In principle, knot untanglement can be presented as a sequence of such simple (and powerful) moves. However, choosing the right combination of the Reidemeister moves in the right order can be very challenging. The whole procedure is often error-prone and thus requires technical expertise. Our goal in this paper is to develop an interactive visual tool that requires minimal knot theory expertise to model and present mathematical knots that can evolve and untangle by themselves into simplified structures. We start with a family of interactive methods to sketch mathematical knots as closed node-link diagrams with "energy" charged at each node. In this way, mathematical knots untangle by themself from a higher energy state to the lower. The complete untanglement can take a fairly long sequence of deformation to simulate. We then proceed to develop an algorithm to capture the key visual frames from the entire simulation where

J. Lin · H. Zhang (✉)
Computer Engineering and Computer Science Department, University of Louisville, Louisville, KY 40222, USA
E-mail: hui.zhang@louisville.edu

Published online: 16 June 2020

successive terms in the sequence differ only by critical changes. The key frames become the "snapshots" of the mathematical movies for us to visualize, explore, and track the entire deformation. The work presented in this paper extends our prior work published in Lin and Zhang (2019) by exploiting parallel computing and more efficient computational algorithm to extract the key visual frames in a much faster means.

The rest of our paper is organized as follows. In Sect. 2, we will review related work and existing interfaces in mathematical knot visualization, as well as computational methods to identify "snapshots" to represent a long sequence of visual frames. In Sects. 3 and 4, we describe the basic force models to untangle knots and methods to identify key frames in our knot interface. In Sect. 5, we discuss the use of parallel computing to accelerate the knot deformation and key frame selection to pursue performance improvement. In Sect. 6, we propose an even faster approach to extracting key frames in real-time fashion for better communicating knot deformation, and the conclusions section will follow.

2 Related work

The advent of interactive computer graphics and visualization has opened a new era for creating a tangible experience with these mathematical objects and their topological phenomena (Liu et al. 2019). Several researchers have devoted to facilitating mathematical knot understanding through various computer interfaces and visualization tools. For example, Knotplot (Scharein 1998) is widely used in drawing and interacting to draw and interact with mathematical knots. KnotApp (Carlen 2010) presents numerical algorithms to create knot shapes from their Fourier representations. KnotSketch (Costagliola et al. 2016) adopts an enhanced version of Gauss code to facilitate the manipulation of virtual knots. Zhang et al. develop a family of user interfaces to visualize knots in 3D and 4D space (Zhang et al. 2007, 2014; Lin and Hui 2019).

There are a range of research efforts involving graph layout algorithms in knot visualization. Pach and Tardos (2002) discuss the computational complexity needed to untangle polygons. Eades (1984) improves graph layout with spring embedder algorithm, a method that only uses the graph's structural information rather than domain-specific knowledge (Kobourov 2012). Similar methods have been applied in large and dynamic graphs (Harel and Koren 2000; Erten et al. 2004; Chen et al. 2019). How to minimize the number of edge-crossings is an important research question in graph drawing, proved as NP-complete in Garey and Johnson (1983). Schaefer (2013) surveys the rich variety of crossing number invariant, considered as a popular tool in graph drawing and visualization. Along with these research efforts, several others have also been focused on simulating mathematical knot's dynamics and topological deformation. For example, Snibble et al. merge springs and constrains with the study of geometric characteristics relevant to knots (Snibbe et al. 1998). Zhang et al. suggest studying mathematical knot equivalence by making step-by-step Reidemeister moves with a computer graphics interface (Zhang et al. 2012, 2016). Wu's MING (XXXX) is a knot drawing and refinement tool that utilizes energy-based minimization model and turns knot untanglement into a computational simulation.

In this paper, we are mainly concerned with how to computationally untangle mathematical knots and visually communicate their topological phenomena by extracting the key moments from the deformation sequence, which often consists of hundreds of thousands of visual frames. The energy model allows complex knotted structures to evolve and untangle by themselves toward simplified structures, and the evolution takes a large number of computational iterations to complete and track. We are motivated to investigate how to computationally identify and capture the key moments when critical changes occur during the deformation and integrate the most efficient and effective way of presenting these key visual frames to our knot interface, in order to communicate, track, and navigate the mathematical knot's evolution in a clear and intuitive visual means. While our basic method for untangling mathematical knots is similar to the energy-based minimization model in Wu (XXXX), the heart and soul of this work is how to untangle the knots much faster and how to communicate this process with just a few visual frames we can extract from the knot deformation.

3 Overall scenario

Figure 1 shows the overall visual communication scenario for our objects of interest, i.e., mathematical knots. Our tool derives from the familiar pencil-and-paper process of drawing 2D knot diagrams, with visual interfaces to allow the creation of mathematical knots, the exploration of the resultant geometric structures,

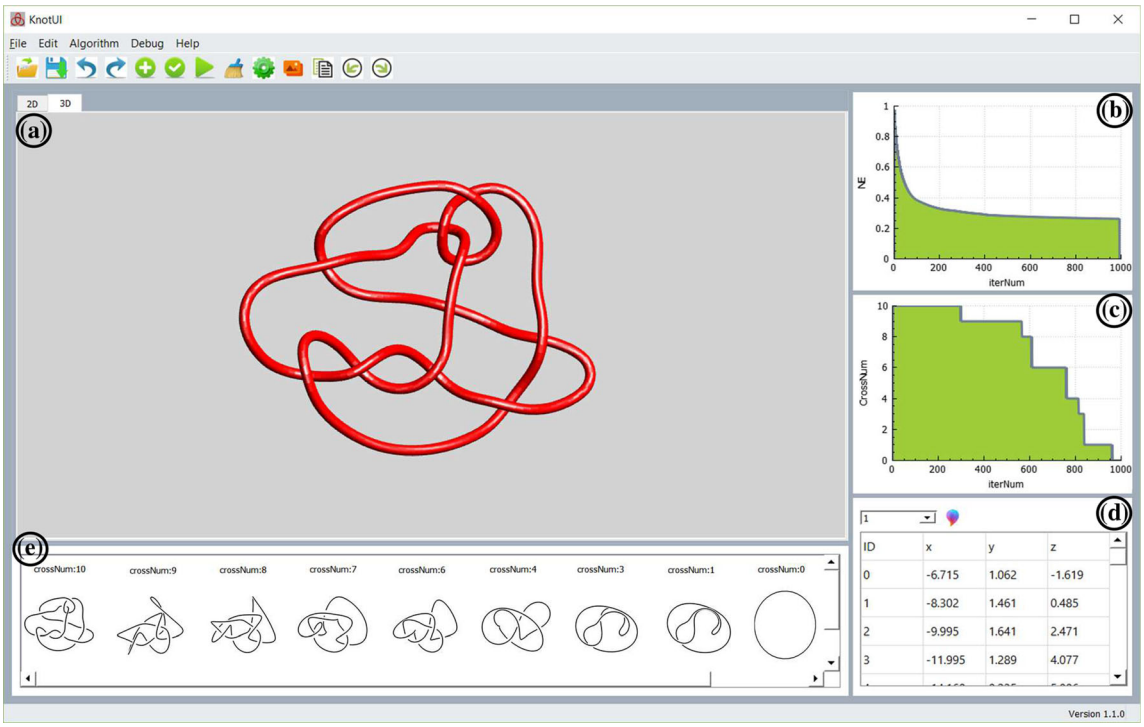

Fig. 1 Visually communicating key moments of mathematical knot deformation: our tool generates, visualizes, and tracks the dynamics of mathematical knot deformation. The tool's user interface elements include: ⓐ—the major visual panel to create, edit, and visualize mathematical knot deformation that automatically simplifies and optimizes knot's 3D structures; ⓑ and ⓒ—panels to plot the changing knot's energy and the resulting crossing number in real time; ⓓ—a property grid that provides access to the knot's geometric information, and ⓔ—the "snapshot" viewer that summarizes the deformation sequence with a set of key moments when only critical changes occur during the deformation

and the multiple synchronized views to observe the knot untanglement rendered as continuous simulation in 3D and 2D, and a sequence of captured key frames to represent all the "critical changes" that have taken place in the entire deformation. The four major user interface elements are listed as follows:

Central Panel—the major visual panel (see Fig. 1a) to create, edit, and visualize mathematical knots and their continuous topological phenomena. One can import and export mathematical knot structures from the *File* menu. Multiple data formats are supported including those from "Knot Zoo" (Scharein 1998). Various knot operations can be enabled by using the *Edit* menu, such as *edit knot*, *run simulation*, *undo* and *redo*, etc. From the *Algorithm* menu, we can choose to use the different simulation kernels for relaxing knots, which we will detail in the next few sections.

