# CONSTRAINT-BASED ROBOT KNOT TYING

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

in

Computer Science

by

Weifu Wang

DARTMOUTH COLLEGE

Hanover, New Hampshire

Dartmouth Computer Science Technical Report TR2016-792

Examining Committee:

(chair) Devin Balkcom

Peter Winkler

Emily Whiting

Dmitry Berenson

F. Jon Kull, Ph.D.
Dean of Graduate Studies

# Abstract

In this thesis, we present different approaches to tying knots using robots by enforcing different types of constraints. We attack the problem from three different directions; mechanical design, motion planning with simple control strategies, and theoretical analysis.

We present the first complete generalized fixture based knot tying approach that is able to *arrange* and *tighten* knots, building on Matthew Bell's work on arranging simple knots using fixtures [Bel10]. Physical constraints are provided by these fixtures to reduce the possible string deformation during knot tying. Fixtures are designed automatically using proposed design principles, and constructed based on the designs using 3D modeling software. First, fixtures are used to transform knots from an untangled configuration to a knotted configuration (*arrangement*). Furthermore, we design fixtures that can *tighten* these arranged knots by pulling the open ends of the knot without sensing. These mechanical designs provide fast, reliable and repeatable knot tying devices.

We also study some simple control strategies for string without the need to simulate the deformations, as part of a collaboration with Berenson [Ber13, WBB15b]. Vector fields are generated by a sequence of virtual rings, which are used to guide the motions of string as local controllers. This simple control strategy is first used in threading string through one or more small openings. The simulated experiments suggest the approach is robust to errors. The initial experiments with a *Da Vinci* surgical robot arm were able to thread a $3.5mm$ diameter yarn into a $4.9mm$ diameter washer under faulty sensing. We further tied knots around generic fixtures without sensing, demonstrating the capability of tying different knots around non-specific fixtures.

We at last study the *tie-ability* of knots. We first analyze the complexity of the knot tying task, in terms of how many points of contact are necessary and sufficient. These fingers serve as both physical and virtual constraints to stretch string into a polygonal arc, so that we can compute possible configurations of the string easily. We present provably correct algorithms to find a sufficient number of fingers for knot grasping, and simple motions that can fold knots sequentially. Further analysis of the Gauss code—a common knot description used in knot theory— and comparison between knots and weaving encourages a novel knot tying approach, using a small number of re-grasps. An algorithm is derived to compute a sufficient number of re-grasps to tie the given knot based on the Gauss code. Physical experiments are implemented to demonstrate the novel knot tying strategy, first using a Da Vinci surgical robot to tie knots autonomously, and then through collaboration between a human and an Adept Cobra industrial arm.

We present robot knot tying methods to physically tie knots and understand the complexity of the task. As a special example of string manipulation, we hope to gain insights on how to better manipulate string and possibly more general flexible objects through our research.

# Acknowledgments

Thanks to my family for everything, especially for supporting my decision for pursuing a Ph.D. degree, and for all the encouragement and support over the years.

I would like to express my special appreciation and thanks to my advisor Professor Dr. Devin Balkcom, for being a tremendous mentor and friend. You took a chance to take me as your Ph.D. student, and I hope you do not regret the decision as it certainly has been a blessing for me. I cannot imagine of having a better adviser and friend. Throughout the years, you gave me countless advice on both research and on my career, helping me to grow as a researcher. Your exemplary mentorship encouraged me to pursue an academic career as well. I would also like to thank Professor Peter Winkler, Professor Emily Whiting, and Professor Dmitry Berenson for being on my thesis committee. Many thanks to my collaborators, Dmitry Berenson and Matthew Mason, for introducing me to different areas of robotics and allowing me to develop different chapters of this thesis, and thanks to Yu-han Lyu and Yinan Zhang for working with me on different projects and helpful discussion to help me work through this thesis. Thanks to Matthew Bell for starting the work on knot tying before my arrival. I have had so much fun working on this problem. Thanks to Andrei Furtuna and Paritosh Kavathekar for all the help and advise to help me pursue my Ph.D.. Thanks to Dartmouth and the department for providing such a relaxing and encouraging academic environment. Despite the freezing winters, I have had the best few years in my life here in Hanover.

# Contents


**1 Introduction** **1**
- 1.1 Potential applications . . . 5
- 1.2 Organization . . . 6

**2 Related work** **9**

**3 Mathematical knots** **14**

**4 Fixture based knot tying** **17**
- 4.1 Four-piece fixture . . . 17
  - 4.1.1 Pulling vs. pushing . . . 21
  - 4.1.2 Arranging knots around objects . . . 24
- 4.2 From arrangement to tightening . . . 24
  - 4.2.1 Automated design process . . . 28
  - 4.2.2 No collision among tilted rods . . . 30
- 4.3 Precise tightening . . . 33
  - 4.3.1 New knot arrangement method . . . 33
  - 4.3.2 New tightening approach . . . 34
  - 4.3.3 Automated fixture design . . . 38
  - 4.3.4 Active tightening fixtures . . . 40
- 4.4 Ruyi knot tying . . . 41
- 4.5 Conclusions . . . 44

**5 String manipulation** **51**
- 5.1 Introduction . . . 51
- 5.2 Diminishing Rigidity Jacobian . . . 52
- 5.3 Knot tying using Da-Vinci robot . . . 62
- 5.4 Conclusions . . . 65

**6 Knot folding** **66**
- 6.1 Polygonal knots and polygonal knot diagrams . . . 66
- 6.2 Knot grasping . . . 68
- 6.3 Knot folding . . . 79

6.3.1 Loose complexity bound . . . 82
6.4 Conclusions . . . 84

**7 Knot tying with few re-grasps** **85**
7.1 Introduction . . . 85
7.2 Knots and weaving . . . 86
7.3 Decoding the Gauss code . . . 87
7.3.1 Tying knots: forming or removing crossings . . . 87
7.3.2 Counting the number of weaving motions . . . 87
7.4 Knot weaving . . . 89
7.4.1 Aligning crossings on a straight line for weaving . . . 89
7.4.2 Re-grasping and weaving . . . 91
7.4.3 Knot weaving with Da Vinci . . . 92
7.4.4 Robot human collaboration . . . 94
7.5 Complexity of knot tying . . . 94
7.6 Conclusions . . . 94

**8 Conclusions and Future work** **98**

# List of Figures


1.1 An example of the cat's cradle game. . . . 2
1.2 Different ways to model string in different areas due to different objectives. . . . 3
1.3 Comparison of two cloverleaf knots, one tied using our fixture, one tied by simple pulling on the open end, without any constraint. . . . 4
1.4 Arranging a knot $7_1$ with Da Vinci robot arm. . . . 6

3.1 A knot diagram of a shoelace unknot, with all the crossings labeled with numbers, and all th cells labeled with letters. The Guass code for the shoelace knot is: $1^+$, $2^-$, $3^-$, $4^+$, $5^-$, $6^-$, $7^+$, $3^+$, $2^+$, $1^-$, $4^-$, $5^+$, $6^+$, $7^-$. . . . 15
3.2 Knot diagrams of unknots. . . . 15
3.3 A trefoil knot is cut into an overhand knot. From left to right, one drawing of trefoil knot, a different drawing of trefoil knot, and an overhand knot. . . . 16

4.1 An example of a four-piece arrangement fixture for an overhand knot. . . . 18
4.2 The Carrick Bend fixture designed with multiple pressurized air inputs. The red plates are additional layers that serve as a manifold that distributes air through the fixture. . . . 19
4.3 The opening mechanism for the four-piece fixture. . . . 21
4.4 Machined complete fixture opener with fixture components attached. . . . 21
4.5 Human-tied river knot, and (unimplemented) four-piece fixture design. . . . 22
4.6 The modular test structure with rollers (silver ball bearings) installed at every turn. . . . 23
4.7 Arranging an overhand knot around a ring. . . . 23
4.8 Two cloverleaf knots tied by hand. . . . 25
4.9 String tightening around set of rods. . . . 26
4.10 The illustration of the search for shortest homotopy curve. . . . 26
4.11 On a continuous path from $a$ to $j$, segment from $a$ to $f$ is a $x$-monotone path, and segment from $f$ to $j$ is another $x$-monotone path. . . . 27
4.12 A monotone path $\pi$, points above $\pi$: $B_U$, points below $\pi$: $B_L$, the lower envelope $U$ of $B_U$, and upper envelope $L$ of $B_L$. . . . 27
4.13 Double coin knot. . . . 28
4.14 The fixture design process. . . . 29
4.15 The tightening fixture for the double coin knot and a knot tightened using this fixture. . . . 30
4.16 Cases for proof of Theorem 2. . . . 31
4.17 Knot tying fixtures with tilted rods for shoelace unknot. . . . 32

4.18 Corresponding knots tied using fixtures shown in Figure 4.17. . . . . 32
4.19 A cloverleaf knot. . . . . 33
4.20 Contact diagrams on loose knot diagrams and machine-tightened sounding line and cloverleaf knot. The range between the brackets indicates the knot units, and the black circles indicates the crossings and their corresponding locations along the string. . . . . 34
4.21 Two different configurations of the gripper grasping the spool used to arrange the knots. . . 35
4.22 Examples of tightening with tilted rods where assumptions breaks. . . . . 35
4.23 When multiple strands of string contact the same rod, insert one extra rod in the cell. . . . . 36
4.24 The notation for the triangle where the string wraps around the moved rod. . . . . 36
4.25 Rod mounted on a slider to move. . . . . 38
4.26 Automated design of the fixture for the cloverleaf knot using presented procedures . . . . . 39
4.27 sounding line arrangement process. . . . . 43
4.28 The first knot unit along the sounding line is tightened around the stationary pin. . . . . 44
4.29 Tying the shoelace (un)knot. . . . . 45
4.30 Tightening a cloverleaf knot. . . . . 46
4.31 Machine-tied cloverleaf knot. . . . . 46
4.32 The first knot unit along the sounding line is tightened around the stationary pin. . . . . 46
4.33 The knot diagram of a cloverleaf knot. . . . . 47
4.34 Model of the track and bar-linkage system for a cloverleaf knot. . . . . 47
4.35 Tightening cloverleaf knot using active fixture. . . . . 48
4.36 A hand tied Ruyi knot. . . . . 48
4.37 An illustration of the capstan equation. . . . . 49
4.38 Friction provided by the rotating rods between different string wrap around different rod surfaces. . . . . 49
4.39 The initial layout of the fixtures and the rotating rods for tying Ruyi knot. Powered rods are labeled with yellow dots, and all other rods are labeled with red dots. . . . . 50

5.1 An example of magnetic field induced by a current carrying planar circle loop. . . . . 54
5.2 Dynamic selection of the second reference point; $d_1 < d_2$, and $\theta_2 < \theta_1$. . . . . 55
5.3 A loop used for guiding the string, and a disk for detecting successful insertion. . . . . 56
5.4 Re-grasping strategy that considers gravity. Left: The configuration before re-grasp. Right: the configuration after re-grasp. . . . . 58
5.5 Simulation results of threading a sequence of loops, Figures 5.5a to 5.5d show several snapshots from the simulation. . . . . 58
5.6 Simulation results of threading a sequence loops with various ring orientations. Figures 5.6b to 5.6d show the frequent switches near the first loop, due to gravity. The rest of the figures show re-grasping at further and further locations. . . . . 59
5.7 Simulation results of threading while enforced grasping far from reference points, showing the deformation of string during execution. . . . . 59
5.8 Simulation results of threading while enforcing grasping even further from reference points, showing more deformation of string during the execution. . . . . 60

5.9 Experiment using Da Vinci robot with the smallest washer. Washer diameter was 4.9 mm, and the yarn diameter was 3.5 mm. The first three frames show how the controller rotates the gripper to align the yarn with the insertion direction, but the first pass missed the washer, as shown in the fourth frame. Then, the controller made a second attempt (with no intervention) and successfully threaded the washer in the last frame. . . . . 61
5.10 Example of threading the needle after the target moved during execution (second frame). Gripper trajectory is shown in yellow. . . . . 62
5.11 The three washers used in experiments, the yarn next to a nickel for scale reference. . . . . 62
5.12 One unit of the generalized fixture used for tying knots. . . . . 63
5.13 Arranging an overhand knot using a Da Vinci robot arm, around generalized fixtures. . . . . 63
5.14 Arranging a shoelace unknot using a Da Vinci robot arm, around generalized fixtures. . . . 64
5.15 Arranging a surgeon's knot using a Da Vinci robot arm, around generalized fixtures. . . . . 64

6.1 An example of the edges related to a crossing. . . . . 71
6.2 The line segments that pass through the first crossing. . . . . 72
6.3 Place all exterior crossings for double coin knot. . . . . 74
6.4 Finding placement for $v$ based on the exterior crossings it connects to. . . . . 75
6.5 Example of intersection of line segment connection even end points belong to the same connected region. . . . . 76
6.6 Connect crossings in $V$ with the exterior crossings based on the Gauss code. . . . . 77
6.7 Connect crossings in $V$ to other crossings in $V$ for double coin knot. . . . . 77
6.8 Perturb the order of the exterior crossings along the boundary will create undesired intersection. . . . . 78
6.9 Folding links given arbitrary small clearance. . . . . 81
6.10 The method to avoid overlapping in Lemma 9. . . . . 81
6.11 The method to avoid overlapping in the first case. . . . . 82
6.12 The method to avoid overlapping in the second case. . . . . 83

7.1 Arranging a knot $7_1$ with Da Vinci robot arm. . . . . 86
7.2 The resulting configuration of a double coin knot so that the last five crossings are on a straight line. . . . . 86
7.3 Arranging a double coin with Da Vinci robot arm. . . . . 87
7.4 Rotating extremal segments to align all weaving crossings on a straight line. . . . . 90
7.5 Reconfiguration of a double coin knot to align type-2 crossings onto a straight line. . . . . 90
7.6 Arranging a double coin by laying out the type-1 crossings on the same height. . . . . 92
7.7 Arranging a knot $8_{10}$ by robot and human collaborating together. . . . . 93
7.8 Arranging a knot $9_{31}$ by robot and human collaborating together. . . . . 93
7.9 Arranging a knot $9_{32}$ by robot and human collaborating together. . . . . 93
7.10 A sheepshank unknot. . . . . 97

# Chapter 1

# Introduction

In this work, we discuss several robot knot tying approaches. As a special case for string manipulation, knot tying capability can push the limits of robot manipulation, which provide insights for general flexible object manipulation which is an important component of human-robot cooperation, and encourage the development of manufacturing and medical robotics.

There are several reasons for our choice of studying knot tying as a gateway to understanding string manipulation. First, we consider knot tying as a nontrivial task in string manipulation, especially considering the number of different type of knots. At the same time, the goal of knot tying usually is not a specified single configuration, but a collection of similar configurations with the same topology and relative positions. What is more, robot knot tying can be helpful in many human inaccessible scenarios for binding, such as in-body surgery, or hazardous environments where humans have to tie knots while wearing thick gloves and suits.

Knot tying has been studied mostly using traditional robot arms with comprehensive sensing. The few contact points provided by the robot arms leave the string highly under-actuated. Under-actuated deformable objects can deform into infinitely many different configurations. Modeling and prediction of deformation is challenging at the very least.

By applying different constraints on the string, we can force the string into a polygonal arc. The simplified model provides simple representation of the string, and leads to simple manipulation with minimum sensing or no sensing at all. Through several presented practical knot tying strategies, we hope to improve our understanding of knots, and understand more about string manipulation.

One of the most interesting string manipulation examples by humans is probably the game of cat's cradle. Even though the string is highly deformable, a human is able to form the string into many different shapes without ambiguity using a limited number of contacts. What is more, there is almost no re-grasping once the contacts have been made.

Humans achieve this level of control by stretching the string between contacts, reducing the deformation of the string to a minimum. Although string does not become knotted in Cat's Cradle, we expect the principle can be applied in the knot tying process.

When deformable objects are in contact with rigid bodies, their configurations can be changed by these rigid bodies. If these rigid bodies are carefully designed to affect the configuration of the objects in a specified manner, the deformation can then be predicted. Consider the cable that is wrapped on spool, the

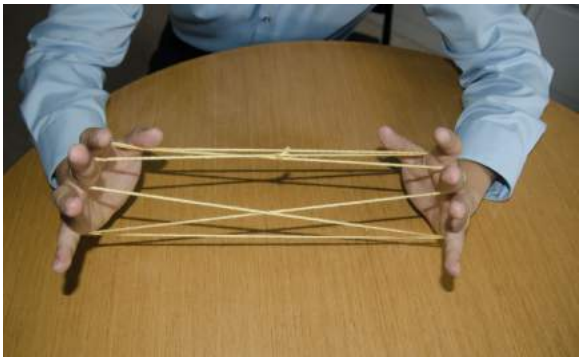


**Figure 1.1:** An example of the cat's cradle game.

configuration can be roughly estimated. Let us refer to these carefully designed rigid bodies as *fixtures*. Fixtures provide support for flexible objects, and have known configuration. Therefore, fixtures can be used to assist flexible object manipulation with limited or no sensing. With fixtures, simple controls can be sufficient to achieve complex tasks.

