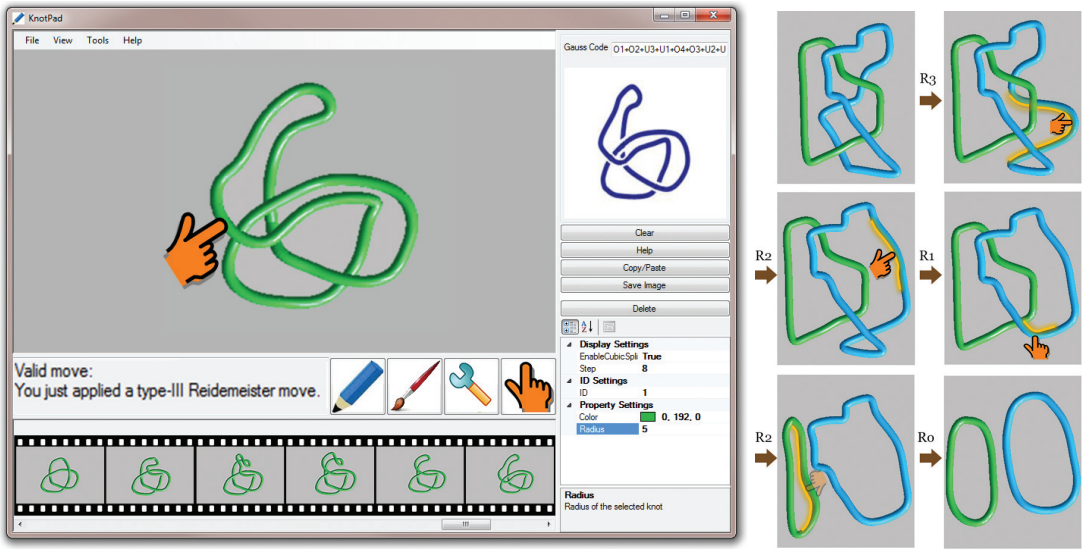
KnotPad: Visualizing and Exploring Knot Theory with Fluid Reidemeister Moves

Hui Zhang, Jianguang Weng, Lin Jing, and Yiwen Zhong



(a) (b)

Fig. 1. (a) The supported manipulations in KnotPad are based on the three types of Reidemeister moves. Our tool allows users to produce and trace equivalent knot diagrams through verifying and storying the mathematically valid moves. (b) Screen images of applying the Reidemeister moves to the mathematical link diagrams.

Abstract—We present KnotPad, an interactive paper-like system for visualizing and exploring mathematical knots; we exploit topological drawing and math-aware deformation methods in particular to enable and enrich our interactions with knot diagrams. Whereas most previous efforts typically employ physically based modeling to simulate the 3D dynamics of knots and ropes, our tool offers a Reidemeister move based interactive environment that is much closer to the topological problems being solved in knot theory, yet without interfering with the traditional advantages of paper-based analysis and manipulation of knot diagrams. Drawing knot diagrams with many crossings and producing their equivalent is quite challenging and error-prone. KnotPad can restrict user manipulations to the three types of Reidemeister moves, resulting in a more fluid yet mathematically correct user experience with knots. For our principal test case of mathematical knots, KnotPad permits us to draw and edit their diagrams empowered by a family of interactive techniques. Furthermore, we exploit supplementary interface elements to enrich the user experiences. For example, KnotPad allows one to pull and drag on knot diagrams to produce mathematically valid moves. Navigation enhancements in KnotPad provide still further improvement: by remembering and displaying the sequence of valid moves applied during the entire interaction, KnotPad allows a much cleaner exploratory interface for the user to analyze and study knot equivalence. All these methods combine to reveal the complex spatial relationships of knot diagrams with a mathematically true and rich user experience.

Index Terms—Knot Theory, Math Visualization.

1 Introduction

Mathematics and physics are two closely related academic disciplines [10]. For example, studying knot theory, namely the three types of Reidemeister moves, lies at the crossroads of mathematics and physics. In knot theory, one considers two knots are the same if one can distort one knot into the other without breaking it, much as we physically play with a real knotted rope where one part is allowed to cross but not intersect with another. Such equivalence relation is mathematically generated on diagrams and related by a sequence of three Reidemeister moves [13]. As illustrated in Figure 2, a Reidemeister move refers to one of three local moves on a knot diagram. Each move is performed on a local part of the diagram that is topologically identical to the original part of the diagram illustrated in the figure without changing the rest of the diagram. These three Reidemeister moves are representative and useful in doing combinatorial topology in mathematical knots and links. By applying the Reidemeister moves we can gain insights of the underlying knot topological structure by obtaining a continuous family of embeddings of circles in three-space from one knot to its

- *Hui Zhang is with Pervasive Technology Institute at Indiana University, e-mail: huizhang@iu.edu.*
- *Jianguang Weng is with Zhejiang University of Media and Communications, China.*
- *Lin Jing and Yiwen Zhong are with Fujian Agriculture and Forestry University, China.*

Manuscript received 31 March 2012; accepted 1 August 2012; posted online 14 October 2012; mailed on 5 October 2012.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

equivalent.

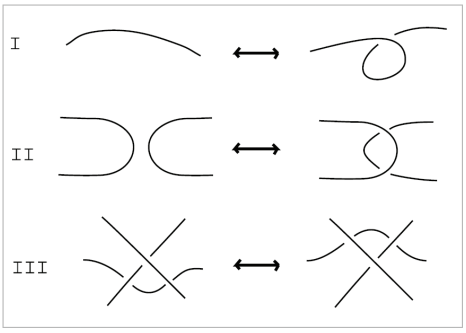


Fig. 2. The Reidemeister moves. Each move operates on a small region of the diagram and is one of three types: (I) Twist and untwist in either direction; (II) Move one loop completely over another; (III) Move a string completely over or under a crossing.

Due to the complexity of knot diagrams and the complex interactions required, it has been very challenging to design a successful interactive system for fluid entry and editing of knot diagrams with support for the Reidemeister moves. Building such an interface is very different from other conventional user interface design, and important unique features for this class of math visualization problems include:

- **Precise 3D depth is relatively unimportant.** Different from many other conventional visualization systems where one typically has to switch to a set of views for precise depth control, when manipulating knot diagrams only the relative depth ordering (i.e., over/under- crossings) is of interest. For example, the type II Reidemeister move (see e.g., Figure 2(II)) allows one to move one loop either completely over or under another. In such cases, the initial precise 3D depth of the two loops are unimportant as if they were totally flattened and the relative depth order can be created when they are slided through each other. Topological rules indeed allow us to generate such moves that are mathematically true but beyond simple geometric manipulations.

- **3D continuous structure, $2\frac{1}{2}$D representation, and 2D interaction.** While mathematical knots are smooth curves embedded in 3D, classic "knot-crossing diagrams" are commonplace in textbooks. A knot diagram is a planar projection of the 3D knot together with some extra annotations to encode 3D structure (the "$2\frac{1}{2}$D" method). Studying knot theory is concerned with applying connected Reidemeister moves (i.e., 2D interaction) to transform one knot diagram into another, in which the user usually favors constant visual connection with the diagrams throughout the entire interaction with step-by-step manipulations.

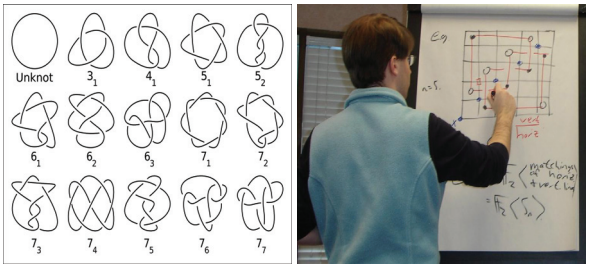
- **Non-strictly physical.** Although manipulation of knot diagrams appears to be as much physical as that of real ropes in our everyday life. Tasks of applying types of Reidemeister moves to knot diagrams are indeed beyond 3D physics. For example, a twist/untwist Reidemeister move typically turns a segment in a knot diagram into a crossing without changing the rest of the diagram. Such a "move" is (usually) not length preserving (see Figure 2(I)), which is beyond a 3D rope's physical dynamics but totally plausible in the mathematical world.