Feature Windows—windows (see Fig. 1b, c) used to plot the changing knot's energy and the resulting crossing number in real time. Knot energy is calculated based on the minimum distance (MD) energy model (Simon 1994). Deforming knots will tend to evolve to its simplified construction with minimal crossing number and knot energy. The plotting of knot energy and crossing number is synchronized with the rendering in central panel.

Property Grid—a window (see Fig. 1d) that provides access to the knot's geometric information, i.e., each node's x, y, z values.

Snapshot Viewer—the "snapshot" viewer (see Fig. 1e) on the bottom of our tool can summarize the deformation sequence with a set of moments when only critical changes occur during the deformation. Each snapshot can be selected to backtrack the corresponding moments in deformation.

4 Knot deformation and key moments

In this section, we introduce our basic user interfaces and methods to create and represent mathematical knots, the algorithm to optimize knots' construction, and our approaches to identifying and capturing the critical changes from knot deformation.

4.1 Deforming knot as node-link diagram

Mathematical knots are represented as node-link diagrams in our work. An initial diagram of a 3D curve can be obtained by projecting each vertex from $\mathbb{R}^3$ ($xyz-$space) to $\mathbb{R}^2$ ($xy-$space) (Fig. 2).

Let $\mathbf{K} = (\mathbf{V}, \mathbf{E})$ represent this initial diagram of a given smooth curve in $\mathbb{R}^3$, where $\mathbf{V} = \{\mathbf{v_1}, \mathbf{v_2}, ldots, \mathbf{v_n}\}$ is the finite set of *vertices* of the polygon and $\mathbf{E}$ is the set of edges. Our basic force model is applied as follows: The vertices on the knot are replaced with electrostatically charged masses and each segment linking between two vertices becomes a stretchy line to form a mechanical system (Scharein 1998). The masses are placed in the initial layout, and the forces will move the system to a stable state. Two types of forces are implemented—an **attractive mechanical force** applied between adjacent masses on the same component,

$$\begin{aligned}\mathbf{F_a(i)} =& H_a||\mathbf{v_{i+1}} - \mathbf{v_i}||^{\beta}(\widehat{\mathbf{v_{i+1}} - \mathbf{v_i}}) \\ &+ H_a||\mathbf{v_{i-1}} - \mathbf{v_i}||^{\beta}(\widehat{\mathbf{v_{i-1}} - \mathbf{v_i}})\end{aligned} \tag{1}$$

where $i \in [1, n]$, (if $i = 1$, $i - 1 = n$; if $i = n$, $i + 1 = 1$); H_a is a constant; $(\widehat{\mathbf{v_{i+1}} - \mathbf{v_i}})$ represents the vector pointing from $\mathbf{v_i}$ to $\mathbf{v_{i+1}}$; similarly, a **repulsive***electricalforce* is applied between all non-adjacent pairs of masses,

$$\mathbf{F_r(i)} = \sum_{||i-j|| > 1} H_r||\mathbf{v_i} - \mathbf{v_j}||^{\alpha}(\widehat{\mathbf{v_i} - \mathbf{v_j}}) \tag{2}$$

where $i, j \in [1, n]$, H_r is a constant. In our studies (Lin and Hui 2019), α=-6, β=2.

The knot deformation driven by the above two forces will update each node's position with a distance up to $0.2 * R$ at each iteration, where R is the predefined thickness of the knot. In this way, the knot deformation will respect the topological constrains: The mathematical knots should stretch or move around without cutting the or passing through itself. At each iteration, after the update of new position for each mass, collision avoidance is strictly performed to detect potential collision between segments. If one segment is going toward another and their distance is less than $\delta < 2 * R$, the two components will be pulled out of the collision range by moving an equal but opposite distance ($|\mathbf{v}| = R - \delta/2$). Figure 3 shows an example deformation where an initial knot diagram is being simplified into a smooth trefoil knot with our proposed force model after iterations.

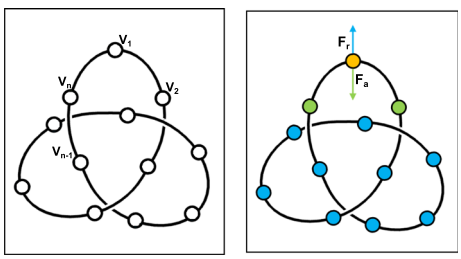

Fig. 2 Deforming mathematical knot with an underlying node-link structure. **a** A smooth knotted structure represented with an array of line segments, nodes, and interpolation splines. **b** The basic force model includes an *attractive* force applied between a node (colored in yellow) and its adjacent masses on the same component (colored in green), and an *repulsive* force applied between a node and its non-adjacent pairs of masses (colored in blue)

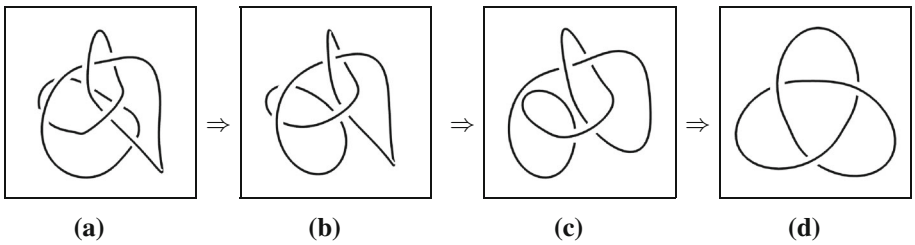

Fig. 3 Deformation of a trefoil knot with the two forces and collision avoidance

4.2 Extracting key moments of deformation

Force-guided knot deformation usually involves a large number of computational iteration. For example, for a "monster" knot with 86 vertices to be fully untangled (see, e.g., Fig. 4), it may take over 10,000 iterations before it reaches its final structure. This poses a significant challenge to visualize and trace the entire knot deformation. Furthermore, a large portion of the deformation are just about geometric updates that are not relevant to the topological question. Communication about the deformation can potentially be improved if we can just focus on those moments of the deformation where only critical changes have occurred.

4.2.1 Identifying critical changes

To extract key moments of the deformation, the first step is to identify the critical changes that occur in the mathematical deformation. In the mathematical area of knot theory, the *crossing number* of a knot is the smallest crossing number from all projective diagrams of the knot. It is a knot invariant—when the knot's crossing number changes, a *critical change* occurs. Calculating the crossing number is not trivial: To find the best diagram with the minimal number of crossings, every single possible diagram (from different projection) needs to be examined and the crossing number in that diagram needs to calculated before we know the minimum crossing number of the knot.

Our approach to finding this best diagram (with minimal number of crossings) is based on the widely used viewpoint entropy from Information Theory (Vázquez et al. 2001). A larger entropy value means more desired information is included in the view (i.e., the projective diagram in our case). Our entropy formula features the total length of the knot and the crossing number in the projective diagram:

$$I(K) = \sum_{i=1}^{N} \left(-\frac{L_i}{L_t} log \frac{L_i}{L_t} + V(i) \right) \tag{3}$$

where L_i represents the projected length of curve segment i, L_t is the total length of the knot projected curve, $V(i)$ is the visibility test function for curve segment i, $V(i) = -1$ if the segment is crossed by another segment, $V(i) = +1$ otherwise. Here, the larger length of projected curves will contribute to a larger entropy value, and the number of crossings (collisions) in the projection contributes to the entropy value as a penalty. In this way, the best diagram is identified as one that contains the maximal length of knot and the minimal number of crossing in the diagram.