Working with deformable objects is very common in different areas of computer science. In computer graphics, simulating string and cloths is a common but challenging task. Computer vision also encounters deformable objects very often. Recognizing configurations of string and cloth is still difficult due to their deformable natures.

In these areas, strings are usually simulated and represented using a continuous curve or an approximation of such a curve (with many small links). In the past, roboticists have also modeled string using similar models.

However, in robotics and in this work particularly, we are trying to manipulate string to achieve a certain goal configuration rather than knowing every detail of the string configuration. Therefore, we have the luxury to *force* the string into a much simpler model–*a polygonal arc*–to complete the task.

Almost all of the knot tying techniques we present in this work rely on fully extending the string one way or an other to avoid deformation, effectively reducing the complexity of string manipulation from an infinite dimension problem. In fact, we eventually give a upper bound on the dimensionality of the knot tying problem for any given knot.

Using a limited number of contacts, we show that knots can be tied using several different approaches.

Our first approach is to use *fixtures* to stretch the knot geometrically, and to maintain the knot topology [BcfaKB14, WBB14, WBB15a, WB16b]. We believe this work to be the first generalized knot tying approach, building on initial work by Matthew Bell [Bel10]. We were able to use these fixtures to not only *arrange*—transform the string from *untangled* configurations to *knotted* configurations, but also *tighten* it into desired shapes. Some fixtures can even be used to tie knots to satisfy specified distance constraints between selected pairs of contact points, which we refer as *precise knot tying* [WB16b].

The arrangement fixture in this thesis encloses a tube that resembles the knot diagrams, sharing the same number and order of over- and under-*crossings*, the projected overlapping locations on the string. The string that is completely within the tube will be in the correct knotted configuration, if it can be extracted from the fixture without compromising the established geometrical and topological properties. The extraction of the

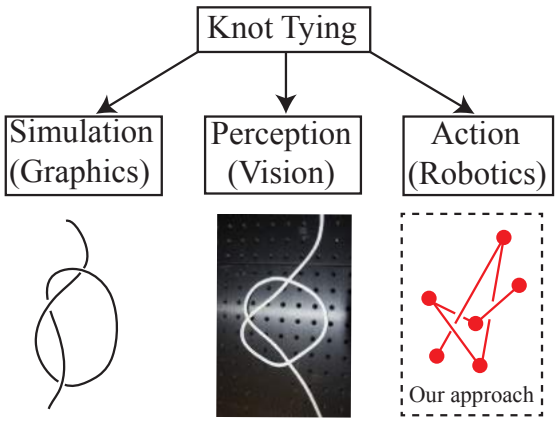



**Figure 1.2:** Different ways to model string in different areas due to different objectives.

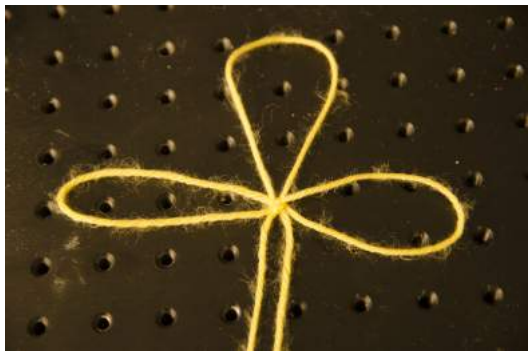
string relies on the correct cutting of the fixture into multiple pieces.

After arrangement, we choose to use one of the most simple control possible to tighten the knots: pulling the open ends of a knot. This simple control can be easily implemented using motors, but additional assists are needed to tighten knots correctly.

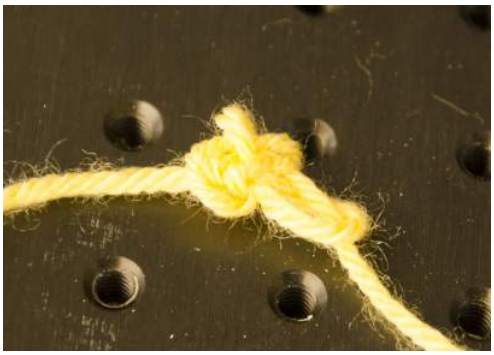
We consider a knot tightened with string locked by friction at different contact points. Simply pulling the open ends of a loose knot without any support may eventually achieve some friction locks–locations along the string where friction immobilize the string against external forces, but may not be able to achieve all of them. For example, consider the comparison shown in Figure 1.3, where a *cloverleaf knot* is tied by our fixture, and by hand pulling the open ends without any constraint. Without any constraint when pulling the open end, the knot looks nothing like the desired shape. The main reason is that the friction locks may not happen at the correct locations relative to each other, because some friction locks will happen prematurely, preventing other friction locks from happening. To avoid this situation, we again used fixtures to delay all friction locks, until the friction locks are all very close to the desired relative positions.

Several different fixture designs are presented along with the corresponding automatic design procedures. Each of the design was introduced to improve the capability of the fixture-based knot tying approach. From the most basic fixture design only capable of arranging knots, to embedding tightening fixtures into arrangement fixtures to automate the entire knot tying process, then to the improved tightening fixtures to allow *precise tightening*—tying knots with predefined distances between selected friction locks, each design was introduced based on the flaws observed in experiments with previous designs. Eventually, we are able to automatically tie knots that are very complex, such as decorative Ruyi knots with 64 crossings, while maintaining the predefined distances between selected friction locks.

The fixture based approach mostly depends on manipulating string while it is taut or near taut. In such conditions, the string is easier to model and to manipulate. We refer to this type of approach as a *passive* approach as the string is passively led into different configurations by wrapping around the fixtures. There are also string manipulation approaches that rely on the active control of the string to achieve different



**(a)** A cloverleaf knot tied by our fixtures, with three roughly equal sized bows.



**(b)** A cloverleaf knot tightened by pulling only on the open ends without constraint.

**Figure 1.3:** Comparison of two cloverleaf knots, one tied using our fixture, one tied by simple pulling on the open end, without any constraint.

configurations.

In this work, we have worked with many knots. We give a brief introduction of knots we have tied using fixtures in Table 1.1. Here, we just present the name and appearance of the knots we have tied using fixtures, without introducing the details of each knot. Some definition of knots are introduced in the later chapters.

Despite the success of passive knot tying approaches, a new fixture is required for each new knot, which is not as flexible as *active* control strategies–strings are directly controlled by grippers grasped at different locations on the string to reach different locations and form different configurations. Therefore, we also present a simple control strategy [WBB15b] that is first applied to thread string / rope through small openings.

The control approach sets up virtual vector fields as controllers, and calculates an *approximate Jacobian* [Ber13] to generate appropriate motion for the grippers, assuming the rigidity of string and motion of the string caused by a gripper diminishes as the distance to gripper increases. The vector fields are generated by sequence of virtual rings placed in the environments.

The Da Vinci surgical robot is a state of art cable-driven surgical robot. The Da Vinci robots are currently manually operated by humans to perform surgeries. Using a Da Vinci robot arm, several knots are tied autonomously around sets of rods fixed in space as a proof of concept, showing that knots can be tied around generic shaped supports with a dexterous robot arm.

Finally, we study the theoretical *tie-ability* of knots using a small number of grippers without re-grasping. The number of contacts needed to tie an arbitrary knot without re-grasping is first bounded in this work, as a first step to understand the complexity of knot tying. We intend to use this small number of fixed fingers as time-varying constraints to force the string into a sequence of line-segments, such that we can almost always compute the configuration of the string without too much trouble.

We first investigate how to grasp a knot using a small number of contacts, stretching the string between adjacent contacts, while at the same time avoiding any string self-intersection. We believe this is an important step in knot tying, since we cannot fold a knot if we cannot even grasp it under the same model without

introducing uncertainty. In this work, we refer to the string model of stretching between contacts while avoiding self-intersection as a *polygonal arc* model. However, please keep in mind that the number of links in our model is finite and relatively small, in contrast to the commonly used infinitesimal link polygonal arc approximation of curves.

Necessary and sufficient numbers of contacts for grasping arbitrary knot as a polygonal arc are presented. For the sufficient number of contacts, an algorithm is used to place those contacts. This algorithm is proved to be correct and can be applied to arbitrary knot types. Furthermore, we present and prove a naive and loose upper bound on the number of links required to fold an arbitrary knot, based on a sequential feeding motion sequence. The number of contacts derived from the analysis is quite large, usually exceeding the number of contacts one can provide with traditional robot arms.

When the open ends are grasped or connected, the knots are usually in a different topological class compared to a straight segment of string. In order to achieve the change in the topology, re-grasps need to be performed during knot tying. Re-grasps requires precise configuration information that is hard to acquire, particularly with deformable objects. Mis-grasps may even lead to total failure in manipulation.

In the last part of this thesis, we investigate the knot tying from the perspective of reducing the use of re-grasps. Inspired by mechanical looms and weaving, we develop a novel knot tying strategy that applies to both robots and human.

Through the analysis of the Gauss code, we derive an algorithm that can yield a sufficient number of re-grasps to tie a given knot. The generated number is usually small. In fact, for many knots, the sufficient number of re-grasps is 1.

We can divide the crossings on the Gauss code into two different categories. The formation of the crossings in the first category does not change string structure topologically, even when the ends of the string are grasped. Figures 7.1a and 7.3a show the arrangement of crossings in the first category for a decorative double coin knot, and the knot labelled as $7_1$ on the standard knot table[1]. The formation of the crossings of the second category changes the topological structure of the string. Theoretically, the crossings in the first category can be formed in parallel, while the crossings in the second category can only be formed sequentially.

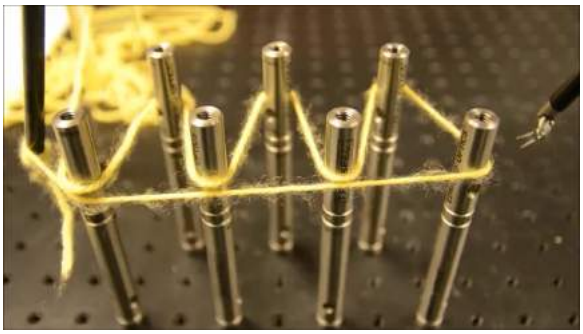
Our approach is to first lay out the type-1 crossings amongst a set of vertical rods, using a single motion without re-grasping. Then we use a simple weaving motion, either along a straight line, or with only vertical displacement to allow an alternating over-under pattern, to complete the type-2 crossings. Figures 7.1b and 7.3b show examples.

We conducted experiments with a Da Vinci surgical robot to show autonomous knot tying with a small number of re-grasps. In a second set of experiments, we used an Adept Cobra industrial arm to arrange the type-1 crossings, while a human arranged the type-2 crossings.

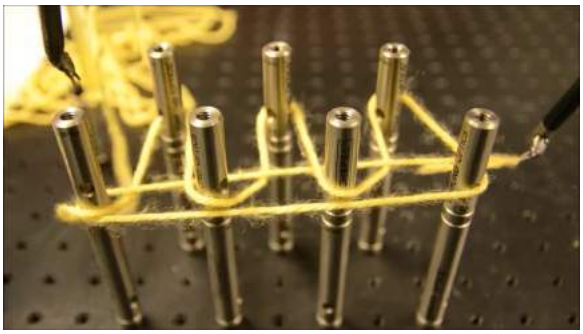
## 1.1 Potential applications

Even though this work does not focus on specific applications where the knots are tied using our methods, the extension from our work might be applied to different areas.

[1] An example of the standard knot table can be found at the following link, https://www.math.unl.edu/ mbrittenham2/ldt/table9.gif



**(a)** Arranging the type-1 crossings around set of straight rods.



**(b)** Completing the arrangement of a knot $7_1$ with one re-grasp.

**Figure 1.4:** Arranging a knot $7_1$ with Da Vinci robot arm.

With the simple control strategy and minimum sensing requirement with our fixture based knot tying, it is possible that the knot tying strategy can be utilized in hazard environments. For example, in a scenario with extreme cold or during a space walk when human has to wear thick gloves to tie a knot, our device can be used to assist.

Our fixture-based knot tying approach is also cheap and repeatable. In a factory setting where a knot needs to be tied over and over again, using our approach can be helpful. One of our future work includes extending knot tying to use materials like ribbons, so that we can tie gift bows automatically. Companies that needs to gift wrap their merchandise can benefit from this approach.

Our knot tying approach also focuses on speed and reliability, which is critical in surgical settings. Even though surgical robots such as Da Vinci robot arms have been widely applied in surgical settings, all tasks are accomplished with human aid. Suturing is one of the slowest steps in a surgery, which requires tying knots over and over again. Our approach also has the potential to lead to autonomous suturing.

## 1.2 Organization

This thesis is organized as follows.

In Chapter 2, we introduce mathematical knot theory as background and the foundation of our knot description. Then, work on knot tying and string manipulation in robotics is introduced.

Chapter 4 presents fixture-based knot tying, from arranging to tightening. Different phases of tying with different fixtures are discussed in detail. Different knots ranging from the simple overhand knot to the very complex Ruyi knot are tied using fixtures more and more complex fixture designs. These fixtures are designed with automated procedures, which are also presented in this chapter.

Chapter 5 discusses the active control strategy of string and knot tying. The design of vector field and the computation of an approximated Jacobian is shown. At the end of the chapter, experiments of Da Vinci robot tying knots around generic fixtures without sensing are shown, laying foundations for future work on fully autonomous knot tying around generic fixtures.

Chapter 6 investigates knot tying from the theoretical side, intending to define the complexity of knot

tying based on the number of contacts needed to tie the knot. From knot grasping to knot folding, different contact placement and motion strategies are presented.

Chapter 7 compares knots with weaving, and simulates the motion of a loom to tie knots. Through analysis of the Gauss code, an algorithm is derived to find a small sufficient number of re-grasps to tie a given knot. A robot-human collaboration in knot tying is shown to demonstrate the novel knot tying approach.

Finally, conclusions of the work are presented, followed by proposed future work.

| Knot type | Knot example | Knot type | Knot example |
|---|---|---|---|
| Overhand Knot |  | Square Knot |  |
| Bowline Knot |  | Sheet Bend |  |
| Strop Bend |  | Harness Bend |  |
| Carrick Bend |  | Shoelace unknot |  |
| Double coin |  | Cloverleaf knot |  |
| Sounding line |  | Ruyi knot |  |

**Table 1.1:** Examples of knots tied using fixtures

# Chapter 2

# Related work

We are not aware of any prior work on fixture based knot tying through our literature search. Our knot tying approach draws inspiration from different studies in topology, geometry, and related deformable object manipulation in robotics.

*Knot theory* [Ada04a, Aex23, Lic97, Rol76, Arm83, CF77, Liv93, Man04] is dedicated to study the knot invariants, which identify whether different drawings represents the same knot. One most common way to describe a knot is through the use of a knot diagram. A knot diagram is the drawing of a knot *regular* projection onto a plane (usually on $x$-$y$ plane), with broken lines indicating some string segments under-crosses some other segments of string. A knot projection is called a regular projection if no three points on the knot project to the same point, and no vertex projects to the same point as any other point on the knot [Liv93]. Based on different projection direction, and different geometry of a knot, the projected knot diagram can be very different.

On the knot diagram, we can identify a collection of *cells*, an open bounded connected region of $\mathbb{R}^2$ enclosed by the knot diagram which is a subset of the complement of the knot diagram. This definition of cells is different from the definition of cells in CW-complex [Hat02, Whi49a, Whi49b]. Cells can be identified on all knot diagrams. Then, we can define the *knot topology* as the number, the ordering and the relation between different cells. Some researchers proposed to study the knot diagrams as a collection of 1- and 2-cells (complex cells) [Adl14].

*Mathematical knots* are the ones that do not have open ends and cannot be tied or untied, which is different from the *physical knots* (referred as knots for simplicity) we attempt to tie in our work. Knots with the same knot topology can have different geometric properties. Our fixture approach primarily focus on achieving the desired knot topology before approximating the geometry.

However, there is an relative new subject area called *physical knot theory* dedicated to study the phenomena of knots and how they hold together [Kau91, Kau95, O'H03, L.98]. Physical knot theory is the study of mathematical models of knotting phenomena, often motivated by physical considerations from biology, chemistry, and physics [Kau91].