- **Error-prone interactions.** Manipulating knot diagrams with the Reidemeister moves means a range of tedious and error prone interactions, very much like one manually performs steps in a derivation. Even a task of untangling a topologically unknotted curve might not be as easy as it sounds: the upper bound on the number of Reidemeister moves required to remove all the crossings from a knot diagram of n crossings proved to be $2^{100,000,000,000 \times n}$ [9]. Furthermore, when making a Reidemeister move to generate a new knot diagram with n crossings, there are 2^n distributions of markers over all crossings and one false marker of a crossing will result in a mathematically wrong move. An essential design issue is therefore to build an interactive, friendly, "error-aware" system: such a system should recognize and only allow mathematically valid moves being applied to knot diagrams.

Our task in this paper is to present such a sketching based knot diagram interface where one can fully interact with 3D mathematical knots through step-by-step 2D Reidemeister moves. We first start from extending the familiar "knot drawing" idea to a pseudo-haptic topological drawing interface with which one can draw knot diagrams by taking advantage of a "physically" reactive mouse cursor. We then proceed to describe how the pseudo-haptic approach can be extended and used to edit the knot diagram by sketching the Reidemeister moves to obtain knot equivalence. Having established the main mechanisms and interaction paradigms, we proceed to an interactive software environment that is capable of manipulating knot diagrams by allowing sketching, rendering, editing, and polishing knot diagrams. We next discuss a family of user interface elements we implemented to improve user experiences including a $2\frac{1}{2}$D rendering method with light and material that adds apparent 3D geometry, depth and shapes to the knot-crossing diagrams, a navigation interface that captures manipulation steps during the entire interaction, and a deformation based editing interface to simplify user input by allowing pulling and dragging on knot diagrams. We finally present a user experience study we conducted and our paper ends with a conclusion and a description of potential perspectives.

2 Related Work

We usually describe mathematical knots by drawing their projective images on flat 2D media such as a blackboard or a sheet of paper (see Figure 4(a)(b)). The knot diagrams (see e.g., Figure 4(a)) are regarded both as schematic pictures of the knots, and as plane graphs with extra structure at the nodes (indicating how one part of the knot cross over or under another by standard pictorial conventions). Diagrams of equivalent knots are connected by a sequence of the Reidemeister moves that happen under the planar projection of the movement taking one knot to another.



(a) (b)

Fig. 4. (a) Knot diagrams up to 7 crossings. (b) Knots and the Reidemeister moves are usually drawn on flat 2D media such as a blackboard or a sheet of paper. (Photo courtesy of Robert Lipshitz.)

Due to the complexity of the Reidemeister moves, finding an available sequence of the Reidemeister moves between two equivalent (but with different appearances) knot diagrams is often fairly challenging. It can be more difficult to compare or deform between distinct presentations of two complex knots. When studying knot theory, one often has to use an eraser to cut and edit pieces of the curve many times to find the sequence of Reidemeister moves between two knot diagrams, and one typically finds it fairly difficult to remember the sequence of moves after manipulating complex knot diagrams with many tries. Another drawback of such hand-drawn diagrams is that once created on traditional 2D media they are nearly impossible to modify, and error-check is very challenging to perform on a large sequence of deformations.

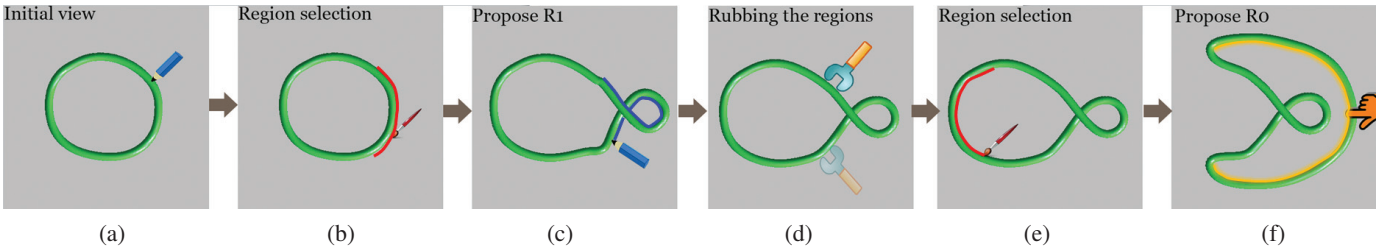


Fig. 3. (a) Sketching a knot diagram. (b) Selecting a local part of the diagram. (c) Sketching and proposing the twist Reidemeister move to replace the selected part in (b). (d) Rubbing parts of the diagram to make the appearance more smooth. (e)-(f) Generating the knot's equivalent with a dragging gesture.

The emergence of computer graphics and visualization techniques has sparked a number of interesting attempts to use interactive computer systems to help describe 3D curves and mathematical knots. For example, Scharein's Knotplot has been widely used to sketch mathematical knots (mainly done in a plane assisted by general 3D rotation) [20] , while Cohen et al. proposed to create and manipulate 3D curves by correlating the curve with its sketched shadow to compute the curve's 3D shape [5] . Wijk's SeifertView [24] visualizes the Seifert surface of knots and links for users to understand their shape and structure. Snibble et al. merge springs and constraints with the study of geometric characteristics relevant to knots (see, e.g., [21]). Others exploit haptic interfaces to edit and simulate 3D knots and ropes (see, e.g, [3, 25, 17, 23]). Wu's MING [28] is a knot drawing and refinement tool that utilizes energy based minimization models. Zhang and Hanson suggested studying mathematical knots by making and exploring their shadow pictures using haptic interfaces. Their approach showcased how haptic and pseudo-haptic exploration of projected mathematical objects (i.e., 2D knot diagrams of 3D knots or 3D projective images of 4D knotted spheres) can support full-featured physical exploration of mathematical knots in the full dimensional space [8, 31, 32, 30, 33].

However, all these efforts are essentially focusing on visualizing the physical dynamics (or pseudo-dynamics) of 3D knots and ropes (or sometimes knotted spheres in higher dimensional space), which is barely helpful for diagrammatically drawing and studying knot theory, namely, the Reidemeister moves. Although some specific knot diagram editing and knot equivalence finding tasks can be illustrated with a physical simulation of 3D knot or rope, knot theory concerns itself with isotopy that preserves shape (e.g., an isotopy can deform a circle of radius one to a circle of radius two) and thus general Reidemeister moves (which are non-physical, local deformations) can not be generated with a 3D simulated rope on a computer screen or with a real knotted rope in one's hand.

Therefore we have been motivated to develop such a math-aware interface where one can study knot theory by creating the knot diagrams, and applying step-by-step Reidemeister moves to gain insight into knot equivalence through the entire interaction, assisted by a family of computer based fluid user interface elements.

3 User Interface Overview

From the user's point, KnotPad consists of a sketching area, where the user interactively draws the knot diagram with a "physically" reactive mouse cursor and the system automatically queries the user for an explicit choice of over/under- crossing to ensure no part of the curve runs into another [33] (see Figure 3(a)). The user can push and pull part of the curve for further geometry control (in 2D), and toggle between classic knot-crossing diagram, Gauss diagram, and $2\frac{1}{2}$D smooth knot rendering view. In addition, the user can freely select a local part of the knot diagram (see Figure 3(b)) and propose the Reidemeister moves either by interactive drawing (see Figure 3(c)) or by pulling and dragging on parts of the knot diagram (see Figure 3(f)). KnotPad automatically performs the legality check of each proposed move by alerting the user when illegal editing attempts are recognized, and construct the new knot diagrams by restricting user manipulations to legal moves only. Hand-drawn curves can be jittery especially when one keeps replacing local parts of the diagrams with newly sketched components. To solve this problem, we supplement our system with a "rubbing" tool that allows users to smooth small regions of the target diagram for better-looking images (see Figure 3(d)). KnotPad can recognize and then "remember" each valid move the user performs to connect one knot diagram to another, therefor one can take advantage of such a "memory" in our system to nagivate through the sequence of the Reidemeister moves generated from the interactions.