With the viewpoint entropy defined above, the knot during the deformation is examined with a huge sampling space of projections, that is, the knot is rotated from 1 to 360 degree incrementally around x, y, and z axis, respectively. The entropy value is then calculated upon each diagram to search for the best diagram with the minimal crossing number and maximal projected knot length. During the knot deformation, when the knot's crossing number is changed, a critical change has occurred (see, e.g., Fig. 4 lists the nine critical changes captured from the untanglement of the "monster" knot.)

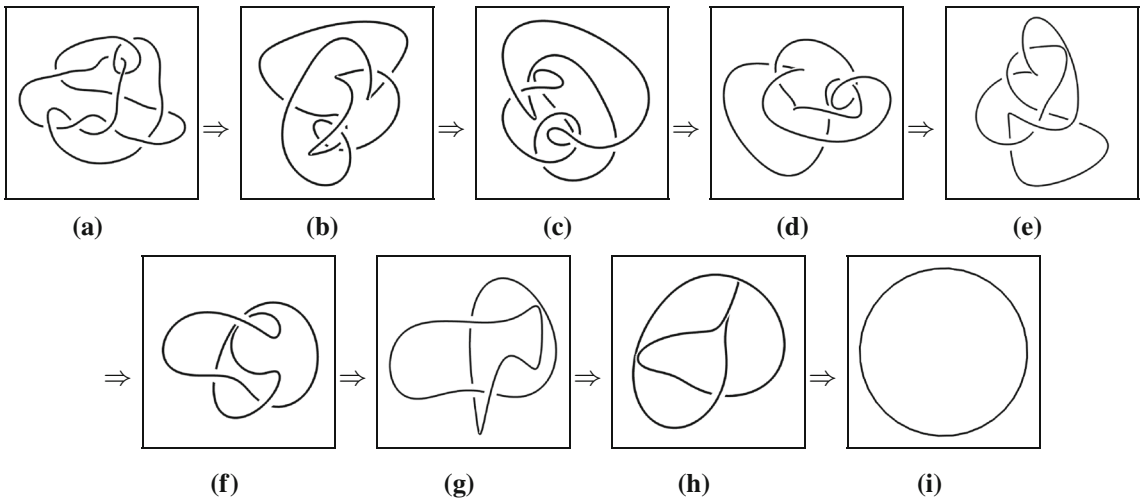

Fig. 4 Extracted critical changes of a "monster" unknot being untangled. These frames do not preserve their original orientations in the deformation. **a** Initial conformation with *crossNum* = 10. **b** *crossNum* = 9. **c** *crossNum* = 8. **d** *crossNum* = 7. **e** *crossNum* = 6. **f** *crossNum* = 4. **g** *crossNum* = 3. **h** *crossNum* = 1. **i** Final conformation with *crossNum* = 0

4.2.2 Retrieving key moments of critical changes

The viewpoint entropy method can help us identify the critical changes from the knot deformation. However, the diagrams representing the critical changes exhibit visual discontinuities from one diagram to another. This is because the viewpoint entropy method searches for the best diagram with the minimal crossing number from rotated knots with all possible angles in the three-dimensional space. The resultant diagrams extracted from each critical changes do not preserve their orientations in the original deformation (see, e.g., the visual discontinuities arise in the successive diagrams in Fig. 4).

Algorithm 1: Extraction of Key moments

Input: Initial Layout
Output: A Key Moment

```
Initialize the current projection P_max with the initial rotating angle, best entropy
  v_max, minimum distance d_min ;
Rotate the knot from 1 to 360 degree incrementally around x, y, z respectively ;
Calculate current entropy v;
if v' >= v_max then
    Calculate current distance d ;
    if d < d_min then
        Update P_max ;
        Update v_max with v ;
        Update d_min with d ;
    end
end
```

Algorithm 2: Knot untanglement with key moments presentation

Input: Initial Layout
Output: A updated Layout
while the stop condition is not satisfied **do**
  for each vertex i in the knot **do**
    Calculate the attractive force $\mathbf{F_a(i)}$, the repulsive force $\mathbf{F_r(i)}$;
    Calculate the total $\mathbf{F_t(i)} = \mathbf{F_a(i)} + \mathbf{F_r(i)}$;
  end
  Calculate the maximum magnitude f of all forces ;
  for each vertex i in the knot **do**
    Update the position $\mathbf{v_i}' = \mathbf{v_i} + 0.2 * R * \mathbf{F_c(i)} * \frac{||\mathbf{F_c(i)}||}{f}$;
  end
  for each vertex i in the knot **do**
    Collision Detection and adjust position;
  end
  Obtain the key moments representation with Algorithm 1
end

To extract the original key moments where these critical changes occurred, we need to align the visual frames of critical changes using the least squares method (see Eq. 4). When searching for the optimal diagrams for critical changes, we choose the visual frame with the minimal number of crossing **and** the least square distance from the previous key frame. This new searching criterion will ensure the extract key frames to preserve the best visual continuity in the extracted sequence. In Algorithm 1, we describe the process of finding the key moments in knot deformation. Algorithm 2 gives a framework for the complete knot relaxation with the key moment finding. Figure 5 shows the sequence of such key moments obtained from the "monster" unknot's deformation. Compared to Fig. 4, the "aligned" key moments in Fig. 5 can provide a better visual communication about the original mathematical deformation.

$$d(\mathbf{K}_p, \mathbf{K}_q) = \frac{1}{n}\sum_{i=1}^{n} \left\| \mathbf{v}_{pi} - \mathbf{v}_{qi} \right\| \tag{4}$$

In Figs. 6 and 7, we show another example of extracting key moments from the deformation of a $Knot_5^1$. Figure 6 shows the sequence of critical changes captured by our view entropy-based searching method. Figure 7 is the result of visually continuous key moments to represent the entire deformation.

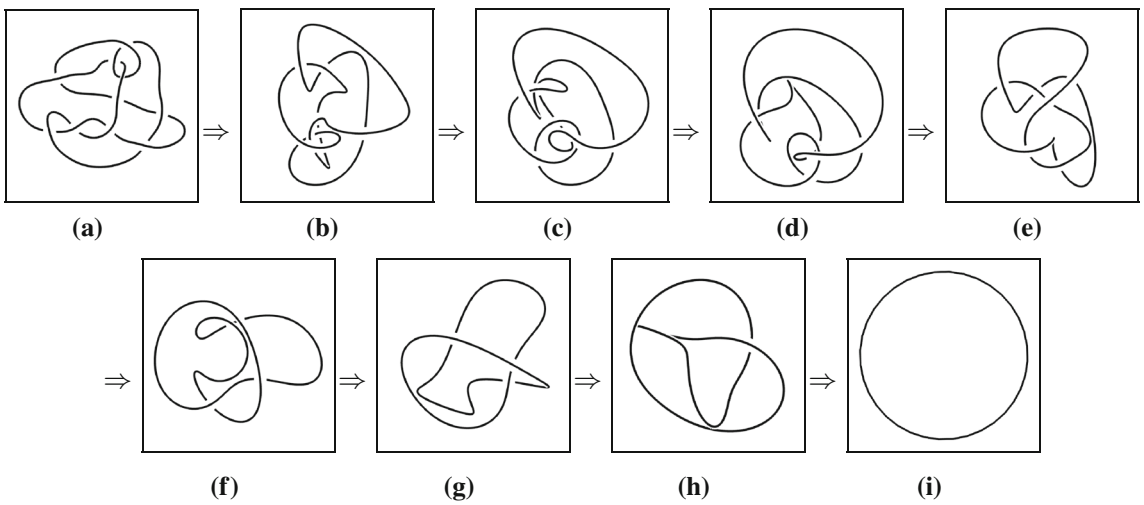

Fig. 5 Improved visual experience by aligning critical changes presented in Fig. 4