*Reidemeister moves* are local changes to the knot diagrams that will lead to different drawings of knots without changing their topology [Ane99, Rei27]. Three are three different types of Reidemeister moves in total, usually referred as twist (untwist), poke (un-poke) and slide (un-slide). Different studies has shown different possible sequence of Reidemeister moves that can transfer one knot drawing to another [Tra83,

Hag06, Öst01, Man04]. Reidemeister moves have also been used to study how to *split a link* [Cro04, Lic04], and untie *unknots* [HL01, HN10, HH10, HK10], where an unknot is a special type of knot that is topologically equivalent to a geometric circle, and can be untied by a sequence of Reidemeister moves, such as the top of a shoelace.

When is a knot really tight? When different materials are used to tie knots, how tight will the knot become? The study of the tightness of a knot has be conducted in applied mathematics, physical knot theory and physics [ACPR11, BPP08, MCS05, Raw98, Kau91, Sim01]. In [ACPR11], the authors introduced a term called *rope length* to help measuring the tightness of a knot, which embeds the thickness of the string into the measure. The authors model the string as a polyhedron with each string-string contact as a vertex.

Physically, a knot is tight because of *friction locks*. Friction lock happens at string-string contacts which prevents the sliding of the string that may potentially change the contacts along the string. Contact friction is difficult to analyse, so our fixture based knot tying approach attempts to delay any contact that may potentially lead to friction.

Previously, different knot tying devices have been invented [BF06, Cha04, Sin04, sho]. However, each design was specific to one type of knot. In our work, even though fixtures are specified for each knot, the fixture design process can take any knot diagram as input and output the corresponding fixture design. The fixture can then be extruded based on the design using any 3D modeling software. Therefore, we consider out approach of knot tying a generalized approach that can be applied across a wide range of knots.

In robotics, knot tying has been studied previously, but mostly using active controls to directly interact with string to reach various designed configuration. Inoue and Inaba used a 6+1 DOF robot arm equipped with stereo machine vision to tie knots [II85] around a ring. A graph based language was developed by Hopcroft *et al.* to program knot tying motions, and tested through knot tying with a robot arm [HKK91]. Real and simulated knots were able to be tied using motion planning algorithms developed by Saha, Isto and Latombe with a string model [SI06, SIL06, SI07]. Sequences of motions were constructed and carried out by a pair of robot arms. They introduced a hierarchical decomposition of the knot using loops as basic structure. In sequence, subtasks were achieved using trajectories generated by motion planning algorithm. Wakamatsu, Arai and Hirai developed a detailed description using a set of four basic operations for transitioning between states to describe the knot tying and untying with robots [WAH06]. Tree search is used to build a planner to find a sequence of motions to tie or untie a knot.

Fixtures were used as manipulators to fold cardboard cartons by Lu and Akella [LA99, LA00]. Motion planning algorithms were developed to generate a folding sequence that could fold the carton blanks into a box using fixtures. Recently, fixtures that contain paths designed inside to guide ribbon cables to desired locations with limited bending and twisting were used in cancer treatment [PPAG14]. The paths inside are planned to increase the coverage of tumor volume.

In our approach, fixtures are used to cage the possible deformations of the string. Similar to caging rigid bodies [BMAP01, RB95, RB96, RB99], our fixtures do not attempt to immobilize the string, only to limit the possible motion of the string in certain directions. Local motions that do not affect global properties, such as the knot topology, are ignored. When strings are taut, the string cannot move towards the unconstrained directions. Such design simplifies the fixture, while still maintaining good control over the string.

We initially used separable multi-piece fixtures to arrange knots. The separation of such fixtures, however, is not trivial. It is similar to extracting casts from corresponding molds. These approaches allowed non-planar parting surfaces [RS90, Che97]. Khardekar *et al.* developed an algorithm to compute a feasible

parting direction for two mold pieces [KBM06] with real-time highlighting of undercuts. More molding pieces were considered later. Chen gave criteria for identifying parting surfaces [CR03], where a reverse glue operation was used to produce multiple mold pieces.

In our early work to study the design of fixtures used to arrange the knots [Bel10, BcfaKB14], graph drawing algorithms [Tam87, GT96] were used to optimize the layout of the knot inside the fixture. Through experiments, we found that the layout of the knot should balance between the shortest length and the least amount of turns, due to our physical capability of pushing the string though the enclosed tubes of the arrangement fixture in the existence of friction.

During tightening, we intend to delay the friction locks from happening prematurely, but at the same time, we are also approximating the geometric shape of the knot that we are trying to tie into [WBB14]. The shape of the knot wrapping around the fixture can be computed if all segments of the string are taut. The computation of such shapes has been studied in the computation geometry community as the shortest curve in the same homotopy class among a collection of point obstacles [Bes03, EKL06, GS97, GS98, HS94]. By knowing the shape of the knot wrapped around the fixture, we can design the fixture to better approximate the desired geometry. Recently, we not only tighten the knots using fixtures [WBB14, WBB15a], but also tighten knots satisfying specific distance constraints between specified contact points [WB16b]: *precise tightening*.

Friction exists between the contacts of string and fixture, and even the contact between different segments of the string. One could analyse the contact mechanics and forces [BTG02, Mas01, BET04] to quantify the friction. Part of the tightening fixture consist of a collection of rods. The friction between the string and the rods can be analyzed using capstan equation [Att99, GJS02]. To simplify, we assume the minimum friction between string and fixture by mounting all movable parts of the fixture on low-friction ball bearings, following a direction explored by Furst and Goldberg [FG92]. At the same time, we carefully design the fixture to avoid excessive string-string contacts.

Linear deformable objects (LDOs) and string have been studied by many researchers, and many different models have been proposed to characterize their properties. A detailed model for deformations of linear objects based on differential geometry has been developed by Wakamatsu and Hirai [WH04], which is used to plan paths for LDOs. Wakamatsu *et al.* also proposed a 2D model to capture the behaviors of LDOs in contact with obstacles which extends to the deformation of LDOs under external forces and moments [WYT+06]. Rods have also been frequently used to model LDOs. Cosserat rods were used to simulate thin strands by Pai [Pai02], who developed a fast simulator for real-time interaction with a virtual suture. Elastic rods, which have been studied in the context of classical mechanics [LS96], recently have been used to study the equilibrium configuration of LDOs [BM13] using a Pontryagin-type of formulation.

To control LDOs, one can take an "active" approach trying to actively maneuver different segments of the deformable object, like the study of the control of snake robots (many link arms) [RBC12]; or we can take a "passive" approach to enforce different segments of the deformable object into desired configuration with the assist of contacting external stationary objects like a pulley. Henning, Hickman and Choset worked on motion planning for serpentine robots serves as an example, and discussed several related work [HHC98]; Degani *et al.*'s work on tube traversing manipulators gives another example.

String manipulation has been explored in many directions in robotics community in industrial [HW00a] and surgical [KW01b, KP10] contexts. A needle that can bend, or spin around the central axis can be used to navigate string in silicon type of semi-solid environments [ALG+05, AGP+03]. String manipulation

using robot arms in collaboration has also been studied [II85, KNMB00]. String manipulation has also been studied in the context of suturing. A spline model formed from linear strings that can simulate real ropes (also supports collisions) was developed by Phillips *et al.* [PLK02]. Some suturing systems have been mentioned by Taylor in his survey of medical robots [TS03]. Kang and Wen developed one of the suturing systems called EndoBot [KW01a], which includes algorithms for autonomously tying knots during suturing and a modified shuttle needle device for robots. Some other suturing work can be found in [WWDL10, RCR+11, HIIH04].

Many of the deformable object manipulation tasks require a detailed model or simulation of the deformable objects [FSAB11, RLA06a]. Motion planning approaches have been used for deformable object manipulation [MK06, SI06, SIL06]. Policy learning from demonstration has also been investigated [MTO+03, RSBH12, SGV+13, SHLA13].

Manipulation of deformable objects beyond LDOs is also relevant. Motion planning of deformable objects among deformable obstacles by Rodríguez *et al.* [RLA06a] maintains constant volume for all objects. Recently, Phillips-Grafflin *et al.* studied similar motion planning problem in the environment of all deformable objects trying to achieve minimum deformation among all objects without simulation for deformation [PGB14].

We recently studied the task of how to insert string into tight tolerance loops in workspace [WBB15b] hoping that it could be the foundation of the knot tying that we are trying to explore using active open loop strategies. We proposed to achieve such high-precision insertion tasks using a sequence of virtual magnetic fields generated by current-carrying circular wire loops as controller for the string to follow. The method we proposed was similar to the work in visual servoing for deformable objects [HW00b, NALRL13, SP09, WHKK01]. In a recent study, Haddadin *et al.* [HBAS11] also used magnetic fields for control.

The insertion tasks resembles the classic robotics problem " peg-in-hole" [Don90, LPMT84], especially when there is only one ring to insert into. Instead of detailed analysis and back chaining for the "peg-in-hole" problem, we focused on an online controller that is intended to be robust to deformation of the string and errors. Our approach is able to drive the string into the target tight-tolerance space even after failed attempts, due to the directed integral curves that penetrate the target area from the correct direction.

Sequential controllers are often used to navigate complex dynamical systems, and deformable objects can be viewed as a complex dynamical system. Sequential motions can be generated using fields [BRK99], or sequence of funnels that could reach goal regions [Ted09, TMTR10].

Although we are not aware of any other generalized fixture based knot tying approaches, and the understanding of manipulating string using fixtures has just begun, this approach is inspired from many different studies in manipulation and motion planning. Keeping the string in the same homotopy class as the originally laid out shape on the fixture during tightening is critical for out approach. Many motion planning approaches decompose or explore the configuration spaces based on different classes of homotopy paths. Recent work by Bhattacharya *et al.* explores algorithms to generate and identify different paths in the same or different homotopy classes [BLK12, BLK11, BLGK13]. Motion planning for flexible bodies in minimum energy configurations inspired our approach to manipulate the string when nearly taut [MK04, RLA06b], as did work on robot origami folding [WD09, BM08].

One of the fundamental challenge in string manipulation is the modeling. Traditionally, string is modeled as sequence of rigid links with spherical joints. Most of theses models require many links to simulate a string. However, we intend to model the string as a small number of rigid links in sequence in the study of parallel

knot tying. By doing so, we hope to understand more about the complexity of the knot tying problem, and come up with simpler approaches to manipulate the string.

When the string is modeled as a small number of links, the study of the wrapping and unwrapping between straight line configuration and some convoluted configurations is similar to the study of carpenter's rule in 3D. In the past, problems of carpenter's rule in 2D has been studied extensively [ADG09, CDD+10, CDD+08, CDIO04, CDR00]. Recently, new progress has been made to approach carpenter's rule in 3D [BDD+99].

In robotics, carpenter's rule has been applied to study the motion of closed or open chain of multi link arms. For example, Trinkle *et al.* [TM02] analyzed the motion of such close chain arms, and proposed algorithms to compute such paths. This work initially did not consider obstacles. The authors further developed algorithms to deal with obstacles and other class of closed chains [LT05, SSLT07]. Han Li *et al.* has worked on finding paths between arbitrary polygonal loop configurations, simulating multi link robot arms.

One of the central ideas of this work is to apply different type of constraints to simplify controls for knot tying. The idea was not new and has been applied to other different scenarios. For example, people use specifically designed devices to assist rehabilitations, and there is a entire field of studies on how to design and improve the performance of robot assisted rehabilitation; more detains can be found in related surveys [MEGH+14, DiY14]. These devices comes with different constraints to guide the limbs to achieve specific motions. Constraints can also be used to generate appropriate motions and forces for manipulation tasks [PLPC04, THL+02]. Passive-dynamic walking systems are also designed with numerous physical constraints [GRC98, McG90a, McG90b, McG91] to derive simple walking strategies.

# Chapter 3

# Mathematical knots

Many of the conventions and the notations we use in this thesis are inherited from *mathematical knot theory*, which is a sub area of topology dedicated to study the *mathematical knots*.

A mathematical knot is an equivalence class of embeddings of a topological circle $\mathbf{S}_1$ into three dimensional Euclidean space $\mathbf{E}^3$ [Ada04b, Arm83, CF77, Cro04, Liv93, Man04]. This notion of the knot is different from the knots we use in our daily life that have open ends and can be tied or untied. We refer the knots with open ends as *physical knots* in this work, and will drop the work "physical" for simplicity if there is no ambiguity.

To describe and identify knots, we usually project the mathematical knots onto a plane, and use the projected diagrams. Such diagrams are called *knot diagrams* if they are the result of a *regular projection*.

**Definition 1** *[Liv93] A knot projection is called a regular projection if no three points on the knot project to the same point, and no vertex projects to the same point as any other point on the knot.*

On the projected *knot diagram*, broken lines are used to indicate where one part of the knot under-crosses the other part of the knot that is directly "above" the broken lines. Such locations are called *crossings*. Each crossing is labeled by a unique number, indicating the order of the appearance when tracing along the string.

For example, consider Figure 3.1, a shoelace unknot diagram is shown, and the crossings are labeled with numbers.

On the plane of the knot diagram projection, the projected knot diagram separates the plane into several disconnected closed regions. We call these regions *cells*. In Figure 3.1, cells are also labeled, with capital letters.

These cells are critical structures to maintain the topological structure of a knot. These cells are the main structure we try to preserve.

Apart from the knot diagram, since we have the labels for the crossings, we can also use the labels to describe the knot. One of such description is called *Gauss code*, which is a sequence of crossing labels where each label appears twice. The Gauss code is generated by tracing along the string on a knot diagram. In this text, we will use numbers to label crossings in the Gauss code, and a superscript "+" or "-" to indicate an over-crossing or an under-crossing. For example, an overhand knot with Gauss code $G = \{1^+, 2^-, 3^+, 1^-, 2^+, 3^-\}$ has 3 crossings, and the length of $G$ is 6 ($|G| = 6$).

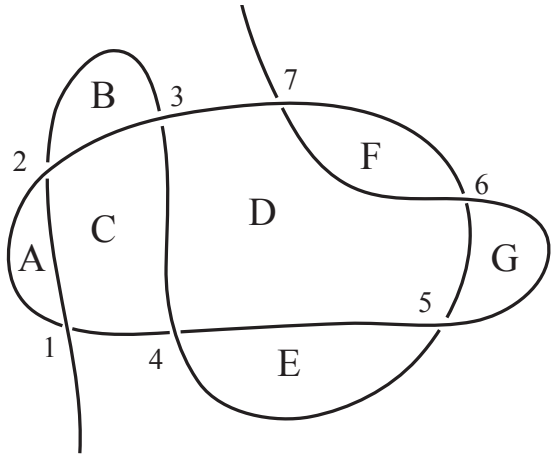



**Figure 3.1:** A knot diagram of a shoelace unknot, with all the crossings labeled with numbers, and all th cells labeled with letters. The Guass code for the shoelace knot is: $1^+$, $2^-$, $3^-$, $4^+$, $5^-$, $6^-$, $7^+$, $3^+$, $2^+$, $1^-$, $4^-$, $5^+$, $6^+$, $7^-$.

**(a)** Shoelace unknot.

**(b)** Culprit unknot, redrawn from [HK10].

**Figure 3.2:** Knot diagrams of unknots.

If a knot can deform (*ambient isopoty*) into a geometric circle, then it belongs to a special subgroup of knot, called *unknot*. Not all unknots are easy to identify. In Figure 3.2, the top of the shoelace knot (referred as *shoelace unknot*), and a culprit unknot are shown. The culprit unknot can be transformed into a circle without breaking the string, but a complex sequence of Reidemeister moves needs to be applied.

The knots we consider are the result of cutting at one location per topological circle on a mathematical knot (or unknot). To better illustrate the difference, let us consider the example in Figure 3.3. We can see that an *overhand knot* is achieved by cutting the *trefoil* knot.

While most of the knots we consider in this work are composed of a single strand of string, there are a few experiments arranged multiple strands of string into a "knotted" configuration. Formally, these structures are called *links*. We briefly show that our fixture based approach can be extended to links with several examples, but the main focus of this work is still knots tied with single strand of string. Even though we did not conduct experiments showing the tightening of links using our fixtures, the principle of tightening can easily be extended to links.

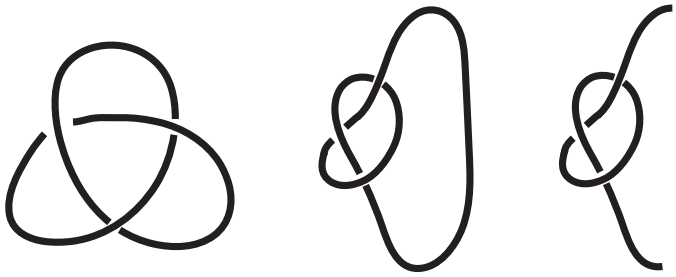


**Figure 3.3:** A trefoil knot is cut into an overhand knot. From left to right, one drawing of trefoil knot, a different drawing of trefoil knot, and an overhand knot.