The major contributions of this paper are summarized below:

- First, this paper proposes a knot diagram interface that provides connected math-aware manipulation. This is the first paper that explores the use of step-by-step Reidemeister move interface and graph visualization to approach knot theory visualization in the closest way to the problems being solved.
- Second, we propose a hybrid, novel interface with 2D visuals and layered 3D depth that still supports full 3D interaction. We showcase in this paper that providing users with 2D diagrammatic visuals with layered 3D depth still allows for a mathematically true and rich experience with the elegant but complex 3D knots.
- Last, we investigate and evaluate a family of novel interaction methods for reducing interaction complexity and preventing cognitive disconnect during the entire interaction. Our user study results further demonstrate the efficacy and applicability of KnotPad system.

By combining graphics, graph visualization, sketching based interface, and pseudo-haptics [20, 5, 21, 8, 32, 30, 33], we feel we can make non-trivial contribution not only to classes of geometric and topological problems with interactive math visualization approaches, but also to the larger community of researchers working with 2D interaction for the design of complex 3D shapes (e.g., working with two-dimensional floor plans to communicate 3D architectural design, and node-link diagram manipulation methods).

4 Implementation Methods

In this section, we describe the families of methods used to implement the interaction procedures and user interfaces. Our fundamental techniques are based on a wide variety of prior art, including haptic interfaces focusing on virtual realism (see, e.g., [1, 14, 15]), pseudo-haptic illusions generated by visual feedback (see, e.g., [12, 18, 33]), and other variants on sketching and exploration techniques [21, 2, 29, 11, 7, 16, 6].

4.1 Drawing Knot Diagrams

4.1.1 Pseudo-haptic Knot Creation

We use the pseudo-force model [33] to simulate a "collision-sensing" cursor. The user sketches knot diagrams with such a "collision-sensing" cursor and relative depth order (i.e., local crossings) is handled as part of the sketching task. The main idea of our pseudo-haptic knot creation approach consists in modifying the motion of the cursor displayed on the drawing panel during the manipulation of the input device by the user.

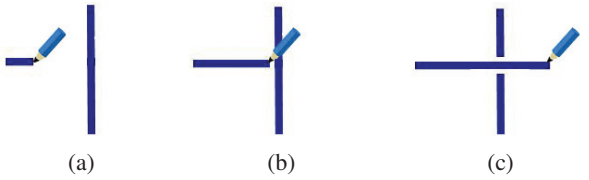


Fig. 5. Sketching part of a knot diagram with a "physically" reactive mouse cursor. (a) D/C = 1. (b) D/C = 0. (c) D/C = 1.

As shown in Figure 5, the Display/Control ratio (D/C) of the mouse cursor is adjusted during users' interaction with the knot diagrams being sketched. When not touching an existing segment, the cursor will always be placed at the mouse position (see Figure 5(a)). The system continually detects potential collisions and adjusts the mouse cursor to keep it from moving inside any existing curve segments (see Figure 5(b)). While the actual position of the mouse device may be inside an existing segment, the displayed mouse cursor will always be kept outside to give an illusion of collision sensing unless an explicit over-or-under choice is being made by the user (Figure 5(c)). As the user moves the mouse device, the path of the displayed cursor is sampled to create the knot structure.

4.1.2 Knot Representations

Although the sketching is done in 2D, all relative depth order information should be encoded, visually and structurally. The sketched knot diagram is represented with a "flat" data structure illustrated in Figure 6. Each rigid segment of the diagram consists of two 2D end-points, and an array of local crossing points with their geometric information (again, in 2D) and corresponding crossing signs. Thus, 3D mathematical knots are now transformed into an array of sketched 2D segments and encodings of relative depth order.

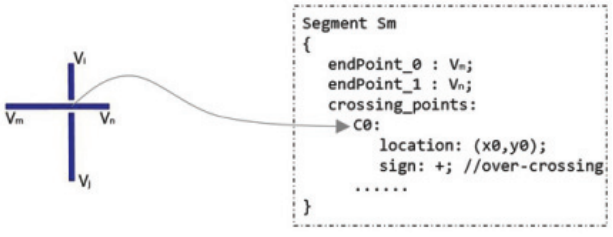


Fig. 6. 3D spatial information of mathematical knots is encoded at the crossings. Crossing points are stored in the data structure of each rigid component.

Our interface is capable to render 2*D* knot diagram, 2*D* Gauss diagram, and $2\frac{1}{2}D$ knot representations during the entire interaction with knots, to provide a visual link of analyzing and manipulating knots of the three representations through the Reidemeister moves (see Figure 7).

2D Knot Diagram. The method of rendering a 2D knot diagram is to attach a thickened curve segment in background color behind each of the curve segments that are rendered in foreground color, so that a visual break is created on each side of an under-crossing [32].

2D Gauss Diagram. The generated diagram, traverse direction, and the resolved crossings indicated by the virtual pen also provides an extended knot notation called *Gauss codes* (a Gauss code is a sequence of labels for the crossings with each label repeated twice to indicate a walk along the diagram from a given starting point and returning to that point [13].) If we render a circle and mark our Gauss code sequence counter-clockwise around the circle, connecting the two occurrence of each label with an arrow pointing toward the under-crossing (the one with "u") and decorating each arrow with a "+" or "-" for the headness of each crossing, we obatain a different knot representation called "Gauss Diagram".

$2\frac{1}{2}$D Smooth Knot. This employs a depth-buffered rendering with some embellishments to emphasize relative depth order information. By creating 3D geometry for strands that cross under others, thickening the resultant curve, providing material features combined with lighting and shading methods, we can obtain the $2\frac{1}{2}D$ knot image. Even better improvement can be achieved by computing a Cubic Spline [27] between each of the nodes along the knotted curve (see, e.g., Figure 7(b)).

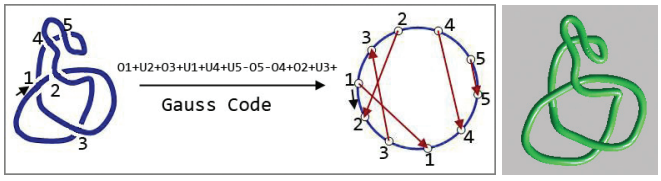


Fig. 7. (a) A sketched knot diagram (left) can be represented as Guass code and Guass diagram (right). (b) 2.5D rendering with light and material adds apparent 3D geometry, depth, and shape to the 2D image.

Tasks of sketching knot diagrams and the three types of Reidemeister moves are mainly concerned with adding and removing crossings in a mathematically correct way. In our implementation, we design our system to ensure that no resultant crossings are closer than a threshold pixel distance d_{close} so as to avoid visual and computational ambiguities (see e.g., Figure 8).

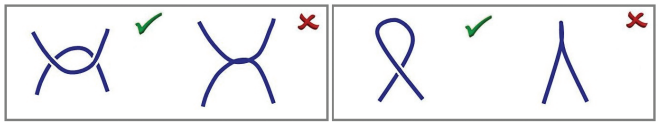


Fig. 8. Crossing points with reasonable distances are favored, in order to avoid visual and computational ambiguities.

4.2 Interacting with Knot Diagrams through the Reidemeister moves

Deforming a knot diagram by sketching the Reidemeister moves is a quite different but important experience from knot creation. Reidemeister moves involve three different types of moves applied to knot diagrams. We can apply these moves to reduce the number of crossings and to transform a given knot diagram into a simpler yet equivalent knot diagram (see e.g., Figure 9). One important point is that the Reidemeister moves are local. This means that if we have a large and complex diagram we can perform the Reidemeister moves on small pieces of the diagram with fairly light computation.