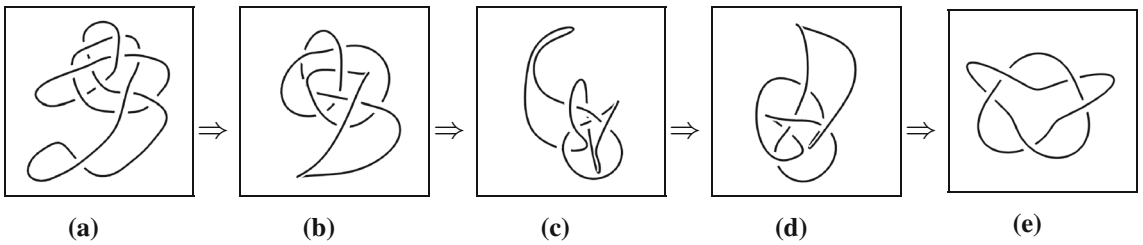

Fig. 6 Critical changes captured from the deformation of $Knot_5^1$, with visual discontinuities in successive frames. **a** Initial conformation with $crossNum = 12$. **b** $crossNum = 9$. **c** $crossNum = 7$. **d** $crossNum = 6$. **e** Final conformation with $crossNum = 5$

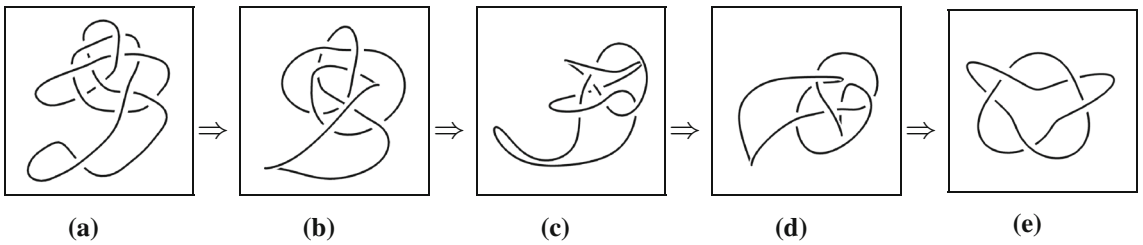

Fig. 7 Aligning critical changes to reconstruct the key moments in the original deformation of $Knot_5^1$. The visual experience is improved from Fig. 6

5 Parallelizing knot deformation and key moment extraction

Thus far, we have proposed the basic framework that makes use of topological relaxation algorithm and key moment extraction to guide and understand knot deformation in the configuration space. Both topological relaxation and key moment extraction are compute-intensive and time-consuming. In this section, we first exploit parallelization to accelerate the computation needed for knot deformation and key moment extraction. We have chosen OpenMP (2020) as the parallel programming platform. Combined with the single program multiple data (SPMD) processing model, OpenMP allows a straightforward conversion from a serial process into a multi-threaded parallel manner, which is appropriate for accelerating the visual communication of the knot deformations in our principal cases.

5.1 Accelerating the deformation

We now focus on the function units in Algorithm 2 for potential parallelization. In Algorithm 2, each iteration consists of five main steps: (1) calculating the two forces for each vertex (line 2–5), (2) finding the maximum magnitude f of all forces (line 6), (3) updating node position based on the calculated f (line 7–9), (4) collision avoidance (line 10–12), and (5) key moment extraction (line 13). Among these five steps, the computation needed for generating the deformation can be accelerated. For example, the force calculation and the position update for each vertex are independent of those for another vertex, and we therefore can parallelize the computing needed for force calculation and position update. Since the maximum magnitude calculation only involves a magnitude comparison, a linear execution should be very efficient. Collision avoidance is a very critical step: Each position update may introduce a new collision; thus, the state of each vertex depends on all related elements, so collision avoidance and position renewal require one sequential process and cannot be parallelized. In Fig. 8, we use a flowchart to show the function units in our parallel relaxation execution. Within this framework, all data points are evenly distributed to the slave threads. Each thread calculates the attractive force, the repulsive force, and the aggregated force for each single vertex. After the calculation of the maximum magnitude of all forces through the master thread, each vertex's position is updated by the slave threads.

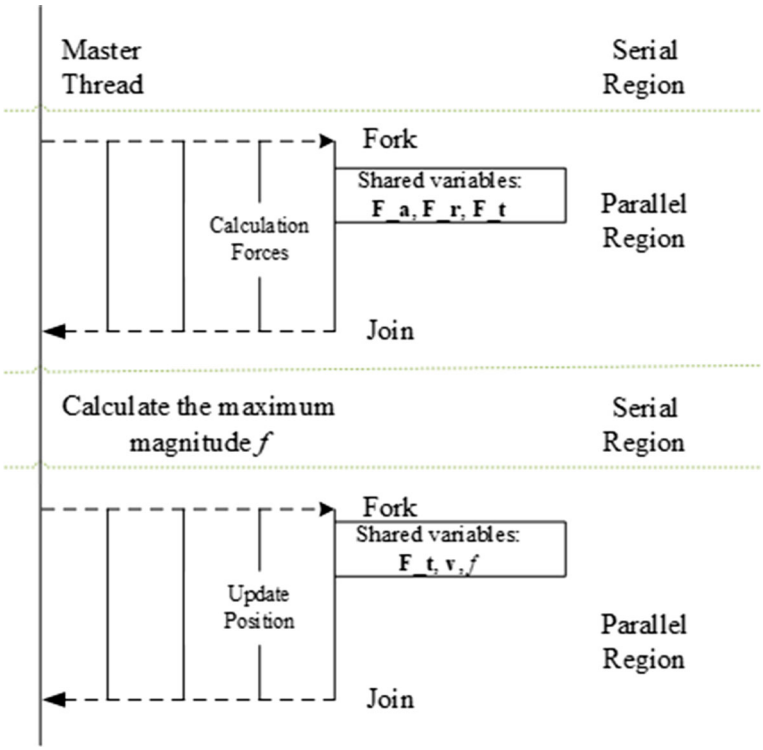

Fig. 8 Functional units in our parallelized topological relaxation

5.2 Accelerating the key moment extraction

Our basic implementation of key moments extraction (presented in Sect. 4) was a brute-force searching algorithm, which during the entire deformation has to examine every single possible diagram of the evolving knot in order to find out the one diagram with the minimal number of crossings at each time point. Such brute-force searching suffers from very heavy computational cost, at each time point, because it requires the knot to be rotated at many angles in 3D space just to find out the knot's crossing number (i.e., the invariant). Since each such operation (rotation and calculation of crossings) is dependent to others, we can again parallelize the key moment extraction with a multi-threaded parallel framework. In our implementation, two lists named *LV* and *LP* are used to store each projection value and projection diagram, respectively. The projection calculation is included within the parallel computation, and the projection value selection is outside the process. This is because we are only interested in those projections with the maximum projection value and compare them with others to identify the one with the minimal distance. The number of these projections of interest is small, and the operation is lightweight. After generating the projection values, a linear scan is used to yield a better performance.

In Algorithm 3, we describe the above-mentioned process. We generate a list of projections in a parallel way at the beginning. After that, the distances between the last frame and the diagram projected by current viewpoint are compared, and the resultant viewpoint is the one with the highest entropy value and lowest distance value. Figure 9 shows the process for $Knot_6^3$. We first rotate the knot to an arbitrary angle. Then, a serial of projection values are examined in parallel and the best projection is generated as the resultant diagram.

Algorithm 3: Key moments extraction in parallel.

Input: Initial Layout
Output: Best projection presentation
1 Initialize a empty projection list LP, projection value list LV, minimum distance d_{min} ;
2 Calculate all projection value and store the value and the structure in LV and LP respectively in parallel. Sort the LV to find projections owns same max projection value;
3 **for** each same projection $\mathbf{P}$ in LP **do**
4 **if** $d^{'} < d_{min}$ **then**
5 Update $\mathbf{P_{max}}$ with $\mathbf{P}$;
6 Update d_{min} with d ;
7 **end**
8 **end**

Our implementation is based on OpenGL and Windows Visual Studio C++. The algorithms run on a Lenovo PC desktop with Intel 4-Core i7 CPU 1.8 GHz. As mentioned above, we adopt OpenMP as the parallelizing implementation. We use speed-up= T_s/T_p as our performance metric, where T_s is the execution time of the serial algorithm and T_p is the execution time of parallel execution. To validate the performance improvement in our proposed parallel algorithm, we test a set of knots with different numbers of vertices ranging from 46 to 1000. (We note that knots of our interest in real applications have less than 200 vertices.) Figure 10 shows the speed-ups of relaxation and projection search for different numbers of vertices. As is shown in the figure, both relaxation process and projection search are able to get around 3x speed-ups in most scenarios in the execution supported by a 4-core processor. Overall, with parallelization, both relaxation and projection search are processed in a faster way.