# Chapter 4

# Fixture based knot tying

In this chapter, we focus on tying knots using fixtures—a set of tools that contact the string and hold its position, such that knots can be tied with minimum re-grasping. This approach requires no sensing and uses simple controls.

One of the major concepts we would like to convey in this chapter is the separation of *arrangement* and *tightening* throughout the knot tying process. Just like human cannot produce a fully tightened knot directly from a strand of untangled string, the robot knot tying should be separated into different phases. A knot must be arranged into roughly the correct shape, obeying the topological structure of a knot, before it can be tightened into an object holding its configuration by friction. In this chapter, the two phases are separated clearly, both in fixture design and the control strategies used to accomplish the task.

All fixtures used in this chapter are designed automatically using presented procedures, and extruded by the author using 3D modeling softwares and eventually prototyped using rapid prototyping machines. The design of the fixtures progressed as the knots we intended to tie became more and more complex.

## 4.1 Four-piece fixture

The first challenge of knot tying is how to transform the string from an untangled configuration to a knotted configuration, with desired over- and under-crossings along the string in specified order.

Let us consider mapping a knot diagram on the $x$-$y$ plane back into a 3D curve, with all over-crossings having the same $z$ coordinates 1, and all under-crossings have $-1$ $z$ coordinate. This 3D curve can always be fully contained by a cube. If the cube can be cut into pieces so that no parts penetrate the curve during the separation, the knot can then be extracted without compromising its structure. Here, by structure, we mean the topological structure of the string when the open ends are grasped or connected.

If such cut of the cubic fixture exists for a given knot, we can achieve the arrangement of the given knot by pushing a strand of string through the fixture following the tube carved inside the cube along the 3D curve; and extract the knot by separating the fixture. Since during separation, no parts penetrate the curve, the string along the curve can remain in the knot shape, thus a knot is achieved. The question is, how to cut the fixture and how many pieces do we need to cut the fixture into?

The answer turns out to be four, instead of depending on the complexity of the given knot. The following theorem was proved by Matthew Bell in his thesis [Bel10].

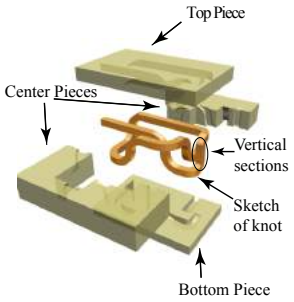



**Figure 4.1:** An example of a four-piece arrangement fixture for an overhand knot.

**Theorem 1** *[Bel10] Given any (mathematical) knot or link consisting of one or more strands of string, and described by a Gauss code, a fixture can be constructed that loosely arranges string into a knot with the same Gauss code, provided that the endpoints of the string are connected together outside of the fixture. Furthermore, this fixture can be cut into four pieces in such a way that all four pieces can be removed by pure translation without interfering with the string, in the sense that if the initial string configuration were treated as a rigid body, no interpenetration between the string and fixture occurs during fixture separation.*

The proof of the theorem can be found in [Bel10]. An example of a four-piece fixture for an overhand knot is shown in Figure 4.1. The top and the bottom pieces cover the segments of the knot that are on top or on bottom of other segments respectively, with vertical transitions between top and bottom layers. One of the vertical sections/transitions is also labeled in the figure. Two middle pieces are used to enclose these vertical transitions. The challenge is to arrange the vertical transitions to not overlap along a particular direction, which is perpendicular the direction that the center pieces are separated along. Knots can be extracted after the separation of the fixture, and these fixtures can be designed by following the constructive proof in [Bel10].

To physically test the capability of the approach, several fixtures are designed by the author. Three dimensional model of the fixtures are extruded using 3D modeling software (Solidworks) following the design, and prototyped using a Stratasys Objet Eden rapid prototyping machine.

The author chose to use pressurized air to push the string through the tube inside the fixture for arrangement. The first consideration is that the pressurized air is a relative simple control that does not vary for different knot fixtures. The second consideration is speed. With a fixture of limited size (true for all fixtures for knots listed in Table 4.1), the pressurized air can push each strand of string through the fixture within half a second.

Originally, we expected the friction between the string and the fixture could affected the insertion, but the use of pressurized air actually limited the effect of friction. The air flowing inside the tube forms an air cushion, letting the string float inside the air, limiting the contact between the string and the fixture, thus helped the insertion task. We expect this effect only to be applicable to the strings that are relatively light, such as the yarns and nylon strings we used in our experiments.

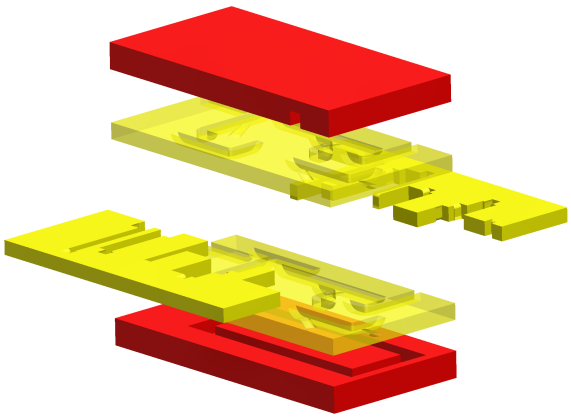


**Figure 4.2:** The Carrick Bend fixture designed with multiple pressurized air inputs. The red plates are additional layers that serve as a manifold that distributes air through the fixture.

We designed fixtures for seven different knots, ranging in complexity from the simple overhand knot (3 crossings) to a *Carrick Bend* knot (8 crossings), and conducted experiments to arrange all seven kind of knots. Most of these knots tie two pieces of string together. Human-tied versions of the knots (because our fixtures arrange string in the desired topology, but do not tighten them), as well as the fixture tube shapes (different colors represent different strand of strings), and the success rates over 100 insertion trials per fixture are shown in Table 4.1.

As seen in the table, most knot fixtures demonstrate a very high success rate, with the exception of the Carrick Bend, the most complicated knot we attempted to arrange using four-piece fixtures. Fixtures other than the Carrick Bend have tube cross-section lengths between 7 and 8 mm, and overall fixture dimensions of less than 15 cm by 8 cm by 5 cm. Due to the otherwise large dimensions of the Carrick-bend fixture ($>$ 20 cm along one side, the maximum envelope size of the 3D printer), we chose a cross-section length of 4 mm for the Carrick Bend.

We believe that limited success with the Carrick Bend is due to the smaller cross-section of the tube (causing increased turbulence and pressure drop along the tube), to a smaller slider, and to increased air loss along fixture seams in longer tubes. In fact, the success rate with the Carrick Bend was initially close to zero due to these problems; the 81% shown in the figure is for a modified fixture with eight additional air inputs along the top and bottom sections of the tube. This implementation is shown in Figure 4.2.

The Carrick Bend certainly seems to be at the limit of complexity of knots that can be tied with our current approach using pressurized air for insertion. The simpler Harness Bend exhibits some similar difficulties along the more complex of its two tubes.

From the dropped success rate of the Carrick Bend, we observed that there are also disadvantages of using the pressurized air. To reduce the pressure drop and maintain the air cushion during the insertion, the fixture should be relatively tight to limit the air loss during the arrangement. Even with relative good seal, the pressurized air still has a physical limit of how complex a knot can it arrange.

Extraction of the knot is simple using a four-piece fixture. We designed a mechanism composed of four-

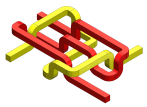
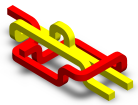
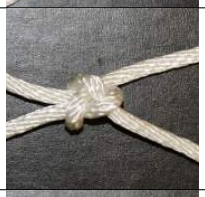
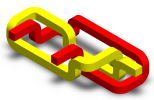
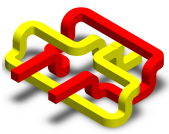
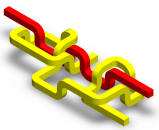
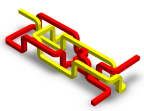
| Knot type | Knot example | Tube shape | Success Rate |
|---|---|---|---|
| Overhand Knot |  |  | 99% |
| Square Knot |  |  | 99% |
| Bowline Knot |  |  | 100% |
| Sheet Bend |  |  | 100% |
| Strop Bend |  |  | 100% |
| Harness Bend |  |  | 95% |
| Carrick Bend |  |  | 81% |

**Table 4.1:** Knot example, designed tube shape, and corresponding success rate of 100 trials for each knot fixture. Different string colors represent different strands of string. Data from [BcfaKB14]. A success means all strands of string is pushed through the corresponding tubes.

bar linkages with a single degree of actuation, shown in (Figures 4.3 and 4.4). The design is inspired by the mechanism used to open tackle boxes that fishermen store fishing equipment in. (Figures 4.3 and 4.4). Using a simple Dynamixel servo motor, opening the fixture takes about 1.5 seconds. Sealing the fixture with the current mechanism design so that pressurized air can be used to insert string remains a challenge.

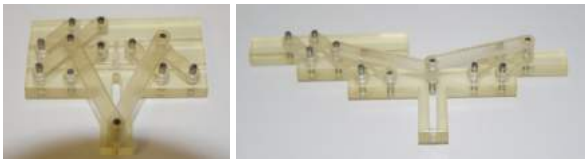


**Figure 4.3:** The opening mechanism for the four-piece fixture.

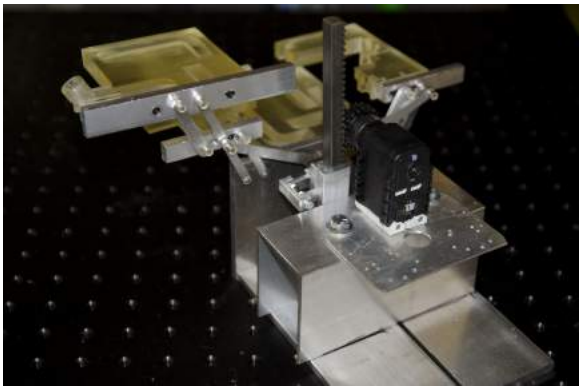


**Figure 4.4:** Machined complete fixture opener with fixture components attached.

The automated design does not tell us all the details about how to build the physical devices. We have talked about the size of the cross-section, which have a big impact on the success of pushing string through the device using pressurized air. Currently, all the turns of the tube inside the fixture are near 90 degree. A much sharper turning angle may lead to failure.

In this work, we are mostly focusing on the automated design of the devices. The turning angle for the tubes inside the arrangement fixtures are fixed around 90 degrees, but we experimentally found the best cross-section size for the tubes. We did not consider other unified metrics that exist across all fixtures designs, so that different fixture designs can be compared. It could be a very interesting direction for the future work to define other metrics in the fixture design, further optimize the fixture based knot tying approach.

### 4.1.1 Pulling vs. pushing

For even more complex knots, such as the River Knot shown in Figure 4.5, the pressurized air approach described above is infeasible using a practically sized fixture. We may need to some slower approaches that are not affected as much by the size of the fixture.

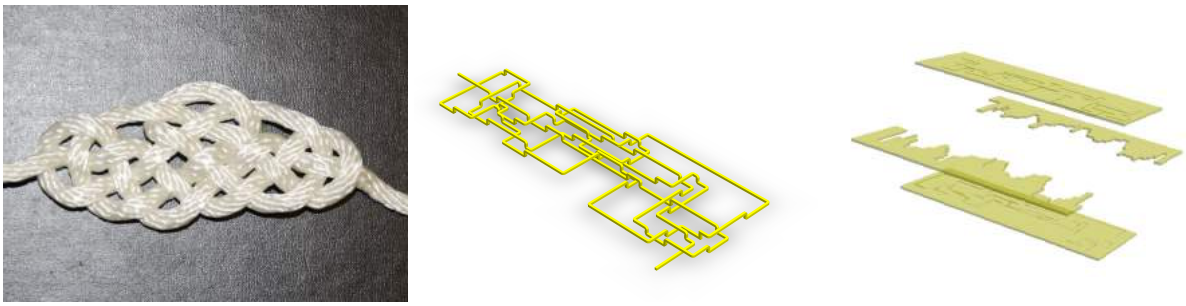


**Figure 4.5:** Human-tied river knot, and (unimplemented) four-piece fixture design.

We have therefore experimentally begun to explore the capabilities needed to pull string through a fixture tube. We have built a test structure, one module of which is shown in Figure 4.6; additional modules can be attached to lengthen the snake-like tube. This structure is not specific to any particular knot, but is rather used to test how many sharp bends a string can be impelled through using pressurized air. We consider three cases: forcing the string through the tube with pressurized air (and a slider), pulling the string through the tube, and pulling the string along a tube with ball-bearing rollers attached at turns.

Each modular section of the test structure contains a tube of length of about 47 cm. The tube has square cross-sections with edge length 7.5 millimeters. Each modular fixture contains twelve 90 degree turns.

With pressurized air, the string attached to a slider traveled through 150 cm on average over ten trials, with a maximum travel distance of 165 cm.

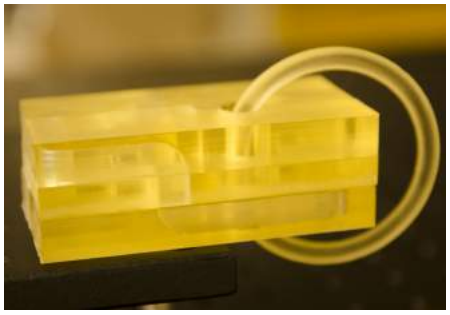
Pulling string without pressurized air was much less successful, demonstrating the strong positive benefit of the air cushion generated by air flow. Friction prevented motion of the string completely after about 25 cm; further force breaks the string.

Most of the friction force occurs at turns. We therefore installed rollers at each turn to reduce the friction and further test the pulling mechanism. The force required to pull the string through a tube of 190 cm is roughly equal to that required to lift a weight of 40 grams.

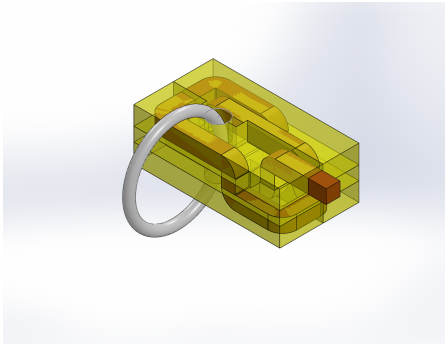
In spite of the promising results of using rollers to reduce friction of pulled string, the increased complexity of the fixture design is a disadvantage, and would presumably lead to larger fixture sizes. It is for this reason that our explorations make use of pressurized air.

Even though theoretically we can arrange any knot with the four-piece fixture design, the physical limits prevent us from arranging interesting knots such as River knots. It is clear that advanced mechanisms are possible to push the limit of our fixtures further, the resulting devices are no longer easy to build. The most promising approach would be the use of multiple pressurized air inputs along the tubes to push the string through the fixture. Due to the limited lab equipments, we are unable to test the River knot fixture, which is both too large to prototype or machine, and requires more channels of air inputs than we can provide in the lab.

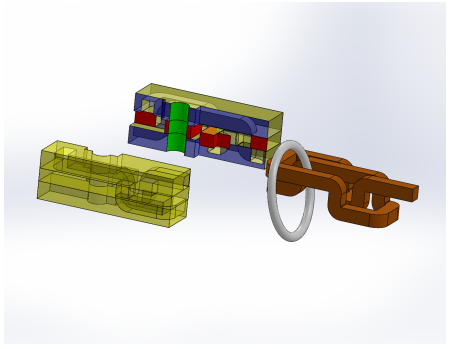
**Figure 4.6:** The modular test structure with rollers (silver ball bearings) installed at every turn.



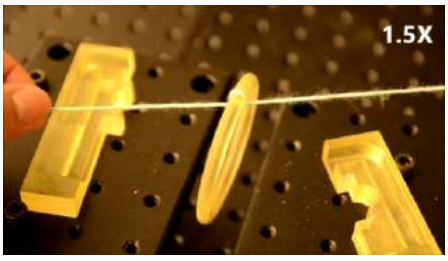
**(a)** Six piece fixture to arrange an overhand knot around a ring.



**(b)** Model of arranging an overhand knot around a ring.



**(c)** Disassembled arrangement fixture, exposing the string (orange tube) around the ring. The cut surfaces are colored blue and red for different layers, while the surrounding cylinder is green.