4.2.1 Restricting user manipulations to the Reidemeister moves

Our interface allows replacing a local part of the knot diagram by proposing a finite sequence of the geometric Reidemeister moves on it (see e.g., Figure 3(b)(c)). Before we detail the logical series of steps, several definitions are in order.

Definition 4.1 (crossing number $\|K\|$) *Let v be the local crossing points in a component K. We denote $\|K\|$ as the size of the v.*

Definition 4.2 (arc component) *An arc component K is a component in a knot diagram that contains no local crossing points (i.e., $\|K\|$=0).*

Definition 4.3 (twist component) *A twist component is a curve in a knot diagram that contains one single crossing that consists one local over-crossing point and one local under-crossing point sharing the same projection location.*

Definition 4.4 (safe component) *A safe component is a curve in a knot diagram that either an arc or twist component, or only goes over the rest of the knot diagram at each local crossing point (namely, a safe over-component), or only goes under the rest of the diagram at each local crossing point (namely, a safe under-component).*

Definition 4.5 (stuck component) *A stuck component is a curve in a knot diagram that is neither a safe component nor a twist component.*

With these definitions, we are ready to outline our interaction scenario. At each interaction loop, the following sequence of events occur:

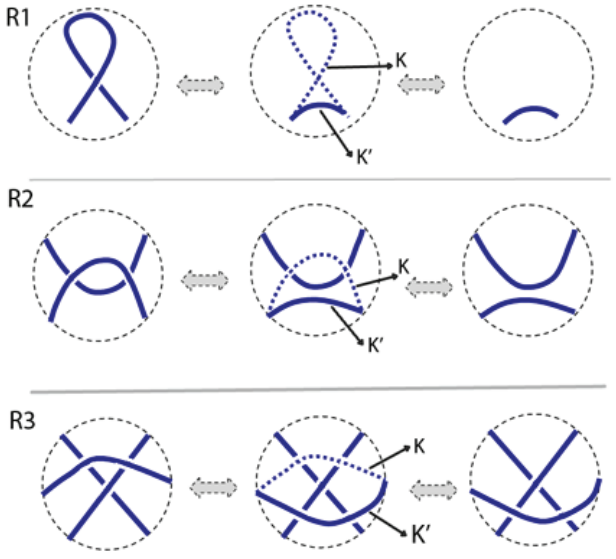


Fig. 9. Sketching the Reidemeister moves. The topmost move is referred to as R_1, the middle one as R_2, and the bottommost as R_3. For any one of these moves R, the move can be applied either from right to left or left to right (the dashed curve segments indicate the local parts of knot diagrams to be overdrawn with a Reidemeister move.)

1. **Select a local component (K) of the knot diagram**. Examples are an arc or a twist part of a knot diagram (see e.g., Figure 3(b)).
2. **K is a safe component**. Only a safe component can be selected for further modifications.
3. **Proposing a new component (K') by sketching the Reidemeister moves**. Examples are attempts to add a twist component to the diagram to replace an existing arc component, or sketch an arc component to remove a twist component (see e.g., Figure 3(c)).
4. **Recognizing the Reidemeister moves and/or planar isotopy moves**. The system only accepts a new component K' proposed to replace the original component K, when K and K' satisfy the following formulae:
   - (I.) K' is a safe component.
   - (II.) $K \cup K' = O$, i.e., the original component and the proposed component should form a unlinked component O (also called "loop" or "simple closed curve" or "Jordan curve").
   - (III.) $K \rightarrow K'$ is identified as one of the three types of the Reidemeister moves as follows:
     - (a.) K is a twist component and K' is an arc component (see e.g., $R_1^{left \rightarrow right}$ in Figure 9);
     - (b.) K is an arc component and K' is a twist component (see e.g., $R_1^{right \rightarrow left}$ in Figure 9);
     - (c.) K is a safe over/under- component and $\|K\| = 2$ and K' is an arc component (see e.g., $R_2^{left \rightarrow right}$ in Figure 9);
     - (d.) K is an arc component and K' is a safe over/under-component and $\|K'\| = 2$ (see e.g., $R_2^{right \rightarrow left}$ in Figure 9);
     - (e.) K and K' are are both safe over-components and $\|K\| = 2$ and $\|K'\| = 2$ (see e.g., R_3 in Figure 9);
     - (f.) K and K' are are both safe under-components and $\|K\| = 2$ and $\|K'\| = 2$ (see e.g., R_3 in Figure 9).
5. **Legal moves are accepted and new diagrams are constructed**. With each legal move accepted, the system constructs the new knot diagram for users to interact with further.

4.2.2 Recognizing the Planar Isotopy Moves

Planar isotopy moves are a means of deformation when transforming knot diagrams. There are typically two types of planar isotopy moves: the detour move (see e.g., Figure 10 $^{left \rightarrow right}$) and the short cut move (see e.g., Figure 10 $^{right \rightarrow left}$). Our system identifies a legal planar isotopy move when K and K' satisfy the following formulae:

(I.) K is an arc component;

(II.) K' is an arc component;

(III.) $K \cup K' = O$, i.e., K and K' form a unlinked component O.

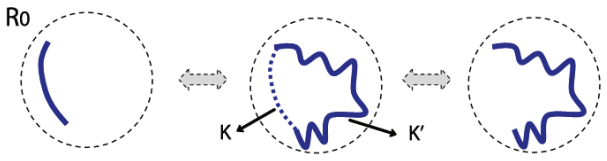


Fig. 10. R_0: the detour move and short cut move (i.e., the planar isotopy).

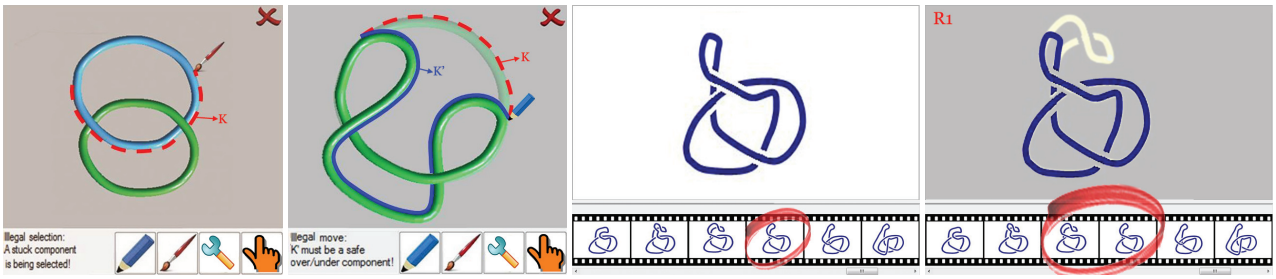
Without loss of generality, we also name the planar isotopy moves as R_0 and support it as an elementary valid move in our software environment.

4.2.3 Improving the Hand-drawn Diagrams

Our system provides a rubbing tool for smoothing knot diagrams. As the user proposes new components to replace local parts of a knot diagram, the curve gradually becomes less smooth (see. e.g., Figure 3(c)). We allow users to rub a region of the diagram to smooth it using the Least-square mesh approach [22] (the connectivity pattern in our case is a set of curves instead of a 2D mesh, thus the Laplacian filtering presented here works on a 1D space.)

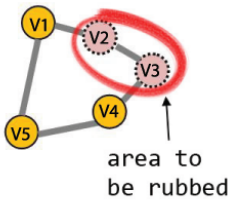
The basic idea is to construct the Laplacian matrix [22] corresponding to the link-node structure and sets the corresponding control vertices: when a local region in the knot diagram is being rubbed, only a finite number of vertices being rubbed are going to be reconstructed while nodes in the rest of the diagram are used as control vertices. For example, Figure 11 shows a very simple graph (a trivial knot sample) where vertices v_2 and v_3 are being rubbed, and vertices v_1, v_4, v_5 are being used as control vertices. The system first constructs the Laplacian matrix L based on the connectivity meshes (in our case, linked nodes), and then adds the equations F of the control vertices to make the matrix full-rank (see e.g., Figure 11, the last three rows of the matrix are the added equations of the control vertices). The following equation is then solved in least-square sense as proposed in [22] to smooth the graph:

$$Ax = b, \tag{1}$$



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

Fig. 12. User interface elements in KnotPad. (a)-(b) KnotPad recognizes illegal attempts of knot manipulation: (a) A stuck component was wrongly selected for editing. (b) An illegal move was proposed. (c) KnotPad automatically saves each valid Reidemeister move being applied as images on disk for each project. The user can navigate or track back to a valid move by selecting a snapshot icon from the Carousel-like "history" navigation interface. (d) KnotPad highlights the "diffs" between subsequent "history" moves by overlaying knot images with transparency.