6 Communicating knot deformation with real-time key moment extraction

Although a parallel execution in the above section can accelerate knot deformation and key moments extraction, the projection search itself is still compute-intensive and thus very time-consuming and nearly impossible to be helpful in our real-time mathematical knot interface. In this section, we will focus on a even more efficient approach to extracting key moments from long mathematical deformations. We propose an adaptive view selection way to accelerate our core computation components. Our fundamental techniques

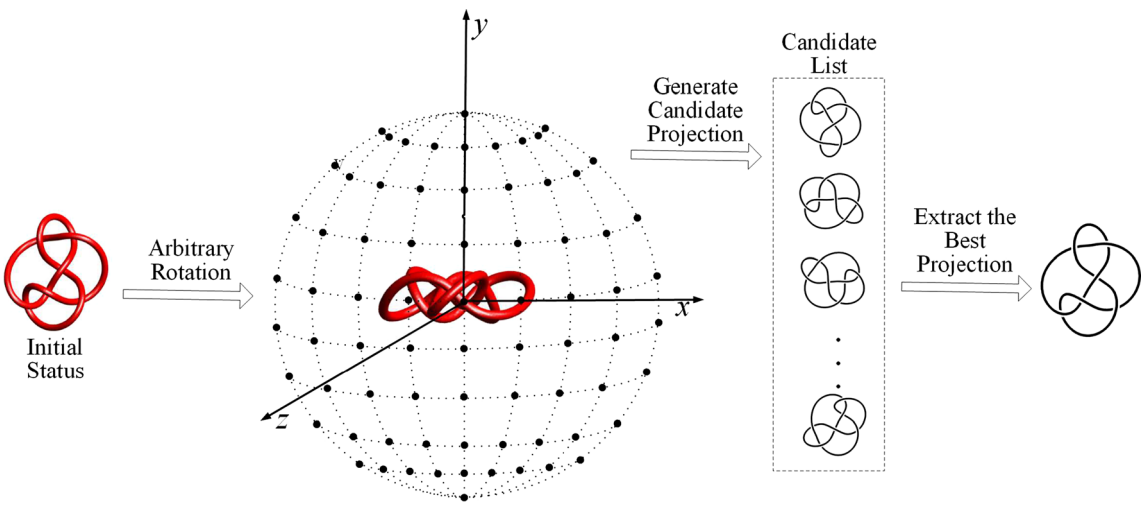

Fig. 9 Projection values generated by the parallel brute-force method. The knot incrementally rotates 1 degree along x, y, and z axis, respectively, and examined in a parallel way

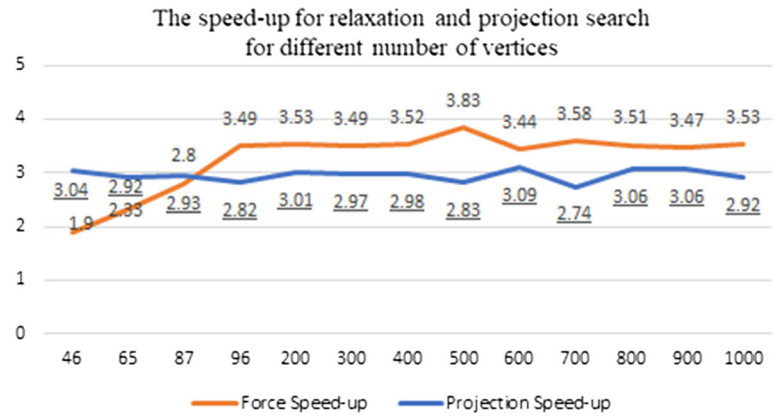

Fig. 10 Speed-ups of relaxation and projection search for different numbers of vertices

are based on a wide variety of prior arts, including Vázquez's the adaptive method to compute best views to improve view selection performance by orders of magnitude (Vázquez and Sbert 2003), Colin's best viewpoint selection with octree for performance improvement (Colin 1988), Lee's center-surround operation on Gaussian-weighted mean curvatures to capture the appealing regions automatically (Lee et al. 2005), etc. Inspired by their view selection criterion, we design our distinct adaptive view selection method in real time. The main different between Vázquez's method (Vázquez and Sbert 2003) and our work is that we sample the initial viewpoints on a regular icosahedron and narrow down the scale adaptively to ensure more uniform search.

6.1 Identifying critical changes by adaptive view selection

The performance bottleneck in our original method lies in the function to identify critical changes at each time point, which is very compute-intensive because it calculates the knot's crossing number from all possible rotation angles in 3D. It is possible to make the selection much faster. For example, instead of rotating the knot, we can examine the viewpoint entropy values from multiple viewpoints. A number of projection entropy values are generated initially; if these values can be compared and used to find the most promising direction for the next round of searching, our method can be more efficient compared to the original brute-force one. In this approach, our algorithm only focuses on the most promising searching area throughout the entire process, by adaptively narrowing down to the best views using a coarse-to-fine strategy.

Our new algorithm starts with a coarse-grained space to calculate the viewpoint entropy value at each sampling vertex, and a fine-grained local search follows to find the best viewpoint recursively until a final optimal viewpoint is reached. The main steps include:

1. Generate an initial set of viewpoints;
2. Evaluate each viewpoint and sort the viewpoint set sort in descending order and generate an initial triangular mesh with the top three best positions;
3. Evaluate the middle points of each edge and generate a new triangle;
4. Apply last step (3) recursively until the terminal criteria is satisfied.

6.1.1 Initial viewpoints generation

There are several viewpoint sampling methods, including longitude and latitude sampling method (Liu et al. 2012), random sampling method (Johan et al. 2011), and pseudo uniform method (Cao et al. 2010). In order to cover an effective searching area with a reasonable sampling distance, we start by sampling a set of points on the surface of the sphere based on the subdivision of a regular icosahedron (see, e.g., Figure 11(a)). This first-level subdivision will produce 12 sampling points around the sphere, which form our elementary viewpoints. By applying the subdivision rule recursively, we are able to narrow down the searching space gradually.

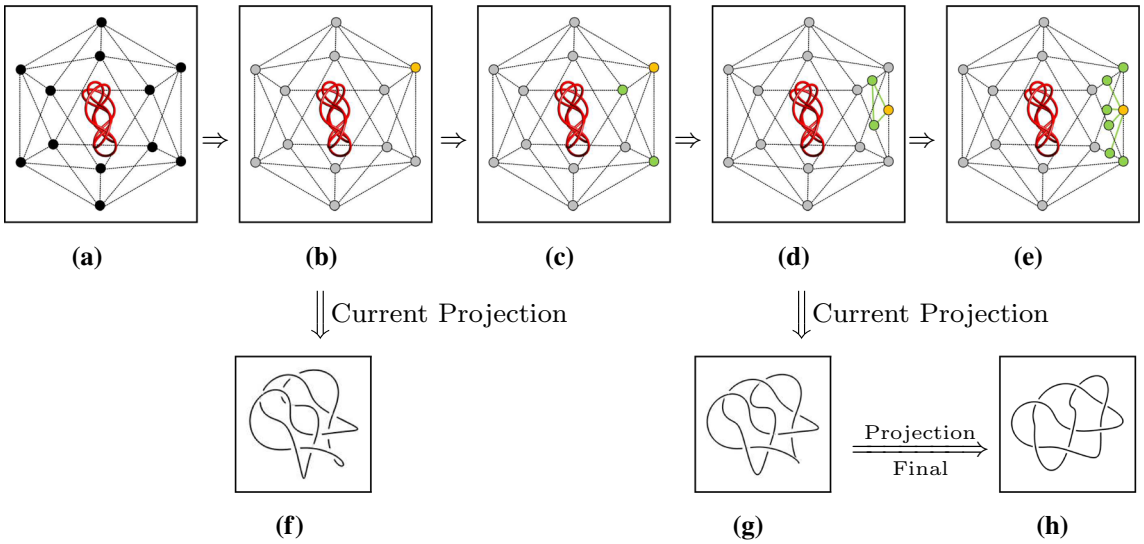

Fig. 11 Adaptive viewpoint selection. **a** The initial 12 sampling viewpoints on the icosahedron. **b** The current best viewpoint. **c** The first viewing triangle around the current best viewpoint. **d** The first candidate viewpoints generated by midpoints from edges. **e** The recursive generation of viewpoints. **f** The current projection presented by the viewpoint in **b**. **g** The current projection presented by the viewpoint in **d**. **h** The final best projection