**(d)** Overhand knot tied around a ring by fixture.

**Figure 4.7:** Arranging an overhand knot around a ring.

### 4.1.2 Arranging knots around objects

In many real life scenarios, the knots are tied around objects as a binding mechanism. Further extension can be made with the four-piece fixture design to allow the knots to be tied around different objects. Topologically, perhaps the most interesting object to tie string around is a ring, since the top and bottom pieces of the fixture cannot be extracted from the ring without cutting the fixture.

For simplicity, we find a vertical bounding cylinder (colored green in Figure 4.7c) around the ring (or other object), and embed the bounding cylinder into the fixture. The middle layer of the four-piece fixture is cut into two pieces to allow the fixture to be separated without intersecting the vertical sections of the string. If vertical cylinders are placed along this *cut surface*, which is colored as blue or red in Figure 4.7c for different layers, the middle layer of the fixture is then separable without intersecting the cylinders. However, the cylinders are longer than the vertical sections of string, and extend into the top and bottom layers of the fixture. The top and bottom pieces also need to be cut along the cut surface for separation without intersection, extending the fixture from four pieces to six pieces.

Using the principle we stated above, the six-piece fixtures can be used to tie knots around various objects. We will not further explore this direction as we believe the proof of concept with the ring is sufficient to support our hypothesis.

## 4.2 From arrangement to tightening

Now that we can arrange knots using fixtures, another issue presents: how to tighten them after the arrangement? Most of the knots, whether for binding or decoration, need to be tightened to serve the purpose.

The knots come out of the arrangement fixture presented in previous section are loose, due to the separation requirement of the fixture. To transform the knot from such loose configuration, which usually do not resemble the desired geometry to a tied goal configuration, we need to gradually control the knot to approximate the desired geometry over time.

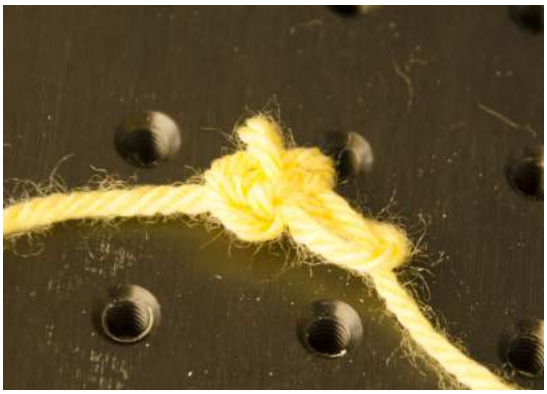
We usually consider a knot "physically tied" only when the knot has been tightened to the point that the application of certain forces, with bounded magnitude and constrained directions, does not further change the location of any string-string contacts when measured along the string. Typically, we expect such *friction locks* to appear at the end of the tightening process, as such friction locks prevent further motion of the string contacts.

Even though the knots are often tightened by human pulling the open ends of the string, re-grasps need to be performed at many contact points to maintain the desired geometry of the tightened knot. Robot arms are not yet able to perform such tasks with ease.

Therefore, we consider a simpler strategy: only pulling on the open ends of the string to tighten the knot while relying on some fixtures to guide the knot into its goal configuration. The main idea we make use of is to build tightening fixtures that delay string-string contact during tightening until the knot is "tight enough" to achieve the desired friction lock.

The choice of the control comes from the observation where simple pulling on the open ends is sufficient to tighten some simple knots, such as overhand knot or figure eight knot. However, simply pulling on the open ends without any constraint is not sufficient when the knot becomes more complex. For example, in Figure 4.8b, a cloverleaf is tied by simply pulling on the open ends without any assists, and the result

**(a)** A manually tied cloverleaf knot by carefully maintaining the knot structure through re-grasping.



**(b)** A cloverleaf knot tightened incorrectly by pulling the ends of the string without fixtures.

**Figure 4.8:** Two cloverleaf knots tied by hand.

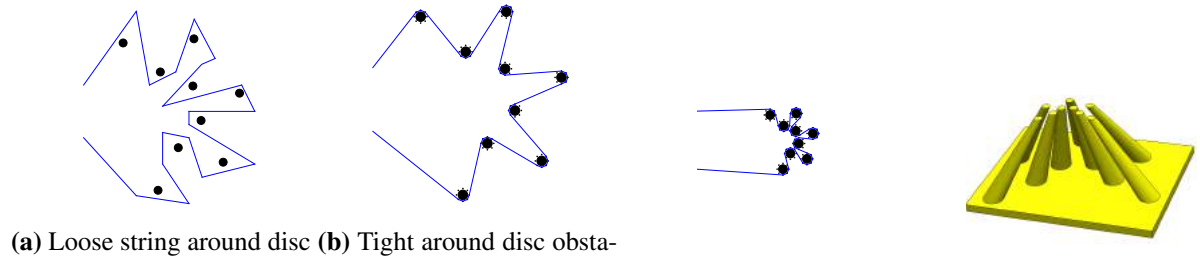
looks nothing like the desired shape shown in Figure 4.8a. Therefore, we need to use fixtures to assist the tightening.

When the perimeter of a cell on a knot approaches zero, one of the two things may happen, a Reidemeister happens and the cell degenerates, or a friction lock happens. If the cell degenerates, which happens mostly to unknots, the knot's topology changes, and we fail to tie the knot we have arranged. If the friction lock happens, the knot topology will not change. However, some friction locks may prevent other friction locks from happening if they are closer to the open ends. We refer such scenario as a *premature friction lock*. Both situations have to be avoided. The existence of an object in the cells that prevents the cells from approaching perimeter zero would serve the purpose very well. For simplicity, we let the object to be discs.

The tightening process is to transfer the knot from a loose configuration to a tight configuration where almost all cells reach corresponding minimum perimeters. But, how does the configuration change during the tightening? From the starting to the final time of tightening, let us consider the change of the projection of the knot configuration on the $x$-$y$ plane. Let us choose a starting loose configuration as a linear scaling of the final tight configuration for simplicity, and choose all the intermediate configurations based on the interpolation of this scaling. This collection of configurations, if arranged by time from start to final, describes our desired tightening sequence.

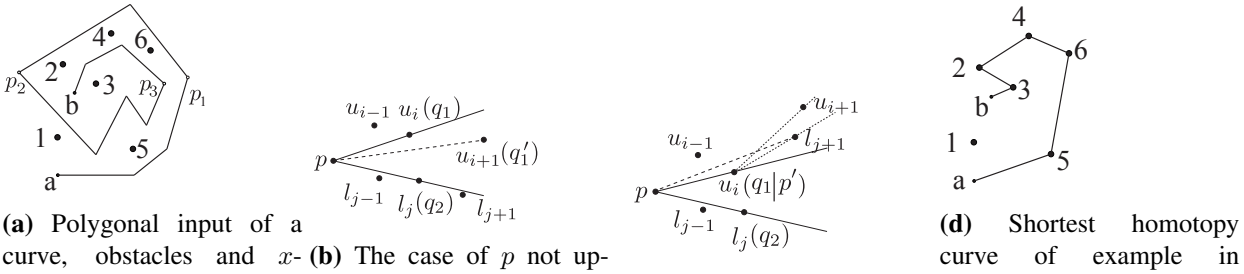
The tightening process (motion of the string) can be described as a surface of the string in $(x, y, t)$ space-time ($t \in [0, 1]$). Each $t_i$ slice of the surface is the knot configuration at the certain time during tightening, with $t = 0$ and $t = 1$ being the start (loose) configuration and goal (tight) configuration.

A configuration of string at time $t_i$ can be enforced by placing a collection of $k$ (a constant, equals the number of cells) discs at appropriate locations, denoted as $g_j(t_i), j \in \{1, 2, \ldots, k\}$. The configuration of the knot around these discs is expected to approximate the designed geometry at $t_i$. Extrude each disc from time 0 to 1, we get a collection of rods that are leaning inwards. This collection of tilted rods ("space-time obstacles") can be physically built by by substituting the $z$ dimension for the time dimension. Starting from



**(a)** Loose string around disc obstacles. **(b)** Tight around disc obstacles. **(c)** Reconfiguration of discs. **(d)** Fixture to tie star shape.

**Figure 4.9:** String tightening around set of rods.




**(a)** Polygonal input of a curve, obstacles and $x$-monotone paths. **(b)** The case of $p$ not updated. **(c)** Case of $p$ gets updated. **(d)** Shortest homotopy curve of example in Figure 4.10a.

**Figure 4.10:** The illustration of the search for shortest homotopy curve.

the base of this tilted rods where the string is loosely arranged with correct topology, we can tighten knots by pulling the open ends while moving the rods in $z$ direction. Figure 4.9 shows an example of using a fixture to tighten star shape. Even though this is not a knot, it shows the general idea of our tightening approach.

What is the configuration of the string wrapped around the discs? A simpler version of the problem has been studied in computational geometry–finding the shortest curve within the same homotopy class as the initial "loose" polygonal curve configuration among a collection of *point obstacles* with known coordinates [Bes03, EKL06, GS97, GS98, HS94]. When we know the string configuration around the discs, we know how to place the discs to approximate the desired geometry.

We used the algorithm presented in [Bes03] to compute the shortest curve within the homotopy class among point obstacles, and we will briefly introduce the algorithm. The basic idea of the algorithm is to progress to the next "anchor" as the visibility of the current "anchor" changes, until the end of the curve has been seen. The input to the algorithm is the coordinates of the point obstacles and a (presumably "loose") polygonal curve.

The polygonal curve is partitioned into a set of $x$ monotone paths such as segments $ap_1, p_1p_2, p_2p_3, p_3b$ in Figure 4.10a. Here, the $x$ monotone paths mean the path either has non-decreasing or non-increasing $x$ coordinates. An example of $x$ monotone path is shown in Figure 4.11. The algorithm then finds a *canonical representation* of each $x$ *monotone path*, describing the relationship between each path and the point obstacles. The canonical representation of the curve shown in Figure 4.10a is $1^-, 2^-, 3^-, 4^-, 5^-, 6^-, 6^+, 5^+, 4^+,$

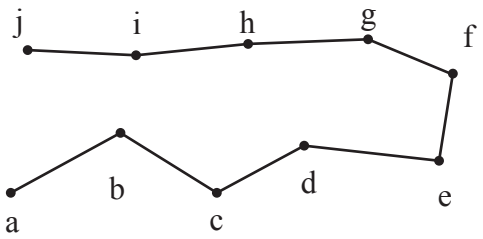



**Figure 4.11:** On a continuous path from $a$ to $j$, segment from $a$ to $f$ is a $x$-monotone path, and segment from $f$ to $j$ is another $x$-monotone path.

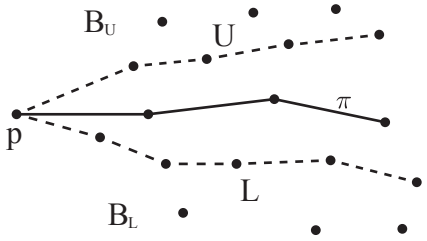



**Figure 4.12:** A monotone path $\pi$, points above $\pi$: $B_U$, points below $\pi$: $B_L$, the lower envelope $U$ of $B_U$, and upper envelope $L$ of $B_L$.

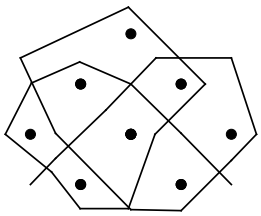
$3^+, 2^+, 2^-, 3^-, 3^+$ after some simplification, where each point obstacles is labeled with $k \in \{1, 2, \ldots, n\}$, with a $^+$ sign if it is above (otherwise $^-$) the corresponding monotone path. The simplification is to delete the adjacent two numbers with same sign and same label, such as $1^+$ and $1^+$ was deleted between $2^+$ and $2^-$.

For each monotone path $\pi$ starting with $p$, let $B_U$ denote the points above $\pi$ and $L_U$ denote the points below $\pi$. Then, denote $U$ as the lower envelope of $B_U$ and denote $L$ as the upper envelope of $B_L$. The goal is to find the shortest admissible curve from $p$ to the end of $\pi$ between $U$ and $L$, such that this found shortest curve is in the same homotopy class as $\pi$. The relations among $\pi$, $B_U$, $B_L$, $U$ and $L$ are shown in Figure 4.12.

The approach is based on visibility. Let us refer to the area with apex at $p$ and between $U$ and $L$ as a funnel. When there exist a straight line between $p$ and $u_{i+1}$ (or $l_{j+1}$) in the funnel, then the shortest curve is updated to $u_{i+1}$ ($l_{j+1}$) with a straight line from $p$ to $u_{i+1}$ ($l_{j+1}$), as shown in Figure 4.10b.

When such a straight line does not exist, ($pl_{j+1}$ (or $pu_{i+1}$) intersects with $U$ or $L$), we need to find $q_1 \in U$ and $q_2 \in L$ such that all $u_k, k \leq i$ are above or on $pq_1$ and all $l_k, k \leq j$ are below or on $pq_2$, as the example shown in Figure 4.10c. Either $q_1$ or $q_2$ becomes the new apex (replace current $p$), depending on whether $pl_{j+1}$ is above $pq_2$ (so, if $pu_{i+1}$ is below $pq_1$, then $q_2$ is chosen as new apex). The shortest curve then then updated to $q_1$ ($q_2$) with a straight line.

**(a)** Input to the design process, the desired geometry of a double coin knot.



**(b)** Rod placement in cells.

**Figure 4.13:** Double coin knot.

Repeat the procedure until the end of $\pi$, and apply this to all $x$ monotone paths. The recorded paths is then the shortest homotopy curve within the same homotopy class as original input.

Using the presented algorithm, we were able to design the location and the tilting angle of the rods to enforce the geometry during tightening. Using the designed titled rods, we can tighten the knot by simply pulling on the open ends of the knot, as long as the knot is correctly arranged at the bottom of these rods.

To arrange the knot at the bottom of these tilt rods; however, is also not a trivial job. We attempted to combine the arrangement fixture together with the tilted rods tightening fixture to tie knots directly from an untangled configuration. In previous section, we showed that six-piece fixtures can be used to arrange knots around various objects. Therefore, by consider the tilted rods as objects the knots need to be arranged around, six-piece arrangement fixtures can be designed.

We presented an automated design process in [WBB14] that can design the tightening fixture and embed them in a *six-piece* arrangement fixture. Similar to the fixtures used to arrange knots around objects, the two additional pieces come from cutting both the top and the bottom pieces into half such that they can be separated to expose the tightening rods. A straight section is extended from the bottom of the tilted rods to embed them into the arrangement fixture. Each straight section of the rods is treated as a vertical section of the string that extends to go through the top and bottom pieces. All the straight sections, including the vertical string sections and the vertical rods, cannot have occlusion along the pre-selected direction.

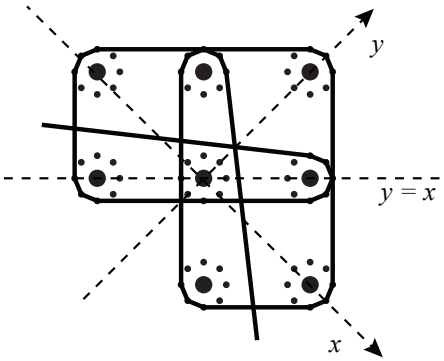
### 4.2.1 Automated design process

We use a double coin knot as an example to demonstrate the automated design process. A double coin knot, and the input to the algorithm are shown in Figure 4.13.

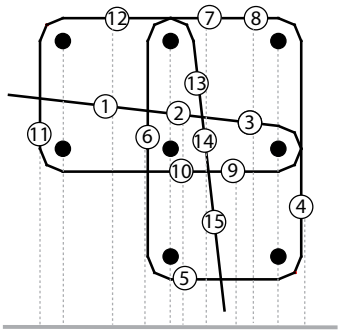
The design process is the reverse of the tying process: from the final desired configuration of the knot, arrange rods, scale up the knot, and embed knot and rods in an arrangement fixture, enforcing that all the vertical string segments and rods are visible from a chosen direction.

We start with a photo of the knot in its desired configuration along with corresponding crossing sequence (knot diagram), such as the double coin knot in Figure 4.13a. By hand, we mark the outline of the knot shape in the photo, roughly identifying the geometric configuration of the knot and the centers of the cells. The centers of the cells then form the top of the tightening fixture, as shown in Figure 4.13b.