V1	1	$-\frac{1}{2}$	0	0	$-\frac{1}{2}$
V2	$-\frac{1}{2}$	1	$-\frac{1}{2}$	0	0
V3	0	$-\frac{1}{2}$	1	$-\frac{1}{2}$	0
V4	0	0	$-\frac{1}{2}$	1	$-\frac{1}{2}$
V5	$-\frac{1}{2}$	0	0	$-\frac{1}{2}$	1
V1	1	0	0	0	0
V4	0	0	0	1	0
V5	0	0	0	0	1

Fig. 11. A sample graph of nodes in a knot diagram and its corresponding Laplacian matrix of the linked nodes.

where

$$A = \begin{pmatrix} L \\ F \end{pmatrix}, F_{ij} = \begin{cases} 1 & j \in control \quad vertices \quad indices \\ 0 & otherwise \end{cases}$$

$$b_k = \begin{cases} 0 & k \leq n \\ x_{k-n} & n < k \leq (n+m) \end{cases}$$

Figure 3(d) shows the smoothed diagram after the user rubs regions of the knot diagram in Figure 3(c). Note that a valid replacement of the original region and the reconstructed (i.e. smoothed) one must be identified as a R_0 move (see Figure 10).

4.3 More User Interface Elements

Our system encourages step-by-step Reidemeister moves, and thus can afford insight of knot equivalence through the entire connected interactions. However, applying sequences of Reidemeister moves is an error prone mental computation. Therefore, we choose to supplement our design with a set of useful user interface elements.

Informing the User of Invalid Moves. For example, by alerting the user when a *stuck component* is being selected for editing (see e.g., Figure 12(a)) or when a illegal move is being proposed (see e.g., Figure 12(b)), our virtual paper-like system can obviate a range of tedious and error prone interactions with a better user experience than that of real pen-and-paper.

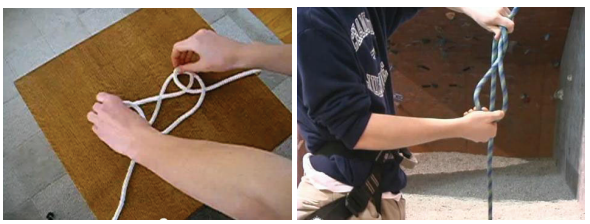
Remembering the Valid Moves. As an additional interface element, our system automatically saves the knot data and image frames of the knot diagrams when legal moves are applied. Therefore, the user can navigate through all generated knot images to examine the whole sequence of moves applied to connect one knot diagram into another (see Figure 12(c)). Visualizing the Reidemeister moves is an exploring process, and it is often helpful to revert to a previous state. KnotPad can also allow the users to "undo" most recent moves by tracing back to one of the previous valid moves recorded.

Displaying the "diffs" Between Moves. The user can select two subsequent snapshots to visually examine the type of move applied to connect them. KnotPad renders the "before" and "after" knot images in different colors and overlays them with transparency, therefore the local change between them can be visualized (see e.g., Figure 12(d)).

5 Reducing Interaction Complexity

5.1 The Problem

The Reidemeister moves drawing interface in principle is sufficient to offer an interactive visualization that is diagrammatically true and closer to traditional paper based analysis and manipulation of knot diagrams, and thus closer to the mathematical problems we want to solve. However, in practice, many interesting knot diagram manipulation tasks involve drawing a large number of Reidemeister moves [9] and a complex interface requiring too much user input can reduce user effectiveness and significantly increase the learning curve.



(a) (b)

Fig. 13. (a) When demonstrating how to tie a Spanish Bowline (sometimes called the double loop knot) with a rope, one brings the two small loops up by pulling part of the loops, very much like performing an R_2. (b) A rock climber makes a bight by simply twisting the rope, which seems very much like adding one knot crossing with an R_1.

Therefore we are motivated to enable the Reidemeister moves interface to be explored and examined in more fluid, and natural ways, yet still benefiting from the advantages of our paper-like interactive system. Making such an improvement is possible, for example, in many times we can propose the Reidemeister moves more easily by deforming a sub-region of the knot image. For example, we can first select a local part of the diagram then perform a "pull", "drag" or "twist" to propose the changes. As shown in Figure 13(a) one can first grab and then drag part of the physical rope over a loop or a crossing, to accomplish a type II or type III Reidemeister move. In Figure 13(b), one can

hold the ends of a rope with one hand and twist the ends of the formed loop with the other hand to make a knot, very much like the type I Reidemeister move. In this section, we first consider how Reidemeister moves can be explored to advantage with a deformation based editing interface yet still capable of restricting user manipulation to the Reidemeister moves. We then discuss how to further simplify user input by combining sets of sequential valid moves into frequently-used elementary moves. We also propose using Gauss code to group frames in the navigation interface to provide more identifiable and interesting manipulation path for visualization purpose.

5.2 The Solution

5.2.1 Recognizing Dragging Gestures for Valid Moves

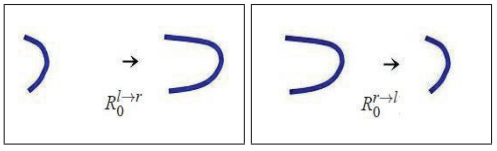
The basic idea is to supplement our interface design with a free-form editing interface with the support for the Reidemeister moves. When the user selects a local part of the diagram and drags one of the vertices to propose a move, the end-points and the manipulated vertex of the local part are set as the corresponding control vertices of the Laplacian matrix, and the local part is deformed in the least-square sense as proposed in Equation 1.

Different from the drawing approach to support the Reidemeister moves where crossings were created and crossing signs were resolved one at a time separately, performing a free-form like move typically adds or removes a set of local crossing points along the whole local part to be overdrawn. Therefore there are two critical components in the implementation of our dragging interface to simplify user input:

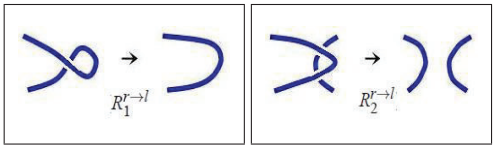
1. **Dragging a local part of the diagram for deformation.** This is implemented again using the Least-square mesh approach [22]: the user selects a local part of the diagram (Figure 3(e)) and then drag one of the vertices to propose the shape reconfiguration (Figure 3(f)).