6.1.2 Initial viewpoints evaluation

After obtaining the initial candidate viewpoints, it is essential to evaluate these viewpoints and find the next "good" starting point. Our proposed algorithm does not cover a full searching space; however, if the initial searching point fails to provide the correct searching direction, the algorithm will correct it in the following iterations. We next evaluate and compare all the viewpoints using Eq. 3 and sort the viewpoints by comparing their entropy values in descending order. If two viewpoints are associated with identical entropy values, the comparison will continue to use their minimum distances and will generate an initial triangular mesh with the three best viewpoints (i.e., with largest entropy value and minimum distance) from the last diagram (see, e.g., Fig. 11b, f).

6.1.3 Adaptive viewpoints selection

Next we use the three mid-points at current view's triangle edges to generate next three viewpoints to continue viewpoint search (see Fig. 11c). By calculating the viewpoint entropy values for the three views, we now choose the new best view (see the mid-points colored in yellow in Fig. 11d, g shows the corresponding projection associated with this viewpoint), and this allows us to identify the next six adjacent views (see the six colored in green in Fig. 11e). Our view search procedure is then recursively performed, leading to a finer searching area in each iteration, until the terminating condition is satisfied (see Fig. 11h).

The description of this procedure is sketched in the following Algorithm 4. After the initial configuration step, the process is executed iteratively until no higher projection value can be found or sampling points have reached a threshold distance. With the adaptive viewpoint method defined above, the best diagram can be identified in an efficient way.

Algorithm 4: Best diagram identification with adaptive view selection

Input: Initial Layout
Output: Best Diagram with Maximal View Entropy Value
Initialize a set of viewpoints placed on icosahedron vertices around the object;
while un-visited viewpoints exist **do**
  find the best viewpoint with maximum entropy value (maximal projective length and minimal number of crossing);
  while the entropy value of current best view is better **do**
    Find all adjacent points, calculate all entropy values on the mid-point of all edges;
    Find the next viewpoint with the maximum entropy value and minimal number of crossing;
    Update new entropy value and un-visited viewpoints;
  end
end

6.2 Reconstructing key moments from identified best views

Compared to the brute-force best view searching strategy, the adaptive view selection method covers a much smaller number of viewpoints and can still obtain an approximate optimal projection value globally. Since the search does not cover the complete searching space, we cannot use the same comparison strategy to select the one that is best aligned to the original diagram. In order to provide the desired visual continuity, we apply a rigid transformation to "re-align" between the diagrams in the final visualization sequence.

We use a homogenous transform (Cashbaugh and Kitts 2018) method to align two resultant diagrams. Given two diagrams $\mathbf{P}_A$ and $\mathbf{P}_B$, a homogeneous transformation can be provided with an augmented transformation matrix that contains the rotation and translation between two the coordinates of two projections (see Eq. 5).

$$\begin{bmatrix} P_B \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} R & T \\ \mathbf{0} & \mathbf{1} \end{bmatrix} * \begin{bmatrix} P_A \\ \mathbf{1} \end{bmatrix} \tag{5}$$

where $R = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{bmatrix}$, $T = [t_x, t_y, t_z]^t$.

In the transformation matrix, R represents the rotations and T is the translation between the two frames. The second row is an orthonormal perspective and homogeneous scaling factor.

With a linear regression to minimize the square of the residual, a volume matrix A is obtained (Eq. 6) to generate each column of the transformation matrix (Eqs. 7–9).

$$A = \begin{bmatrix} \sum(x_{Ai}^2) & \sum(x_{Ai}y_{Ai}) & \sum(x_{Ai}z_{Ai}) & \sum(x_{Ai}) \\ \sum(x_{Ai}y_{Ai}) & \sum(y_{Ai}^2) & \sum(y_{Ai}z_{Ai}) & \sum(y_{Ai}) \\ \sum(x_{Ai}z_{Ai}) & \sum(y_{Ai}z_{Ai}) & \sum(z_{Ai}^2) & \sum(z_{Ai}) \\ \sum(x_{Ai}) & \sum(y_{Ai}) & \sum(z_{Ai}) & n \end{bmatrix} \tag{6}$$

where $i \in [1, n]$.

$$\begin{bmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \\ t_x \end{bmatrix} = [A]^{-1} \begin{bmatrix} \sum(x_{Bi}x_{Ai}) \\ \sum(x_{Bi}y_{Ai}) \\ \sum(x_{Bi}z_{Ai}) \\ \sum(x_{Bi}) \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \\ t_y \end{bmatrix} = [A]^{-1} \begin{bmatrix} \sum(y_{Bi}x_{Ai}) \\ \sum(y_{Bi}y_{Ai}) \\ \sum(y_{Bi}z_{Ai}) \\ \sum(y_{Bi}) \end{bmatrix} \tag{8}$$

$$\begin{bmatrix} r_{zx} \\ r_{zy} \\ r_{zz} \\ t_z \end{bmatrix} = [A]^{-1} \begin{bmatrix} \sum(z_{Bi}x_{Ai}) \\ \sum(z_{Bi}y_{Ai}) \\ \sum(z_{Bi}z_{Ai}) \\ \sum(z_{Bi}) \end{bmatrix} \tag{9}$$

We note that the best projections are all generated in xy plane. Since all data points are in the same plane, A is not invertible. Here, the pseudo-inverse is used to replace the real inverse matrix. A general solution called the Moore–Penrose method is used (Penrose 1955) and an approximate inverse is generated through SVD procedure. When the valid transformation is acquired, an accurate transformation matrix can further produce the rotation angle. More derivation and proof can be found in Cashbaugh and Kitts (2018).

With the best matching rotation angle, we find the optimal match between the current key moment and last key frame and then rotate the current key moment in xy plane to approach the last key moment. In this way, the sequence of visually continuous frames can provide a summary of the long knot deformation. Figure 13 shows the improved key moment extraction where the least interruptions are introduced in each of the snapshots with visual continuity maintained across snapshots. Figures 14 and 15 give the similar results for $Knot_5^1$.

6.3 Performance evaluation

In order to validate performance improvement, we have tested the same set of curves and make the comparison between brute-force and the adaptive view selection method. Table 1 shows our results: The searching space for the adaptive method is far less than the brute force; thus, it significantly reduces the time complexity of the algorithm with significant speed-ups. We also observe that knot diagram with a large number of vertices will gain better speed-ups with an adaptive view selection method, which means the method will be particularly beneficial for large and complicated diagrams.

To further benchmark our algorithms, we introduce a complex knotted curve (which is an unknot if untangled) and execute a complete topological relaxation with both parallel brute-force algorithm and adaptive view selection method. The resultant key frames are presented in Figs. 16 and 17, respectively. The result in these two figures shows that the adaptive view selection method only needs to examine a much smaller number of viewpoints, while obtaining the same visual effects comparable with those from the brute-force method and dramatically reducing the processing time to nearly real-time fashion.