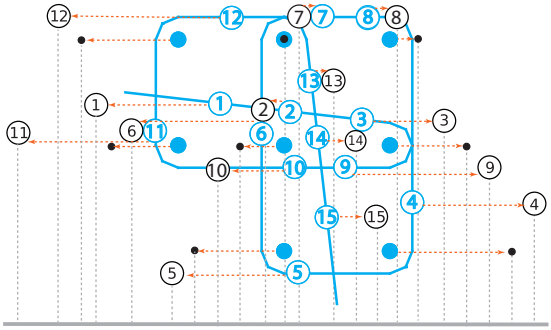
The next step is to choose an arbitrary direction along which to enforce the visibility property on all rods and vertical string elements. In Figure 4.14, we chose the original $y = x$ line to be the axis onto which



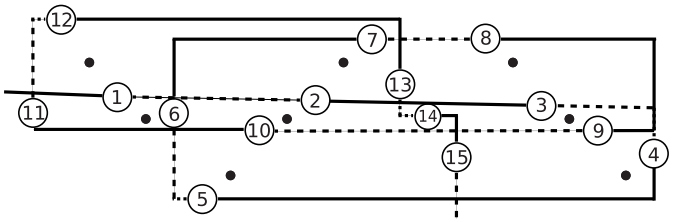

**(a)** Finding the shortest curve around points guaranteeing a minimum clearance from rods. Dashed lines show the original axis and the chosen $y = x$ axis on which the vertical sections will be projected.



**(b)** Based on the sequence of crossings, identify the vertical sections of string (circles on the string), and the rods. Occlusions are shown by projecting rods and vertical string to the chosen axis (gray line on the bottom) using dashed lines.

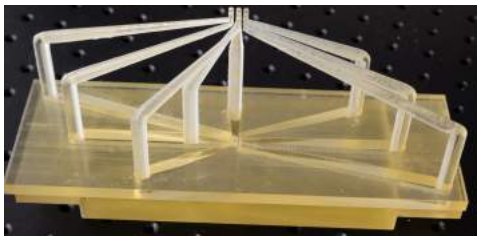


**(c)** Remove occlusions.



**(d)** Simplified layout of string around rod positions, using computed vertical segment positions from (c).

**Figure 4.14:** The fixture design process.



**(a)** The tightening fixture for a double coin knot.

**(b)** A double coin knot tightened using the fixture in Figure 4.15a.

**Figure 4.15:** The tightening fixture for the double coin knot and a knot tightened using this fixture.

vertical elements are projected to determine visibility. For convenience, we then transform the coordinate system so that the projection axis is the x-axis.

In the transformed coordinate system, find the shortest homotopy curve of the string among given rods. We chose to use eight points to approximate each disc, with some clearance to allow for the arrangement fixture to surround the rods. An example is shown in Figure 4.14a.

Along the shortest homotopy curve, based on the knot crossing information, identify all the vertical string segments (denoted as segment nodes in [Bel10] [BcfaKB14]), as shown in Figure 4.14b. (For convenience of the human designer, we also orthogonalize the tubes for the string.)

Finally, remove occlusions using the procedure proposed in the proof of Proposition 2. Figure 4.14c shows the occlusion-removal procedure and Figure 4.14d shows the resulting layout of the tubes. Based on this layout, a human designer can model the arrangement and tightening fixture in SolidWorks or any other 3D modeling software. (This step is required because the layout, although it contains all of the most interesting required information, is two-dimensional and not formatted for 3D printing.)

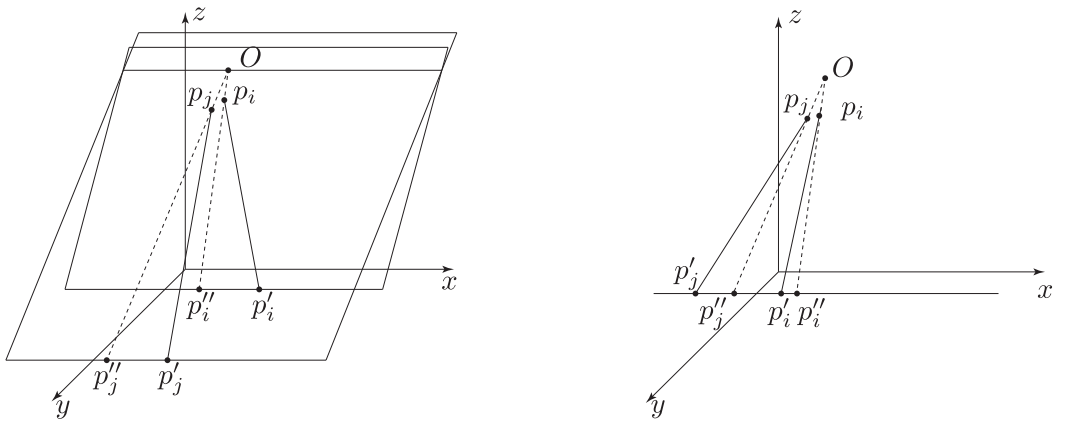
The resulting tightening fixture for a double coin knot is shown in Figure 4.15a. We applied our tightening approach using the printed fixture, and tied the double coin knot shown in Figure 4.15b, which is similar to the configuration shown in Figure 4.13a.

### 4.2.2 No collision among tilted rods

There is a slight technical problem that we might be concerned about. The apices of the parallel portions of the rods are the bases of the slanted portions of the rods. Arbitrary placements of the parallel rods might lead to slanted sections that intersect each other.

Fortunately, by careful design of the slanted sections, we can avoid this potential problem. Consider the top, slanted-rod section of the fixture. Before taking into account the need to move the parallel rods to eliminate occlusion, we might design this top section such that the fixture enforces a radial scaling of the goal configuration of the knot outwards to some less tight configuration. Since rays from a common point (formed by the scaling of the knot down to size zero) do not intersect unless they are coincident, there are no intersections between slanted rods using this approach. In fact, perturbations of these rays to avoid occlusion between their bases can also avoid intersection:

**Theorem 2** *Given a set of points $p_i, i = 1, 2, \ldots, n$ in $R^3$ with the same $z$ coordinates, there exist a set of*




**(a)** Slanted rod bases have different $y$ coordinates.

**(b)** Slanted rod bases have the same $y$ coordinates.

**Figure 4.16:** Cases for proof of Theorem 2.

*points $p'_i, i = 1, 2, \ldots, n$ on the $x$-$y$ plane where $x(p'_i) \neq x(p'_j)$ for $i \neq j$ such that the $n$ line segments connecting each $p_i$ to $p'_i$ do not intersect each other.*

*Proof:* The set of points $p'_i$ can be found in the following way. Let $z(p_i) = c$ for all $i$, where $z(*)$ represents the $z$ coordinates of a point and $c$ is a positive constant. Choose a value $h > 0$ representing the desired height of a radial projection center point above all $p_i$. Let this center point $O$ have coordinates $(\frac{1}{n}\sum_{i=1}^{n} x(p_i), \frac{1}{n}\sum_{i=1}^{n} y(p_i), c + h)$. Then for any $i$, the point $p''_i$ is on the ray $Op_i$ with $z(p''_i) = 0$.

If $y(p''_i) \neq y(p''_j)$ for $i \neq j$, choose any $p'_i$ and $p'_j$ such that $y(p'_i) = y(p''_i)$ and $y(p'_j) = y(p''_j)$ (Figure 4.16a). Denote the plane $\mathbb{P}_k$ as the plane that contains the line $y = y(p''_k)$ and ray $Op''_k$ for any $k$. Plane $\mathbb{P}_i$ and plane $\mathbb{P}_j$ intersect at the line that passes through $O$ parallel to the $x$ axis. We know $z(p_k) < z(O), k \in \{1, 2, \ldots, n\}$. Choose $p'_i$ and $p'_j$ to have the same $y$ coordinates as $p''_i$ and $p''_j$ respectively, they belong to two different planes. Therefore, line segments $p_ip'_i$ and $p_jp'_j$ do not intersect.

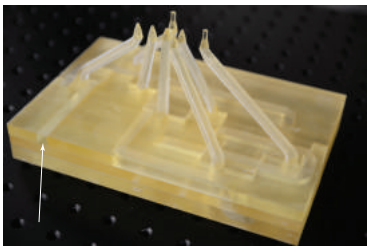
If $y(p''_i) = y(p''_j)$, choose $p'_i$ and $p'_j$ such that $\text{sign}(x(p'_i) - x(p'_j)) = \text{sign}(x(p_i) - x(p_j))$ (Figure 4.16b), $p_ip'_i$ will not intersect $p_jp'_j$.

Overall, we can easily enforce $x(p'_i) \neq x(p'_j)$ for $i \neq j$; therefore we have found $p'_i$. ■

Therefore, if we first radially scale all rods outwards from $p''_i$, then move to $p'_i$ to guarantee visibility, the slanted rods connecting $p'_i$ to $p_i$ for all $i$ will not intersect each other.

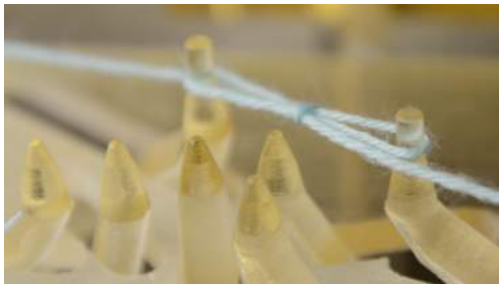
The proof assumes the rods are of zero radius, but if the cross sections of rods are circles with radius $r$, and $|y(p'_i) - y(p'_j)| < r$, the rods can still intersect. In this case, we can find $O'$ where $z(p_i) < z(O') < z(O)$, such that the new $p'''_i$ on the ray $O'p_i$ satisfying $|y(p'''_i) - y(p''_j)| > r$. Using $p'''_i$ to replace $p''_i$ and finding new $p'_i$ resolves the problem.

Using the design process, we designed different fixtures to tie different knots, such as shoelace unknot, and double coin knot, the fixtures are shown in Figure 4.17 and Figure 4.15a. Using the fixtures, we were able to arrange and tighten corresponding knots, shown in Figure 4.18.

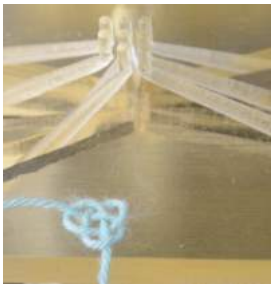


**(a)** Fixture for shoelace unknot.

**Figure 4.17:** Knot tying fixtures with tilted rods for shoelace unknot.



**(a)** A tied shoelace unknot.



**(b)** A tied double coin knot.

**Figure 4.18:** Corresponding knots tied using fixtures shown in Figure 4.17.

## 4.3 Precise tightening

In addition to the knots we considered earlier, there are also other knots that can be even harder to tie correctly. For example, we were unable to control the relative sizes of two bows in the previously tightened shoelace unknot. But for appearance purpose, the two "bows" should have the same size. There are other knots like the shoelace unknot that requires certain part of the knot to take on certain size, such as *cloverleaf knot*, which is shown in Figure 4.19.

**Figure 4.19:** A cloverleaf knot.

Some other knots, such as *sounding line* with knots tied at six foot (one fathom) intervals along the line which are used by sailors, require similar distance constraint between selected contacts. We would further want to tie these knots satisfying these constraints, which has rarely been considered before in robotics community.

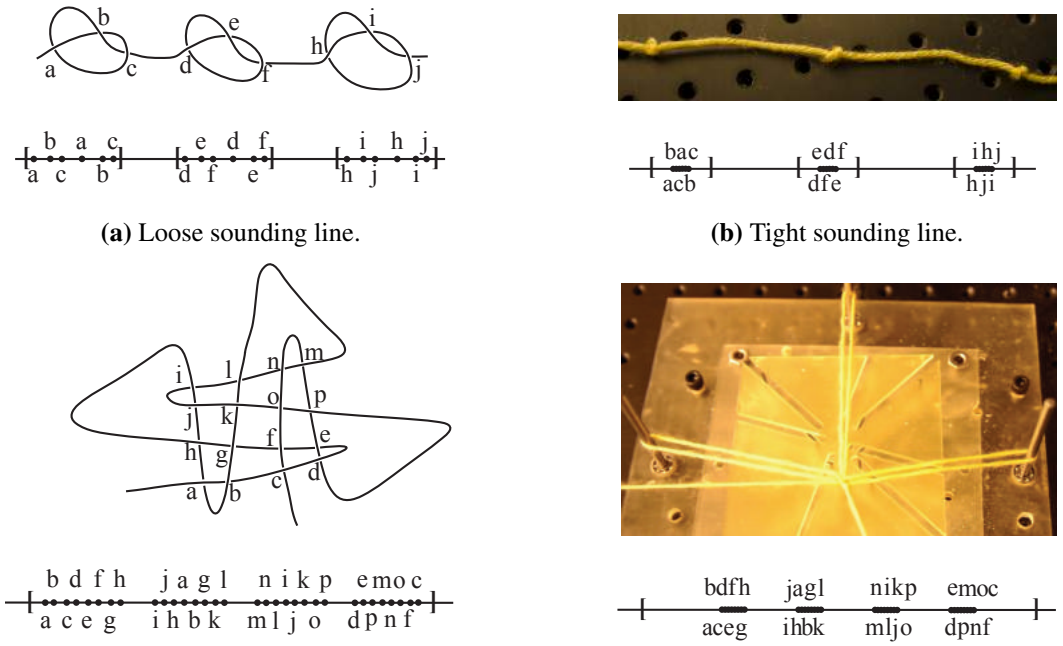
We define such knot tying task, where certain distance constraints have to be satisfied between selected friction locks, as *precise tightening*.

To better describe the distance constraints on the knots, we introduce a *contact diagram* where each black circle represents a crossing (regardless an over- or under-crossing) along the untangled string, and a pair of matching brackets represents a *knot unit*. Here we define a knot unit as segments of string within which all labels in the Gauss code have appeared twice. Note that there will be no more than $O(n^3)$ knot units in a knot with $n$ crossings; we will primarily be concerned with *atomic* knot units that do not contain any other non-empty knot units. The sounding line that we consider is a *compound knot* in the sense that it is a collection of knot units. The cloverleaf knot; however, contains only a single knot unit.

In Figure 4.20, contact diagrams of loose and tight sounding line and cloverleaf knot are shown. With contact diagram, the goal of precise tightening can be clearly described.

### 4.3.1 New knot arrangement method

At first glance, it seems that several units of sounding line are not as complex as Carrick Bend, so we can still design a fixture and use pressurized air to arrange the knots. However, the distance constraint between knot units make it more challenging. To maintain the distance between knot units, the tightening fixture for each knot unit have to be distance $d$ away from each other. This means a segment of string with length




**(a)** Loose sounding line.

**(b)** Tight sounding line.

**(c)** Loose Cloverleaf knot.

**(d)** Tight cloverleaf knot.

**Figure 4.20:** Contact diagrams on loose knot diagrams and machine-tightened sounding line and cloverleaf knot. The range between the brackets indicates the knot units, and the black circles indicates the crossings and their corresponding locations along the string.

$d$ needs to be arranged between knot units. This makes the fixture for arrangement very large. The string presumably cannot be pushed through such large fixture with pressurized air. There is also the possibility to arrange each knot unit using an arrangement fixture, but the re-grasping task for insertion at each fixture is also complex. Therefore, we need to find alternative ways to arrange the knot.

To arrange the knot around stationary fixtures with predefined distance along the string, we used a robot arm with a special gripper. The gripper is built with two linear motors with a electromagnet attached to the tip of each motor. The knot is laid out using a spool. The choice was made after we observe obvious resistant force when the knot is pulled by the open ends. The gripper can pick up the slider where the spool was attached by either linear motor (shown in Figure 4.21), or both motors. The top of the slider was mounted with two metal plates for the easy pick up using electromagnets. With this gripper, a under-crossing can be arranged without dropping the spool, simplifying the re-grasp task. The entire procedure of arrangement is just to follow the knot diagram around the tightening fixture and make all under-crossings.