2. **Resolving crossings to ensure valid moves.** We need a solution to resolve the crossing signs to ensure mathematically valid moves being produced with each dragging motion. The three basic types of Reidemeister moves can be decomposed into ten elementary moves in our user interface with support for dragging gestures, and these moves fall into four categories:

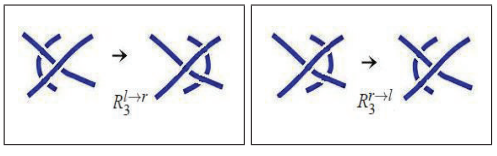
   (a) moves that have no crossings involved in either initial or goal state ($R_0^{l\to r}$, $R_0^{r\to l}$)



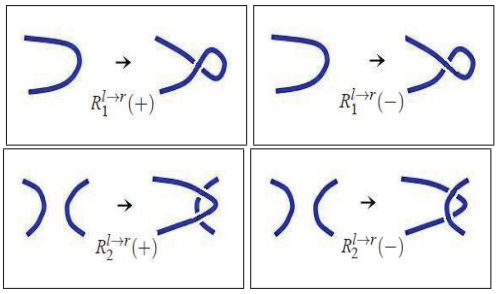
   (b) moves that have all crossings removed in the goal state ($R_1^{r\to l}$, $R_2^{r\to l}$)



   (c) moves that keep the same crossing signs in both states ($R_3^{l\to r}$, $R_3^{r\to l}$)



   (d) moves that start with no crossing in the initial state but end with crossings in the goal state ($R_1^{l\to r}(+)$, $R_1^{l\to r}(-)$, $R_2^{l\to r}(+)$, $R_2^{l\to r}(-)$)



Elementary moves in categories (a) and (b) can basically be implemented using the free-form editing method without resolving the crossings, the user can grab and drag a control point on part of the knot diagram to perform the valid moves while our system recognizes the valid moves by ensuring there are no resultant crossings in the goal state. Moves in categories (c) can be implemented following the same fashion, except that our system remembers the crossing signs in the initial state and automatically assigns the same crossing signs in the goal state (see e.g., Figure 14). Now the elementary moves in category (d) needs crossing signs handling, they start from the initial state with no crossings but produce a group of crossings (of the same sign) in the goal state. To resolve the crossings for these moves, we follow the same fashion of resolving crossing signs in our knot drawing interface: KnotPad will explicitly query the user for over- or under-crossing choice when the curve is being dragged and resultant collisions are about to happen during the dragging motion.

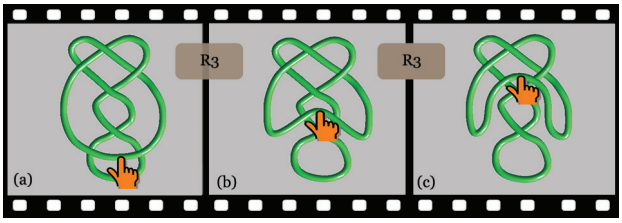


Fig. 14. Pulling a local part of the diagram to create the Reidemeister move. (a)-(c): the resultant crossings on the deformed curve segment is determined by its original relative depth order to ensure a legal Reidemeister move being produced.

5.2.2 Combining Moves Into New Elementary Moves

It is also possible to simplify user input by combing some sequential elementary moves into a new one. For example, as we looked into the recorded user data (using KnotPad's navigation interface) from preliminary user studies, we observed a new "detour" move that should be also recognized and accepted as an elementary move in our interface, since it has been frequently used in various manipulation tasks. As illustrated in Figure 15, this frequently-drawn detour move on a strand along the direction of the other strand crossing over it had to be decomposed into a sequence of two (or more) consecutive R_2 Reidemeister moves. To simplify user input, our interface chooses to recognize and support it as a variant of R_0 elementary move.

5.2.3 Grouping Frames in Navigation Interface

Many tasks of finding knot equivalence can generate very long sequences of valid moves. However when we navigate through recorded valid moves, we most likely are only interested in more identifiable

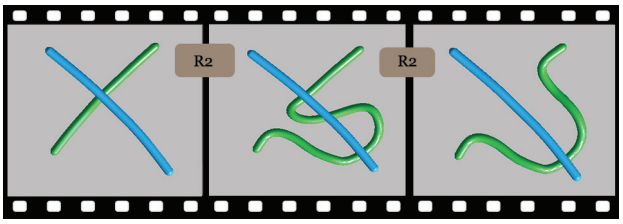


Fig. 15. A "detour" move that displaces the local crossing point. The move can be decomposed into two R_2.

moves. There are many moves such as "detour" or "shortcut" that were only applied to decorate the knot diagram in preparation for a more identifiable move (e.g., see Figure 16, although an R_0 changes the shape of the diagram, the Gauss code in the goal state is not changed from the initial state), therefore our navigation interface automatically groups these moves to shorten the navigation path in our user interface.

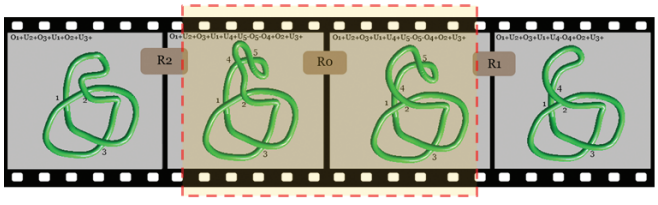


Fig. 16. A "detour" move that was applied using a R_0 generates a "decorated" diagram with same Gauss code. We group frames produced by R_0 (or with same Gauss code as previous frame)

6 More Samples

With the Reidemeister moves (R_1, R_2, R_3) and the planar isotopy moves (R_0), we can now deform one knot diagram to another by proposing the Reidemeister moves (and with only these moves). We can apply these moves to reduce the number of crossings and to transform a given knot diagram into a simpler (or sometimes more complex) yet equivalent knot diagram.

Studying Knot Equivalence. In Figure 17, a sequence of Reidemeister moves and planar isotopy moves are applied to the trefoil knot, which is believed to be the simplest knot. The resultant diagram in Figure 17(k) contains a knot diagram very different than that of Figure 17(a), yet these two diagrams represent equivalent knots: the trefoil knot-type.

Interacting with Mathematical Links. One more important point is that the Reidemeister moves are local. This means that if we have a large knot or link diagram we can perform a single Reidemeister move on small pieces of the diagram, locally in each quality. Figure 1(b) shows how one can apply a sequence of Reidemeister moves to a tangled link diagram.

Interacting with Braid Diagrams. In Figure 18 and Figure 19, a sequence of Reidemeister moves are applied to a group of strands each of which begins on a horizontal top and ends on the bottom with no intersections of each other (mathematically named *braid diagram*). These two figures show that we can use the Reidemeister moves to generate braid equivalence, and represent an interactive visual proof of the two elementary relations used in braid theory [4].

7 User Study

Our user interface is based on OpenGL and Windows WinForm API. The software runs on a Dell PC desktop with 3.2GHz Intel Pentium 4 CPU. The control of mouse cursor motion is programmed using Win32 APIs. A study to gauge user experience with our user interface design

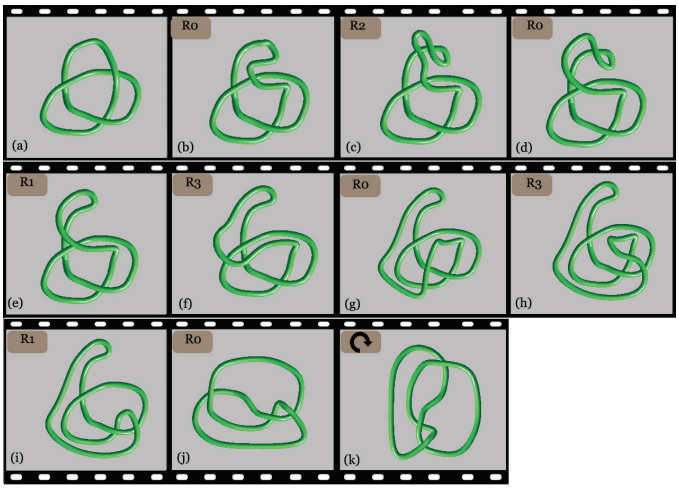


Fig. 17. A trefoil knot is turned into a "new" diagram through a sequence of Reidemeister moves.

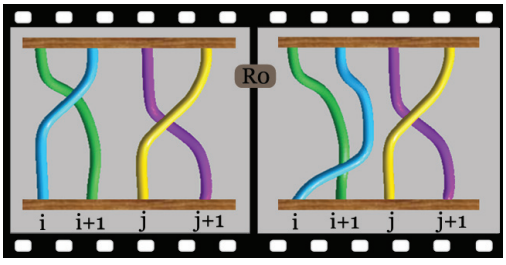


Fig. 18. A braid diagram for $\sigma_i\sigma_j$, and after applying the R_0 we have a new braid diagram for $\sigma_i\sigma_j$.