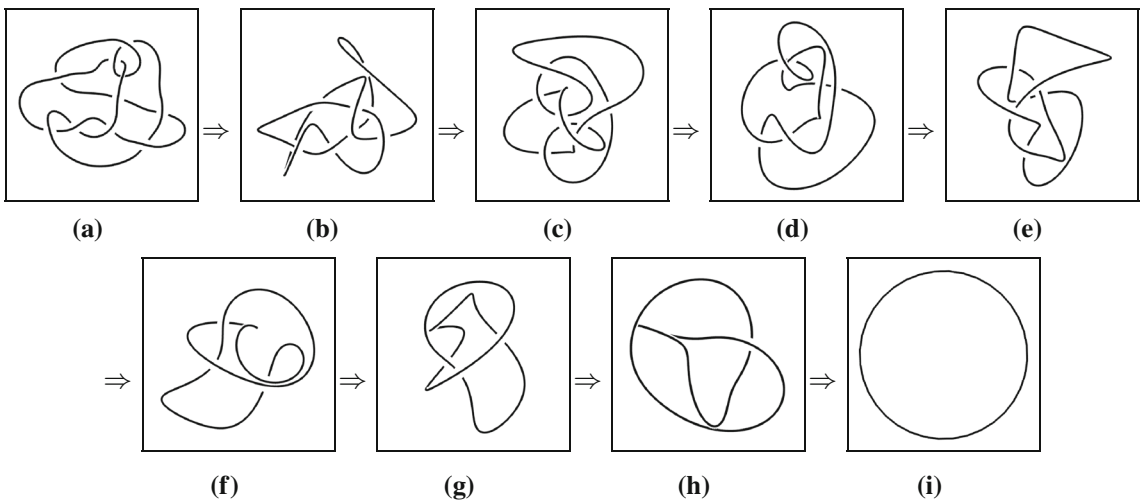

Fig. 12 Critical changes extracted from the "monster" unknot's deformation, using the adaptive viewpoint selection method

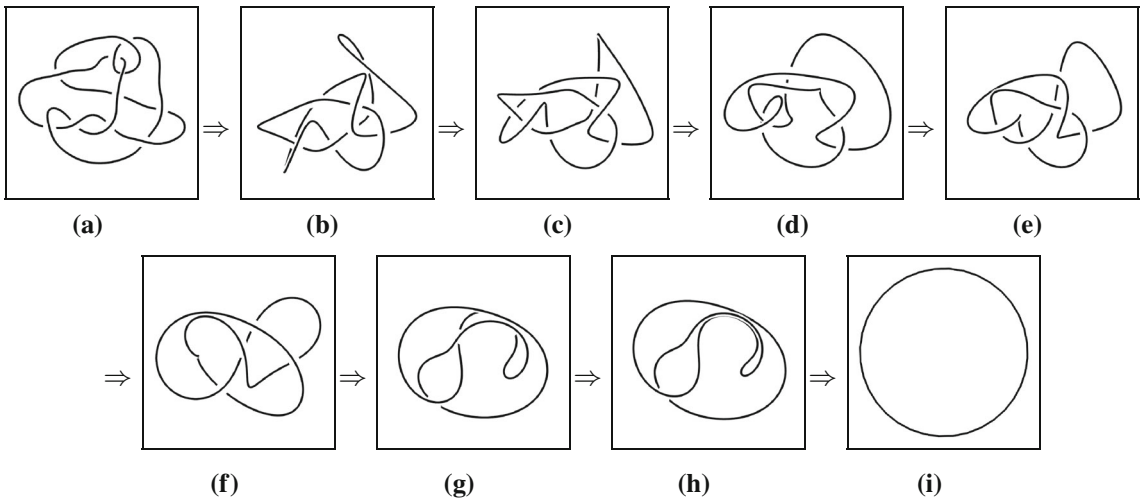

Fig. 13 Critical changes extracted and aligned from the "monster" unknot's deformation, compared to Fig. 12

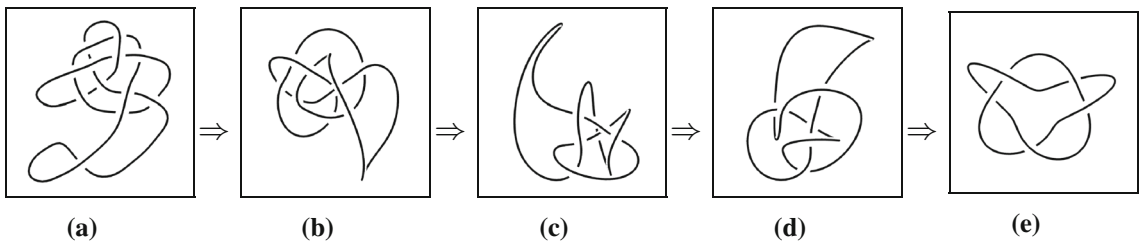

Fig. 14 Relaxation for $Knot_5^1$ with adaptive best projection presentation

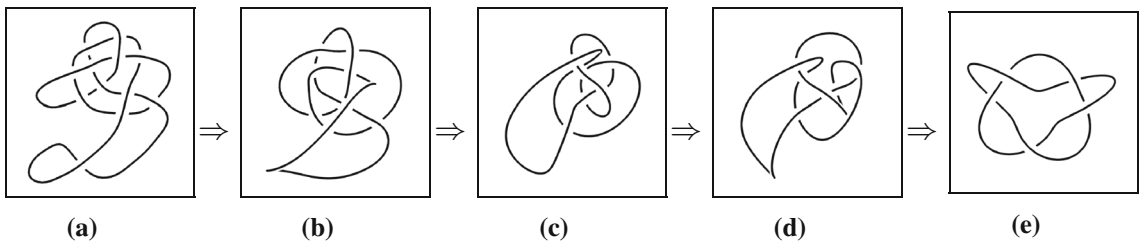

Fig. 15 Relaxation for $Knot_5^1$ with improved aligning adaptive best projection

Table 1 Performance comparison between parallel brute-force and the adaptive view selection method

Vertex number	Parallel brute-force method execute time (s)	Adaptive method execute time (s)	Speed-up
46	6.031	0.219	27.54
65	10.391	0.25	41.56
87	16.969	0.766	22.15
96	21.031	0.578	36.39
200	68.641	2.078	33.03
300	145.688	3.5	41.63
400	253.063	6.032	41.95
500	427.187	9.281	46.03
600	603.922	13.422	44.99
700	804.61	19.719	40.8
800	1070	22.453	47.66
900	1370	29.281	46.79
1000	1680	36.703	45.77