### 4.3.2 New tightening approach

Tightening with tilted rods showed positive results in previous work; however, a few observations also challenges its applicability to precise tightening. Our tightening using the tilted rods was based on the

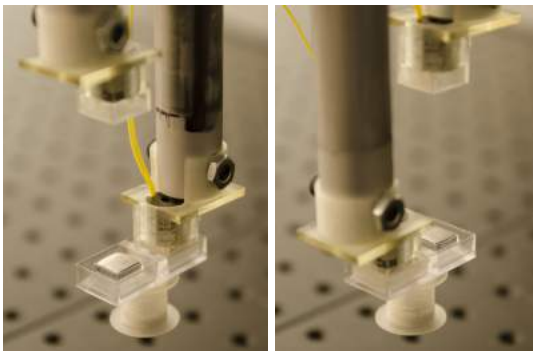
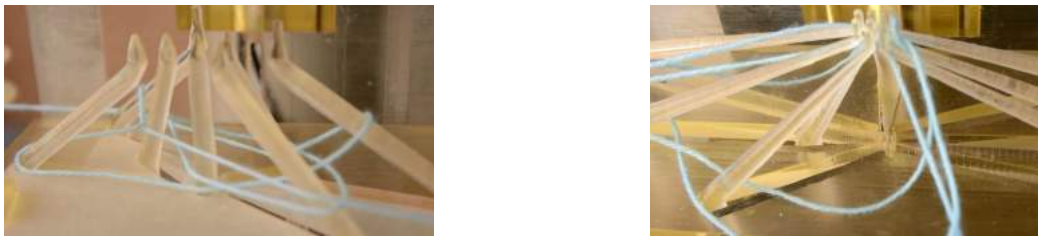


**Figure 4.21:** Two different configurations of the gripper grasping the spool used to arrange the knots.

assumption that during tightening, the knot remains flat. However, the assumption does not hold. Due to the different slope of the tilted rods, and different distance to the open ends where the forced is applied, different segments of the string have different tensions, leading to different "climbing" speed on the tilted rods. The tightening also relies on the knot being taut during tightening, which does not always hold either. Figure 4.22 shows the tightening of a shoelace unknot and a double coin knot where the assumption breaks. What is more, we also begin to question if all the rods used in previous approach are necessary.



**(a)** Tightening a shoelace unknot, where the knot does not remain flat.

**(b)** Tightening a double coin knot, where the knot is not taut.

**Figure 4.22:** Examples of tightening with tilted rods where assumptions breaks.

When we isolated the problems, we came up some hypothesis to explain the violations of the assumptions. String on tilted rods usually will not maintain flat, due to tensions, frictions and distribution of force. Different slopes of the rods makes the analysis even more complex. Admittedly, we can analyze the tension along the string loosely based on the capstan equation [Att99], and thus estimate the friction on each contact, and lead to a re-design of the tilted rods to have different slops. This approach is very complex, and may eventually violate the final knot geometry that is even more critical for tightening. A simpler solution would be to just use straight rods instead of tilted ones, though these rods need to move closer to reach the final knot geometry.

Partial knots that remain loose during tightening can only be caused by friction between string and rods,

or between string and string. We need to prevent or at least delay either type of contact, which was the purpose of our tightening fixture in the first place. According to the previous proposal to use straight rods, we can mount all these rods on low-friction ball bearings to reduce the friction between string and rods. However, after testing, we discover that the remaining string-string contact still lead to loose segments of the knot. After a closer look, we isolated the problem to multiple strands of string contacting the same rod, as shown in Figure 4.23 on the left.

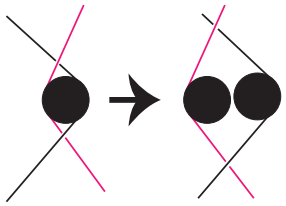


**Figure 4.23:** When multiple strands of string contact the same rod, insert one extra rod in the cell.

We also would like to avoid letting multiple strands of string contact the same rod, since two strands might move at different velocities relative to the rod, preventing free rotation. We separate contacting strands of string by doubling and separating such rods; an example is shown in Figure 4.23.

How should duplicate rods be placed? Denote the rod as $r$ and the two strands of string as $arb$ and $crd$. For $arb$, let the new rod be $r'$, such that the vector $rr'$ is perpendicular to vector $ab$. We would like to minimize friction around unpowered rods, to allow tension to best be distributed. Friction can be minimized by minimizing the wrapping angle around the rods.

**Theorem 3** *Moving a duplicated rod along the direction pointing from the original rod location to the middle of the original contact arc with a distance $d$ minimizes the angle that string wraps around the new rod.*

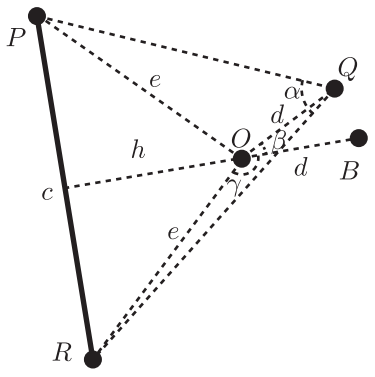



**Figure 4.24:** The notation for the triangle where the string wraps around the moved rod.

*Proof:* Consider the extreme case where the radius of the rod is 0, with only one strand of string (the other strand is symmetric). Choose coordinates so that the location of the original rod is located at $O = (0, 0)$. The vertex $Q$ is on the circle with radius $d$ with $P$ and $R$ fixed as shown in Figure 4.24.

Let the length of $PR$ be $|PR| = c$, the distance from $O$ to line $PR$ is $h$, $|PO| = |OR| = e$, and $|OQ| = |OB| = d$. Let $OB$ be perpendicular to $PR$, where $B$ is achieved by moving $O$ along the direction pointing from $O$ towards the midpoint of the original contact arc with a distance $d$. The angle $\angle PQR = \alpha$, $\angle QOB = \beta$ and $\angle BOR = \angle BOP = \gamma$.

Using the notation in Figure 4.24, denote the area of the triangle $PQR$ as $A$. We have the following geometric relations:

$$\begin{aligned} 2A &= c \cdot (h + d \cdot \cos\beta) && (4.1)\\ &= xy \cdot \sin\alpha && (4.2)\\ \cos\alpha &= (x^2 + y^2 - c^2)/2xy && (4.3)\\ xy &= (x^2 + y^2 - c^2)/2\cos\alpha && (4.4)\\ \cos(\gamma - \beta) &= (e^2 + d^2 - x^2)/2ed && (4.5)\\ \cos(\gamma + \beta) &= (e^2 + d^2 - y^2)/2ed && (4.6)\\ \cos\gamma &= h/e. && (4.7) \end{aligned}$$

Adding Equation 4.6 to Equation 4.7, we have

$$x^2 + y^2 = 2e^2 + 2d^2 - 4dh\cos\beta. \tag{4.8}$$

Combining Equation 4.8 and Equation 4.3,

$$2A = (e^2 + d^2 - (c^2)/2 - 2hd \cdot \cos\beta) \cdot \tan\alpha. \tag{4.9}$$

Combining Equation 4.2 and Equation 4.9,

$$\begin{aligned} (e^2 + d^2 - \frac{c^2}{2} - 2hd \cdot \cos\beta) \cdot \tan\alpha &= c \cdot (h + d\cos\beta)\\ \Rightarrow \frac{c}{\tan\alpha} &= \frac{e^2 + d^2 - (c^2)/2 - 2hd \cdot \cos\beta}{h + d\cos\beta}\\ \Rightarrow \frac{c}{\tan\alpha} &= \frac{d^2 + e^2 + 2h^2 - \frac{c^2}{2}}{h + d\cos\beta} - 2h \end{aligned} \tag{4.10}$$

Since $d^2 + e^2 + 2h^2 - (c^2)/2$ and $c$ are constants, when $\alpha$ is increasing, the left side of Equation 4.10 is decreasing, which leads to the increase of $\cos\beta$ and decrease of angle $\beta$. Therefore, when $\alpha$ increases, $\beta$ decreases. When the angle $\beta$ decreases, area $A$ increases. When $\beta$ reaches 0, the area $A$ reaches a maximum, as does $\alpha$, minimizing the wrapping angle of the string around the rod. ■

To summarize, we use a different approach to arrange the knot, and use only straight rods instead of tilted rods to tighten the arranged knots. The tightening fixture consists of stationary and moving rods. The moving rods are used to bring crossings closer to each other to control the locations of friction locks. The

stationary rods are used to enforce distance constraints. We choose to mount the moving rods onto a slider, which is shown in Figure 4.25, and let them move along a collection of carved tracks on the tightening fixture.

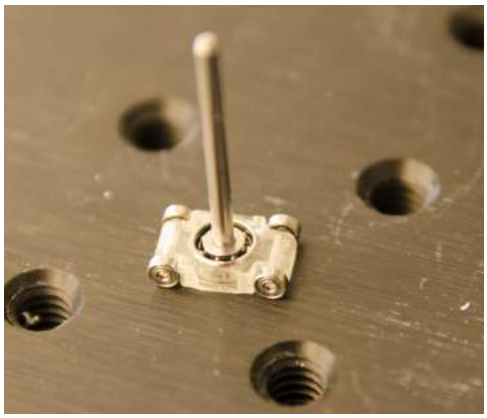


**Figure 4.25:** Rod mounted on a slider to move.

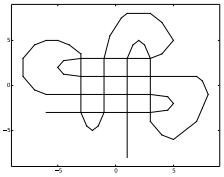
### 4.3.3 Automated fixture design

As the tightening fixture transforms the knot from a loose configuration to a tight configuration, the fixture needs to be able to represent both configurations of the knot. This section describes a procedure that takes a loose knot diagram as input, as well as the required distance between contacts.

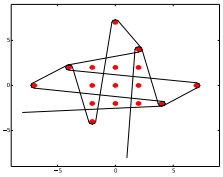
The outline of the approach is as follows. First, the procedure needs to determine where to place rods to support the loose knot. Based on the required distance between contacts, the procedure then needs to identify the rods that will remain in position as the knot is tightened (*e.g.*, to hold the bows of a shoelace in place). If there are rods that contact more than one strand of string, the procedure duplicates these rods. The procedure then compares the knot layout and the size of the spool, expanding the layout if necessary to allow the arrangement. Then, the procedure moves all non-stationary rods closer towards their geometric center. The rope length between contacts are then calculated, and adjustments are made to satisfy the required distances. The trajectories of the individual rods become the tracks.

We now describe in more detail each of the steps of the design process. In previous work [WBB14] on knot arrangement, we showed that one rod per cell is sufficient. In the current work, we prefer to avoid using a rod for each of the 16 cells in the cloverleaf knot, or for each of the 64 cells in the Ruyi knot. Procedure RodPlacement first identifies the cells, putting a rod in each cell. We then find the shortest curve among these rods, using an algorithm for finding the shortest curve of a particular homotopy class among points [Bes03]. (We sample points along the boundaries of discs representing rods, as in [WBB14].) We remove rods not in contact with this curve. For the cloverleaf knot, 7 rods are placed.

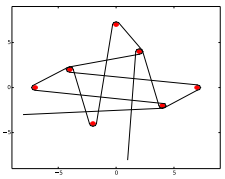
The current fixture can be used to support the knot during arrangement, but some places might be too small for the gripper designed to perform the re-grasp task. We can add certain size constraints to each cell where a re-grasp might happen. This step (Procedure IncreaseCellSizes) happens after Procedure RodPlace-



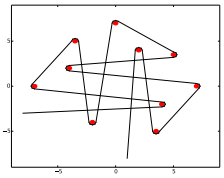
**(a)** The polygonal approximation of the cloverleaf knot diagram.



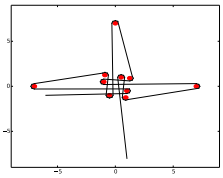
**(b)** Place a rod in each cell, following Procedure RodPlacement.



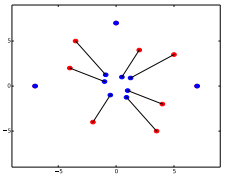
**(c)** Delete all rods that do not cause cell degeneration, following Procedure RodPlacement.



**(d)** Split rods to let each rod contact only one strand of string using Procedure IncreaseCellSizes.



**(e)** Reduce the rope length using Procedure ReduceRopeLength. This figure shows the results after applying the procedure.



**(f)** Design the tracks by connecting corresponding rods using a line segment.

**Figure 4.26:** Automated design of the fixture for the cloverleaf knot using presented procedures

ment.

After all the rods are placed, we find a set of line segments (tracks) that guide the moving rods to a nearly tight configuration using Procedure ReduceRopeLength.

Finally, it is important to control the distance between contact points for precise tightening. The distance is calculated between two nearest crossings using the computed shortest curve length. If the distance is not yet satisfied around the rods, we move the corresponding static rods parallel to the direction of the contact normal at mid-point of the string contact to meet the distance constraint.

Figure 4.26 shows the execution of the previous three procedures in sequence to design the fixture for the cloverleaf knot.

We then followed the design procedure and produced several fixtures for different knots, and conducted experiments to test the precision of the tightening for *sounding line*, *shoelace unknot* and *cloverleaf knot*. On the tightening fixture, the moving rods are powered by the tension along the string when it is pulled on the open ends. Since these moving rods are mounted on sliders that have low friction, a small force should be sufficient to motivate them. The pulled length is related to the rope length difference between loose and tight configuration of the knot. So, no sensing is needed during tightening.

The sounding line is a sequence of overhand knots (top of Figure 4.20b) with equal distance between adjacent units. We consider the sounding line to be a *compound knot* in the sense that it is a collection of *knot units*: segments of string within which all labels in the Gauss code have appeared twice. Note that

**Procedure** RodPlacement

**Input**: polygonal drawing $\mathcal{D}$; cells of $\mathcal{D}$ : $\mathcal{C}_i$;
**for** *each cell* $\mathcal{C}_i$ **do**
  Find the largest containing disc with center $d_i$ and radius $r_i$;
$r \leftarrow \min_{\forall i} r_i$;
Place a disc of radius $r$ at each $d_i$;
Find the shortest homotopy curve among these discs;
**for** *disc centered at* $d_i$ **do**
  remove the disc, detect if any cell degenerates;
  **if** *cell degenerates* **then**
    place disc with radius $r$ centered at $d_i$;
**while** $\exists$ *disc at* $d_i$ *contacting two stands of string* **do**
  double the disc so each disc only contacts one strand of string;

there will be no more than $O(n^3)$ knot units in a knot with $n$ crossings; we will primarily be concerned with *atomic* knot units that do not contain any other non-empty knot units.

We attempted to place adjacent knot units of a sounding line at 78mm (rather than one fathom) apart. The machine-tied sounding line (with three knot units) had average distances 75mm and 77mm between adjacent knot units. Figure 4.27 shows the fully automated arranging process of tying a sounding line, and a snapshot of the automated tightening process is shown in Figure 4.32. For both the sounding line and cloverleaf knots, distance errors (measured along the string) were less than $5\%$, averaged over 100 trials.

A tied cloverleaf knot is shown in Figure 4.31, and the tightening process is shown in Figure 4.30. This knot has 16 crossings, and its knot diagram is shown in Figure 4.33. Three bows of the cloverleaf knot between $b$ and $c$, between $d$ and $e$, and between $f$ and $g$ should have equal lengths. On the fixture, we enforce the distance to be 200mm, and the machine-tied knots had average lengths of 189mm, 190mm and 191mm for the three bows in our experiments. We believe the error is due primarily to the stretch of the yarn.

### 4.3.4 Active tightening fixtures

Observe Figure 4.30c, some rods did not reach the desired positions. This effect may cause errors in the tightening result. To further improve the tightening result, we proposed an alternative way to move the moving rods.

A bar-linkage system was designed to connect all the sliders to a single motor. The motor then can move all the moving rods at the same time, and guarantees them to reach the designed positions. We refer to this improved fixture design as *active fixtures*. A CAD model of an active fixture for cloverleaf knot is shown in Figure 4.34.