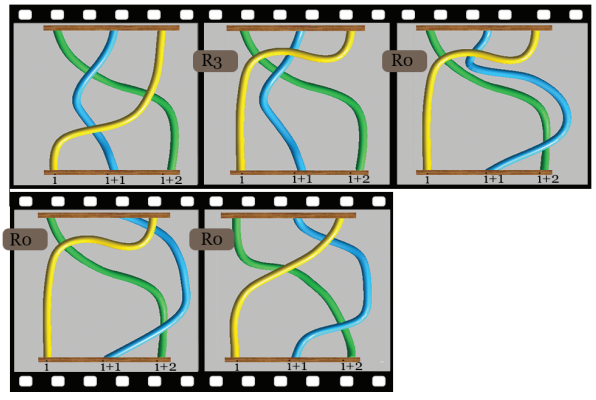


Fig. 19. A braid diagram for $\sigma_i\sigma_{i+1}\sigma_i$, and after applying a sequence of Reidemeister moves, we have a new braid diagram for $\sigma_{i+1}\sigma_i\sigma_{i+1}$.

was conducted. A total of 18 participants were involved in this study: 16 males and 2 females, aged between 17 and 23. They were all undergraduate students having considerable familiarity with personal computers and sketching interface like Microsoft Paint software. Prior to this experimental study, none of them had used the system we developed in this work or had knowledge of knot theory. Before performing the required tasks, all of them were given a brief introduction of knot theory and the three types of Reidemeister moves, plus a demonstration of how each user interface element works in our system. We note that in order to evaluate the different aspects of our user interface, we divided the participants into two groups where the first group (Group_I) was demonstrated and allowed to perform the Reidemeister moves by just drawing the new topological situation and the second

group (Group$_{II}$) was demonstrated how to perform the Reidemeister moves by both drawing and by dragging control points.

7.1 Tasks and Measures

The 18 participants were all asked to perform the following six tasks using our software:

1. Sketch a trefoil knot (i.e., $knot_1^3$);
2. Select part of $knot_1^3$ and perform the first type of Reidemeister move;
3. Select part of $knot_1^3$ and perform the second type of Reidemeister move;
4. Select part of $knot_1^3$ and perform the third type of Reidemeister move;
5. Prove the following knot diagram is a trivial knot (now in Figure 20(a)), and provide the sequence of Reidemeister moves needed;
6. Prove the following two knot diagrams are equivalent, and provide the sequence of Reidemeister moves needed (now in Figure 20(b)).

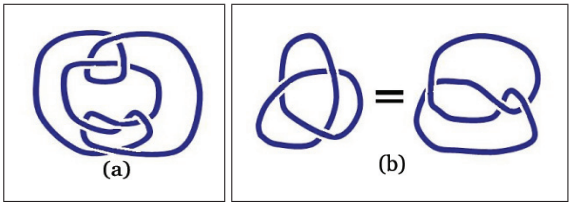


Fig. 20. Knot diagram samples used in our user study.

We recorded and analyzed the **task completion rate**, **average completion time**, **average number of moves** used to complete the tasks, **average duration for a valid move** for participants in both groups. We also asked a graduate student from math department to give **numeric ratings** (5 point scale) to participants based on their final generated images' correctness and smoothness.

Among the six listed operations, the majority found the first four tasks to be the easiest to perform as the knot structure is the simplest and the tasks only required a small number of Reidemeister moves. Most participants also found task 5 was to be intuitive because the knot structure is easy to comprehend. Task 6 was found to be the most difficult, with several participants (in both groups) having problems accomplishing the task within the given time. However, most of them were all able to move forward and finish the task given some hints and extra time.

As shown in Table 1, most participants in both groups found that the software allowed them to draw and deform various mathematical knot diagrams. However, Group$_{II}$ has been recorded significantly improved completion time (914.01 sec. compared to 1210.34 in Group$_I$) and reduced number of moves compared to those in Group$_I$ (33 compared to 43 in Group$_I$), which proved that by supporting dragging based moves our interface has been able to simplified user inputs. We also notice the average duration to complete a valid move is also reduced in Group$_{II}$ (15 sec. compared to 21 sec. in Group$_I$), which is also mainly due to Group$_{II}$ has been using dragging based moves to finish approximately 83% of the tasks. In addition, the generated knot images in Group$_{II}$ are relatively more smooth compared to those from Group$_I$.

7.2 Evaluation of KnotPad

Our general observation was that users had a positive experience using the system for sketching knot diagrams and deforming knot diagrams by applying the Reidemeister moves. With a little bit of instruction and practice, the participants were soon able to learn how to use the interface.

After finishing the above experiments, the participants in both groups were required to rate the two systems on a scale of 1 to 5, where 1 means completely disappointed and 5 means completely satisfied. KnotPad receives an average score of 4.5 from Group$_{II}$ as compared to 4.1 from Group$_I$.

Table 1. Performance of KnotPad's features: completion rate, average time, number of moves undertaken, average time for accomplishing a valid move, percentage of dragging motion used, and numeric rating received for the accomplished tasks.

Group	CMPL. rate	CMPL. time (sec.)	♯ of moves	Duration per move (sec.)	PCT. of dragging	Rating
I	0.88	1210.34	43	21	0%	3.5
II	0.90	914.01	33	15	83%	3.8

The following is a summary of the common responses of the participants who used our system:

- Features of the application participants liked most: the ability to alert users when illegal moves are being proposed, the free-form like deformation method to create Reidemeister moves, and the collision-sensing mouse cursor.
- Suggestions and comments: several participants indicated they would like to see a touch-interface version of our software. One participant suggested knot manipulation can be further assisted with a multi-touch interface. Several participants mentioned that it is a well-designed virtual paper for studying knot theory.

7.3 Compare with Physically based 3D Knot Visualization

We also invited the participants in Group$_{II}$ to work with *KnotPlot* [20] and observe their performance on the same 6 tasks using the 3D knot visualization software. Without surprises, most participants can use *KnotPlot* to complete knot drawing tasks (basically done in a plane assisted by general 3D rotations). However, since the knot structure and dynamics are modeled as beads connected by the (linear) springs, general Reidemeister moves can not be easily accomplished with the 3D physically based interface. Although participants can try pulling and dragging on the 3D knots to verify some specific knot equivalence, none of them are able to provide the real topological path through the Reidemeister moves space.

Just like *KnotPlot*, many previous knot visualization systems focus on simulating the 3D *global* physics of knots and ropes, thus when deforming the ropes (typically done by dragging a control point) all collisions must be detected and avoided at every interaction cycle, as each of them will affect the rope's global motion at the next cycle. On one hand, such a knot or rope is modeled as a large collection of small rigid pieces a_1, a_2, ..., a_n, thus the collision detection and avoidance become computationally expensive ($O(n^2)$) even with some improving approaches such as building a bounding-volume hierarchy (BVH) etc. On the other hand, these previous physically based modeling approaches to simulate 3D knot deformation are intended for global configuration (e.g., to simulate real-time knot tying), and are less close to knot theory which is mainly concerned with applying local mathematically valid moves, i.e., the step-by-step Reidemeister moves.

Compared to all these previous efforts, KnotPad transforms knot deformation task into sequences of local moves (i.e., the Reidemeister moves). Collisions are handled when components are being sketched (locally). When the user is sketching, collisions are detected at every sketching cycle between the last rigid piece ($a_{n-1}a_n$) and the rest of the rigid pieces (a_1a_2, a_2a_3 ... $a_{n-2}a_{n-1}$) (or only a subset of the rest, if BVH approach used), it always takes $\Theta(n)$ at each cycle, which is optimal in the worst case. When the user finishes sketching , the system only needs to validate the proposed move at most $O(1)$ (with the algorithm we proposed) in each quality. Therefore KnotPad has linear scalability and its performance is only limited to the total number of the rigid pieces, rather than the number of crossings or the number of distinguishable knots. KnotPad's scalability has allowed users to draw and manipulate various non-trivial mathematical knots, links, and braids. Users were able to sketch and manipulate a group of mathematical links of up to 90 crossings in total, formed by around eight hundred rigid pieces (we note that this performance was limited to the screen pixels available rather than KnotPad's scalability.)