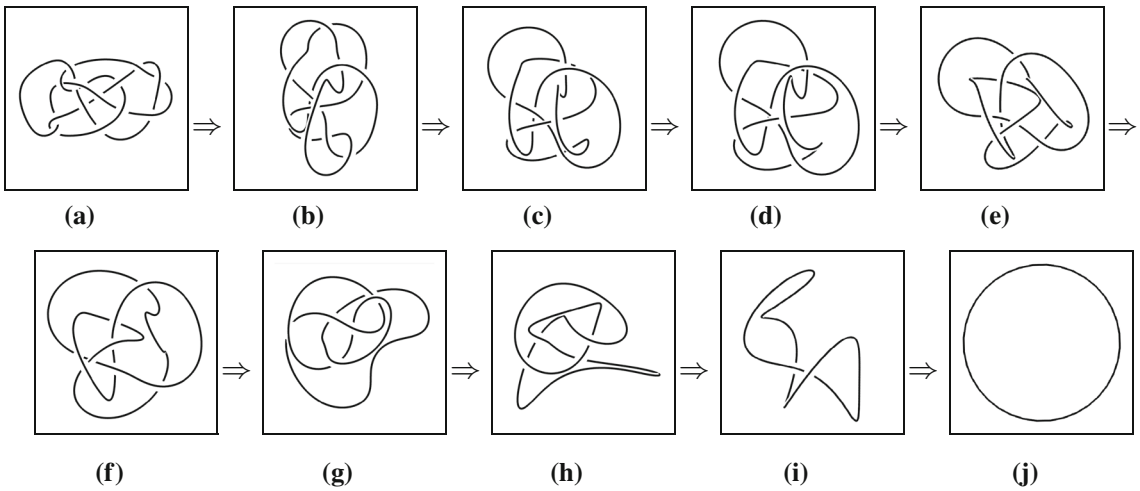

Fig. 16 Relaxation for complex unknot with adaptive best projection presentation

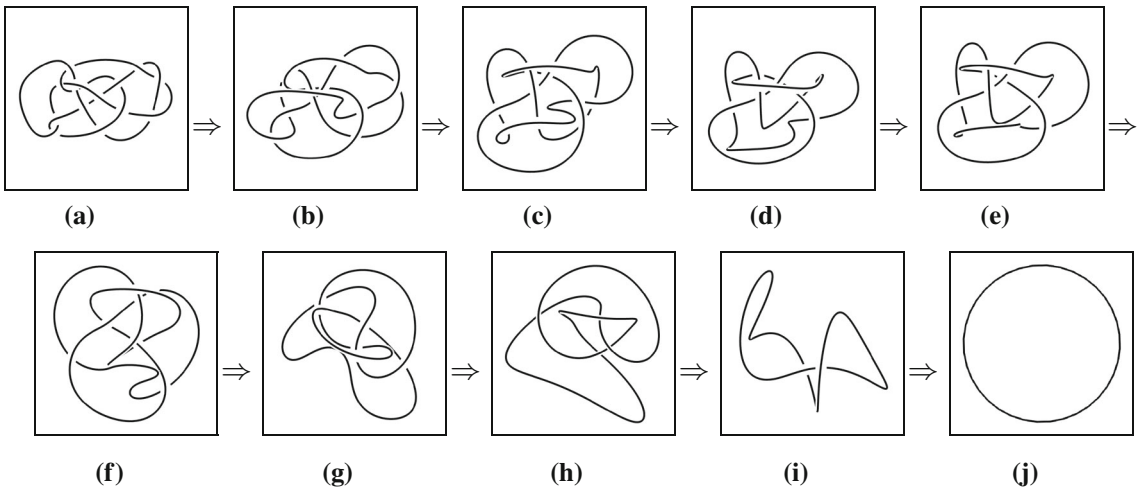

Fig. 17 Relaxation for complex unknot with improved aligning adaptive best projection presentation

7 Conclusions and future work

Our ultimate goal is to facilitate our understanding of mathematical knots and their topological refinement. In order to achieve this goal, we have proposed a family of interactive methods and their parallelized counterparts to simulate and visualize mathematical knot deformation with an energy-based model. To extract the critical changes and the moments associated with these changes from the long sequence of deformation, we have proposed an innovative method to computationally identify the critical changes that has occurred in knot deformation and capture them to visually communicate the entire deformation.

In this work, we use energy model-driven computational simulations to untangle mathematical knots in three-dimensional space. This method is different from the mathematical way of untangling knots, i.e., studying knots with the Reidemeister moves (Trace 1983). While our interface is easy to understand and requires minimal knot expertise for one to interact with, the underlying simulation-based method cannot guarantee that only one Reidemeister move occurs between two successive key moments extracted by our interface.

Our future direction includes integrating with high performance computing in the back-end to accelerate the mathematical knot deformation and a user interface for end users to easily interact with high-performance computing resources. Beginning with this primary framework, we plan to extend to attack more

complicated mathematical entities such as surfaces and manifolds embedded in -dimensional space to understand and visualize their deformation in a more effective parallelizing means.

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

Adams CC (2004) The knot book: an elementary introduction to the mathematical theory of knots. American Mathematical Society, Providence
Cao W, Hu P, Li H, Lin Z (2010) Canonical viewpoint selection based on distance-histogram. J Comput Aided Des Comput Gr 22(9):1515–1521
Carlen M (2010) Computation and visualization of ideal knot shapes. Tech. rep, EPFL
Cashbaugh J, Kitts C (2018) Automatic calculation of a transformation matrix between two frames. IEEE Access 6:9614–9622
Chen Y, Guan Z, Zhang R, Du X, Wang Y (2019) A survey on visualization approaches for exploring association relationships in graph data. J Vis 22(3):625–639
Colin C (1988) A system for exploring the universe of polyhedral shapes. In: Eurographics 88
Costagliola G, De Rosa M, Fish A, Fuccella V, Saleh R, Swartwood S (2016) Knotsketch: a tool for knot diagram sketching, encoding and re-generation. J Vis Lang Sentient Syst 2:16–25
Eades P (1984) A heuristic for graph drawing. Congr Numer 42:149–160
Erten C, Harding PJ, Kobourov SG, Wampler K, Yee G (2004) Exploring the computing literature using temporal graph visualization. In: Visualization and data analysis 2004, vol 5295. International Society for Optics and Photonics, pp 45–56
Garey MR, Johnson DS (1983) Crossing number is np-complete. SIAM J Algebraic Discrete Methods 4(3):312–316
Harel D, Koren Y (2000) A fast multi-scale method for drawing large graphs. In: International symposium on graph drawing. Springer, pp 183–196
Johan H, Li B, Wei Y et al (2011) 3d model alignment based on minimum projection area. Vis Comput 27(6–8):565
Kobourov SG (2012) Spring embedders and force directed graph drawing algorithms. arXiv preprint arXiv:1201.3011
Lee CH, Varshney A, Jacobs DW (2005) Mesh saliency. In: ACM transactions on graphics (TOG), vol. 24. ACM, pp 659–666
Lin J, Hui Z (2019) Visualizing mathematical knot equivalence. In: IS&T international symposium on electronic imaging 2019, Visualization and Data Analysis 2019 proceedings, VDA 2019. Society for Imaging Science and Technology
Lin J, Zhang H (2019) Visually communicating mathematical knot deformation. In: Proceedings of the 12th international symposium on visual information communication and interaction. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3356422.3356438
Liu L, Silver D, Bemis K (2019) Visualizing events in time-varying scientific data. J Vis 5:1–16
Liu YJ, Fu QF, Liu Y, Fu XL (2012) 2d-line-drawing-based 3d object recognition. In: International conference on computational visual media. Springer, pp 146–153
Openmp. http://www.openmp.org/. Accessed Jan 7, 2020
Pach J, Tardos G (2002) Untangling a polygon. In: Mutzel P, Jünger M, Leipert S (eds) Graph drawing. Springer, Berlin, pp 154–161
Penrose R (1955) A generalized inverse for matrices. In: Mathematical proceedings of the Cambridge philosophical society, vol. 51. Cambridge University Press, pp 406–413
Schaefer M (2013) The graph crossing number and its variants: a survey. Electron J Combin 1000:21–22
Scharein RG (1998) Interactive topological drawing. Ph.D. thesis, Citeseer
Simon JK (1994) Energy functions for polygonal knots. J Knot Theory Ramif 3(03):299–320
Snibbe S, Anderson S, Verplank B (1998) Springs and constraints for 3d drawing. In: Proceedings of the third phantom users group workshop
Trace B (1983) On the reidemeister moves of a classical knot. In: Proceedings of the American Mathematical Society, pp 722–724
Vázquez PP, Feixas M, Sbert M, Heidrich W (2001) Viewpoint selection using viewpoint entropy. VMV 1:273–280
Vázquez PP, Sbert M (2003) Fast adaptive selection of best views. In: International conference on computational science and its applications. Springer, pp 295–305
Wu Y (1996) An md knot energy minimizing program. Department of Mathematics, University of Iowa
Zhang H, Thakur S, Hanson AJ (2007) Haptic exploration of mathematical knots. In: International symposium on visual computing. Springer, pp 745–756
Zhang H, Weng J, Jing L, Zhong Y (2012) Knotpad: visualizing and exploring knot theory with fluid reidemeister moves. IEEE Trans Vis Comput Gr 18(12):2051–2060
Zhang H, Weng J, Ruan G (2014) Visualizing 2-dimensional manifolds with curve handles in 4d. IEEE Trans Vis Comput Gr 1:1–1
Zhang H, Zhong Y, Jiang J (2016) Visualizing knots and braids with touchable 3d manipulatives. In: 2016 IEEE Pacific visualization symposium (PacificVis), pp 24–31. https://doi.org/10.1109/PACIFICVIS.2016.7465247