8 Conclusion and Future Work

In topology, knot theory is the study of mathematical knots using the three types of Reidemeister moves. Based on a wide variety of prior art, we have developed an innovative paper-like topological drawing interface where one is allowed to study mathematical knots through applying step-by-step Reidemeister moves. We described how a "collision-sensing" cursor can be exploited to draw knot diagrams and propose valid moves on knot diagrams. Our paper detailed our interaction model for recognizing and supporting digital Reidemeister moves in our software environment. A family of user interface elements were introduced on how to polish knot diagrams, obviate error prone interactions, and to make Reidemeister moves based interaction more easily and friendly. The results of our user study show that the users were able to use our interface to create and deform sophisticated knot diagrams through the Reidemeister moves interface successfully.

Our next step is to conduct another evaluation of our system with knot theory experts to identify, design, and implement more desired features to be supported in KnotPad. For example, our tool currently supports applying step-by-step Reidemeister moves to produce knot equivalence, and exploits visualization and visual cues to record and compare subsequent knot images produced from one user. We plan to work with mathematicians and knot theory experts to find the best way to support identifying and comparing knot diagrams (produced by different users) both computationally and visually (e.g., Jones Polynomial is already known to be an invariant of an oriented knot or link and can be computed to identify knots from a topology perspective [26].)

Starting from this basic framework, we also plan to extend the range of objects for which we can support to include more complex knots, links, and Riemann surfaces. The interaction paradigm presented in this paper has an 4D analogy: effective manipulation of 4D surfaces (e.g., 4D knotted sphere) could potentially be greatly assisted by a friendly interface supporting 4D Reidemeister moves [19]. We anticipate that a graphical user interface that supports step-by-step 4D Reidemeister moves can be used to simulate more complex and abstract math phenomena such as untying a 4D twist-spun trefoil knot.

Other direction of future work includes migrating our software to touch environment (e.g., on iPad) to make interaction with knot diagrams more facile and satisfying by using abilities of the *hand* beyond simple position control.

References

[1] W. V. Baxter, V. Scheib, and M. C. Lin. dAb: Interactive haptic painting with 3D virtual brushes. In E. Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 461–468. ACM SIGGRAPH, ACM, 2001.

[2] E. A. Bier. *Snap-Dragging: Interactive Geometric Design in Two and Three Dimensions*. PhD thesis, EECS Department, University of California, Berkeley, May 1988.

[3] J. Brown, J. C. Latombe, and K. Montgomery. Real-time knot-tying simulation. *The Visual Computer*, 20(2-3):165–179, 2004.

[4] M. Chiodo. An introduction to braid theory. Master's thesis, University of Melbourne, 2005.

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

[6] C.W.Reynolds. Steering behaviors for autonomous characters. In *Proceedings of the 1999 Game Developers Conference*, pages 763–782, 1999.

[7] B. Forsyth. Intelligent support of interactive manual control: Design, implementation and evaluation of look-ahead haptic guidance. Master's thesis, The University of British Columbia, 2004.

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

[9] J. Hass and J. C. Lagarias. The number of reidemeister moves needed for unknotting. 1998.

[10] L. H. Kauffman. The mathematics and physics of knots. *Reports on Progress in Physics*, 68(12):2829–2857, 2005.

[11] J. M. Kennedy. Optics and haptics: The picture. In *Proceedings on Multimodality of Human Communication: Theory, Problems and Applications.* University of Toronto, 2002.

[12] A. Lecuyer, S. Coquillart, A. Kheddar, P. Richard, and P. Coiffet. Pseudo-haptic feedback: Can isometric input devices simulate force feedback? *Virtual Reality Conference, IEEE*, 0:83, 2000.

[13] C. Livingston. *Knot Theory*, volume 24 of *The Carus Mathematical Monographs*. Mathematical Association of America, Washington, U.S., 1993.

[14] A. M. Okamura. *Haptic Exploration of Unknown Objects*. PhD thesis, Stanford University, Department of Mechanical Engineering, California, USA, June 2000.

[15] A. M. Okamura and M. R. Cutkosky. Feature detection for haptic exploration with robotic fingers. *The International Journal of Robotics Research*, 20(12):925–938, Dec. 2001.

[16] J. G. Park and G. Niemeyer. Haptic rendering with predictive representation of local geometry. In *HAPTICS*, pages 331–338, 2004.

[17] J. Phillips, A. M. Ladd, and L. E. Kavraki. Simulated knot tying. In *ICRA*, pages 841–846. IEEE, 2002.

[18] A. Pusch, O. Martin, and S. Coquillart. Hemp-hand-displacement-based pseudo-haptics: A study of a force field application and a behavioural analysis. *Int. J. Hum.-Comput. Stud.*, 67(3):256–268, 2009.

[19] D. Roseman. Reidemeister-type moves for surfaces in four-dimensional space. *Knot theory (Warsaw, 1995)*, pages 347–380, 1998.

[20] R. G. Scharein. *Interactive Topological Drawing*. PhD thesis, Department of Computer Science, The University of British Columbia, 1998.

[21] S. Snibbe, S. Anderson, and B. Verplank. Springs and constraints for 3d drawing. In *Proceedings of the Third Phantom Users Group Workshop*, Dedham, MA, 1998.

[22] O. Sorkine and D. Cohen-Or. Least-squares meshes. In *Proceedings of Shape Modeling International*, pages 191–199. IEEE Computer Society Press, 2004.

[23] J. Spillmann and M. Teschner. An adaptive contact model for the robust simulation of knots. *Comput. Graph. Forum*, 27(2):497–506, 2008.

[24] J. van Wijk and A. Cohen. Visualization of the genus of knots. In *Visualization, 2005. VIS 05. IEEE*, pages 567 – 574, oct. 2005.

[25] F. Wang, E. Burdet, A. Dhanik, T. Poston, and C. L. Teo. Dynamic thread for real-time knot-tying. In *WHC '05: Proceedings of the First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 507–508, Washington, DC, USA, 2005. IEEE Computer Society.

[26] E. Witten. Quantum field theory and the jones polynomial. *Communications in Mathematical Physics*, 121(3):351–399, 1989.

[27] G. Wolberg and I. Alfy. Monotonic cubic spline interpolation. In *Computer Graphics International, 1999. Proceedings*, pages 188 –195, 1999.

[28] Y. Wu. An md knot energy minimizing program. Department of Mathematics, University of Iowa.

[29] M. A. Zahariev and C. L. MacKenzie. Auditory, graphical and haptic contact cues for a reach, grasp, and place task in an augmented environment. In *Proc. of 5th Intl. Conf on Multimodal Interfaces (ICMI)*, pages 273–276, New York, 2003. ACM, ACM Press.

[30] H. Zhang and A. Hanson. Shadow-driven 4d haptic visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1688–1695, 2007.

[31] H. Zhang and A. J. Hanson. Physically interacting with four dimensions. In G. Bebis, R. Boyle, B. Parvin, D. Koracin, P. Remagnino, A. V. Nefian, M. Gopi, V. Pascucci, J. Zara, J. Molineros, H. Theisel, and T. Malzbender, editors, *ISVC (1)*, volume 4291 of *Lecture Notes in Computer Science*, pages 232–242. Springer, 2006.

[32] H. Zhang, S. Thakur, and A. J. Hanson. Haptic exploration of mathematical knots. In G. Bebis, R. D. Boyle, B. Parvin, D. Koracin, N. Paragios, T. F. Syeda-Mahmood, T. Ju, Z. Liu, S. Coquillart, C. Cruz-Neira, T. Müller, and T. Malzbender, editors, *ISVC (1)*, volume 4841 of *Lecture Notes in Computer Science*, pages 745–756. Springer, 2007.

[33] H. Zhang, J. Weng, and A. Hanson. A pseudo-haptic knot diagram interface. In *Proc. SPIE 7868 (VDA)*, volume 786807, pages 1–14, 2011.