We used the improved tightening fixture to tie the cloverleaf knot, and the procedure is shown in Figure 4.35. Notice in Figure 4.35d, all moving rods reached the desired positions. This design principle can further improve the precision of the knots tied. However, additional controls need to be used to achieve

**Procedure** IncreaseCellSizes

**Input**: Crossing sequence: $\mathbb{S} = \{s_1, s_2, \ldots, s_n\}$, mapping $f(s_i) : \mathbb{N} \to \mathbb{R}^2$

**for** *under-crossing (over-crossing)* $s_i$ *and next over-crossing (under-crossing)* $s_j$ **do**

  **if** *distance between* $f(s_i)$ *and* $f(s_j)$ *too small* **then**

    Find previous rod $r_{i,p}$ and next rod $r_{i,n}$ of $s_i$;

    Find previous rod $r_{j,p}$ and next rod $r_{j,n}$ of $s_j$;

    **if** $s_i$ *and* $s_j$ *shares a common adjacent rod* **then**

      Find the common rod $f(r_{ij})$, and the middle point $f(s_{i,j})$ between $f(s_i)$ and $f(s_j)$;

      Create rod $f(r'_i)$ and $f(r'_j)$, such that $f(r'_i)f(r'_j)$ is perpendicular to $f(s_{i,j})f(r_{ij})$ and pass through $f(r_{ij})$, let $f(r'_i)f(r_{ij}) = f(r'_j)f(r_{ij}) = d$;

      Denote the other rods adjacent to $s_i$ and $s_j$ as $r_{i,o}$ and $r_{j,o}$;

      Move $f(r_{i,o})$ to $f(r'_{i,o})$ and $f(r_{j,o})$ to $f(r'_{j,o})$, such that $f(r_{i,o})f(r'_{i,o})$ is parallel to $f(s_i)f(s_j)$, and $f(r_{j,o})f(r'_{j,o})$ parallel to $f(s_i)f(s_j)$, let $f(r_{i,o})f(r'_{i,o}) = d$ and $f(r_{j,o})f(r'_{j,o}) = d$;

      Let $2d + f(s_i)f(s_j) > D$ where $D$ is larger than the spool size;

    **else**

      Move $f(r_{i,p})$ to $f(r'_{i,p})$ such that $f(r_{i,p})f(r'_{i,p}) = d$ and $f(r_{i,p})f(r'_{i,p})$ parallel to $f(s_i)f(s_j)$;

      Move $f(r_{i,n})$ to $f(r'_{i,n})$ such that $f(r_{i,n})f(r'_{i,n}) = d$ and $f(r_{i,n})f(r'_{i,n})$ parallel to $f(s_i)f(s_j)$;

      Move $f(r_{j,p})$ to $f(r'_{j,p})$ such that $f(r_{j,p})f(r'_{j,p}) = d$ and $f(r_{j,p})f(r'_{j,p})$ parallel to $f(s_i)f(s_j)$;

      Move $f(r_{j,n})$ to $f(r'_{j,n})$ such that $f(r_{j,n})f(r'_{j,n}) = d$ and $f(r_{j,n})f(r'_{j,n})$ parallel to $f(s_i)f(s_j)$;

      Let $2d + f(s_i)f(s_j) > D$ where $D$ is larger than the spool size;

the goal. Currently, all the active fixture designs are accomplished by the author, and have not been automated. What is more, the single bar-linkage system is not capable of controlling the moving speed of each individual rods. In order to control them independently, even more complexity needs to be added.

All these considerations leads to more complex fixtures. As a proof of concept, we are happy with the current results, and will not investigate further into the details of improving the mechanisms of the fixtures.

## 4.4 Ruyi knot tying

To demonstrate the generality of the fixture based approach, we will show how to use such fixtures to tie a complex knot that no researcher has attempted before: *Ruyi knot*. A Ruyi knot consists of four cloverleaf knot units, as shown in Figure 4.36. Knot units 1, 2 and 3 are embedded into knot unit 4. Knot units 1 through 3 are located on the "bows" of the fourth unit. The tightening of the Ruyi knot is interesting, because the complexity, and the size.

**Procedure** ReduceRopeLength

**Input**: rope thickness $\mathrm{Thi}(\gamma)$; small clearance $a$, geometric center of all rods $c$; a set of stationary rods $D$

**Output**: Tight ropelength $Lp$; disc locations $\mathcal{D}$

**for** *disc* $d_i \notin D$ **do**

  connect $d_i$ and $c$;

  $k \leftarrow$ number of intersections between $\vec{d_i d_j}$ with the knot drawing;

  move $d_i$ along $\vec{d_i c}$ to $d_i'$ such that $d(d_i', c) = k \cdot \mathrm{Thi}(\gamma) + a$;

  $d_i \leftarrow d_i'$;

Calculate distance between contact points;

Move rods if distance constraint is not met;

Calculate ropelength $\mathrm{Rop}(\gamma)$;

$Lp \leftarrow \mathrm{Rop}(\gamma)$;

Initial attempts to tie a Ruyi knot were unsuccessful using a fixture with purely passive rods like those used to tie the cloverleaf knot. We observed that regardless of the amount of force exerted at the ends of the string, inner segments of string remained loose. Tension along the string caused rods to tilt away from the axis of rotation permitted by ball bearings. To distribute tension evenly along the string, we mounted several rods on motors that spin the rods. .

How should powered rods be placed? The amount of tension that a powered spinning rod can distribute depends on the frictional force between the rod and the string, which can be modeled using capstan equations [Att99] that relate the wrapping angle of the string around a rod to the maximum difference in tensions on each side of the string before slipping, based on the coefficient of friction.

Let the friction coefficient be $\mu$ and let there exist forces $T_1$ and $T_2$, with $T_1 > T_2$, applied along two ends of the string. Let the contact between string and rods span an angle of $\varphi$, as shown in Figure 4.37. There exists an angle $\varphi^{int}$ such that the following equations—capstan equations—hold:

$$\varphi^{int} = \log(T_2/T_1)/\mu \tag{4.11}$$

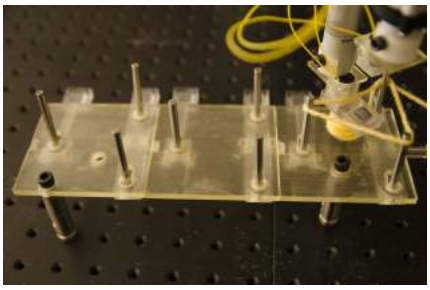
$$T(\varphi(t)) = T_2, \varphi(t) \in [\varphi^{int}, \varphi] \tag{4.12}$$

$$T(\varphi(t)) = T_1 \exp(-\mu\varphi(t)), \varphi(t) \in [0, \varphi^{int}] \tag{4.13}$$

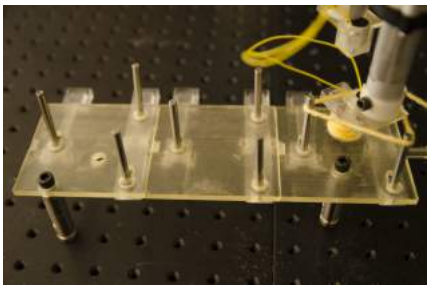
The region between 0 and $\varphi^{int}$ is called the *slip zone*. Any of the three quantities $T_1$, $T_2$ and $\varphi^{int}$ can be derived if the other two are given. The string slips around the disc if $\varphi^{int} \geq \varphi$.

We conducted experiments to test different contact material (to change friction coefficient) and different wrapping angles, trying to confirm the friction provided by the rotating rods follows the capstan equation. We used matte rods and rubber sheets around the rods to contact yarn and nylon string, and get the following plot shown in Figure 4.38. The result is consistent with that calculated from capstan equation.

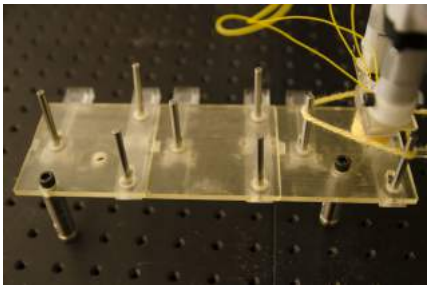
We therefore placed powered rods—rods actively spinning by motors—in a way to ensure large wrapping angles, so that sufficient tension could be distributed along the string to allow consistent tension throughout the string as the string is pulled from the ends. The layout can be seen in Figure 4.39, where the



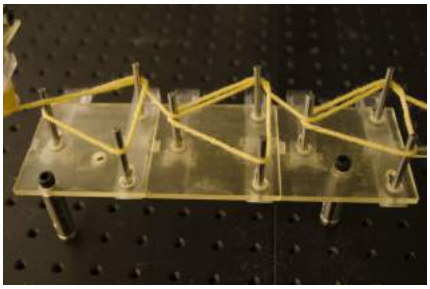
**(a)** Before re-grasp task in the first unit.



**(b)** After the re-grasp task in the first unit.



**(c)** About to finish arranging for the first unit of sounding line.



**(d)** Finished arranging three units of sounding line.

**Figure 4.27:** sounding line arrangement process.

powered rod locations are labeled with yellow dots, and all other rod locations are labeled with red dots.

Because we are only pulling on the open ends of the string, a tightening order of the knot units needs to be derived. Consider three points $a$, $b$ and $c$ along the string. If the distances between $a$, $b$ and between $b$ $c$ are longer than required, and $b$ and $c$ are fixed by friction locks, then the distance between $a$, $b$ and between $b$, $c$ cannot be changed to satisfy the requirement.

We can only plan the tightening order based on the distance to the open end based on the control strategy we choose. The further a pair of contact is to the open end, the earlier it needs to be tightened into place.

Based on different level of abstraction, we can plan the tightening order on a knot unit level, or on a string level. On a knot unit level, we should tighten knot unit 2 first, then tighten knot unit 1 and 3 before we start to tighten knot unit 4. On a string level, after the tightening of knot unit 2, some parts on the knot unit 4 is further from the open end compared to knot unit 1 and 3. Therefore, we will tighten those parts first before we tighten knot unit 1 and 3.

Experiments with both tightening orders were conducted. When we tighten the Ruyi knot based on the distance of each knot unit to the open end, only $20\%$ of the five trails succeed. But if we tighten the knot based on the distance of each pair of string contact to the open ends, all five trails succeeded ($100\%$ success rate). All predefined distances on the tightened Ruyi knot measured along the string were all within $6\%$ of target distances.

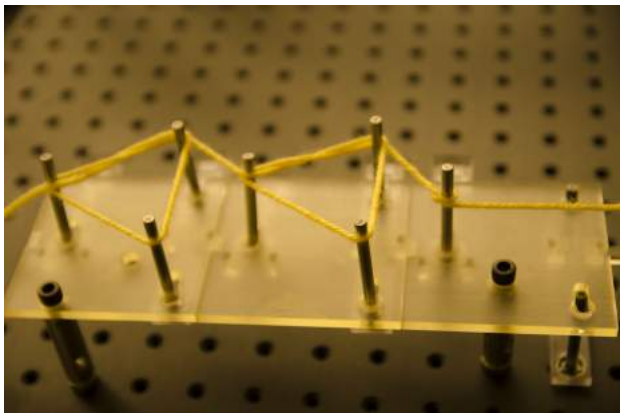
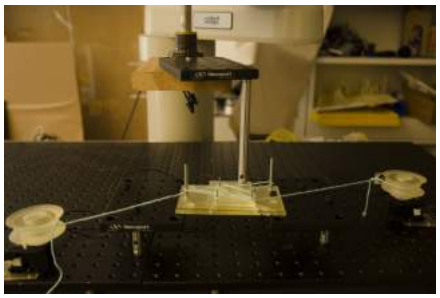


**Figure 4.28:** The first knot unit along the sounding line is tightened around the stationary pin.

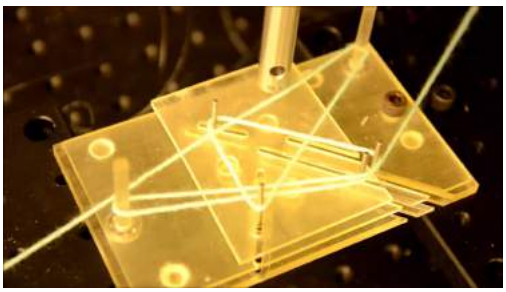
## 4.5 Conclusions

In this chapter, several iterations of fixture designs were presented to assist knot tying to a different level of requirements. For each different fixture type, they can be designed automatically using the procedures we presented in this chapter. We believe this is the first set of general knot tying approaches that applies across many different knots.

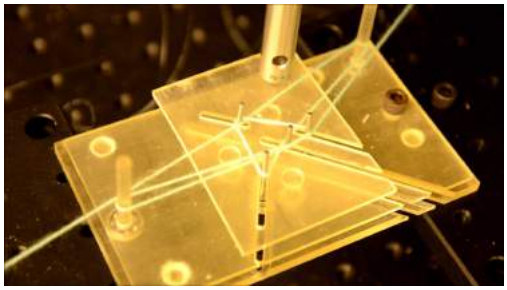
On the fixture, the string are almost always stretched taut. The direct result is the small string deformation observed, and the realization of fully controlled knot tying. The work presented in this chapter focus heavily on the mechanism design, but we were able to extract the commonality of the mechanisms, and composed the automated design process for these fixtures.



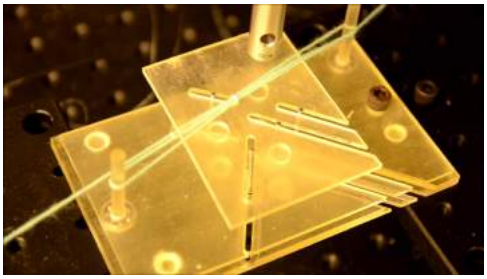
**(a)** The tightening fixture system along with motors and arm grippers.



**(b)** The starting configuration of the knot around the tightening system.



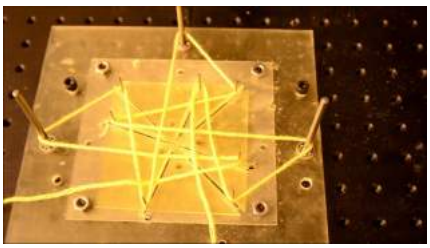
**(c)** The knot is pulled tight around the rods, and the sliding rods cannot move any further.



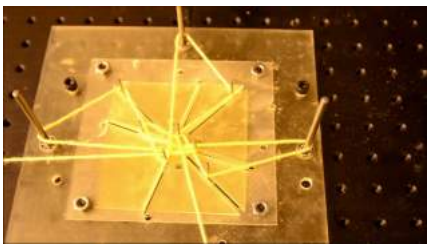
**(d)** The final configuration of the knot tightened using the passive tightening system.

**(e)** A shoelace unknot tightened by the fixture.

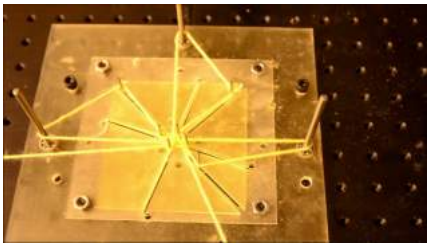
**Figure 4.29:** Tying the shoelace (un)knot.



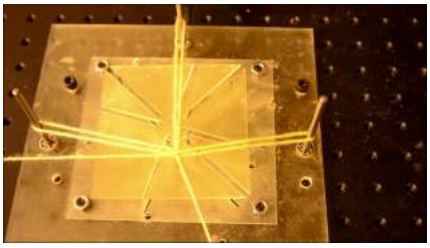
**(a)** After arrangement, the cloverleaf knot is spanned on the fixture.



**(b)** After pulling the ends for a small amount, some pins moves.



**(c)** The strings are taut around the rods, cannot be pulled further.



**(d)** The cloverleaf knot is tight around the tightening fixture.

**Figure 4.30:** Tightening a cloverleaf knot.

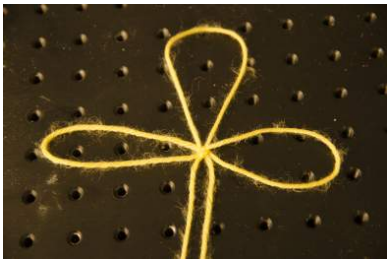


**Figure 4.31:** Machine-tied cloverleaf knot.

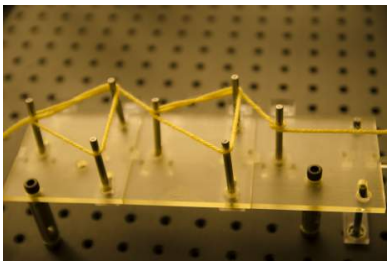


**Figure 4.32:** The first knot unit along the sounding line is tightened around the stationary pin.

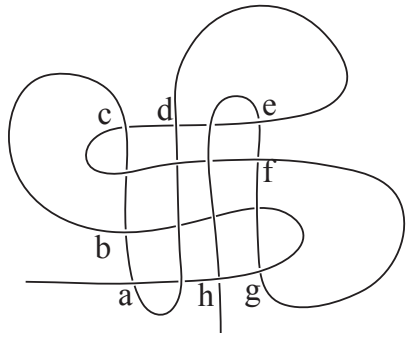
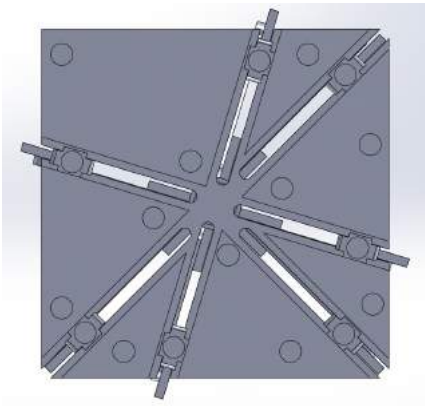
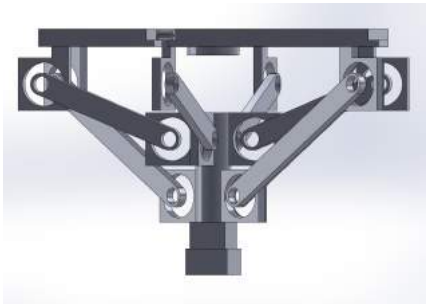



**Figure 4.33:** The knot diagram of a cloverleaf knot.

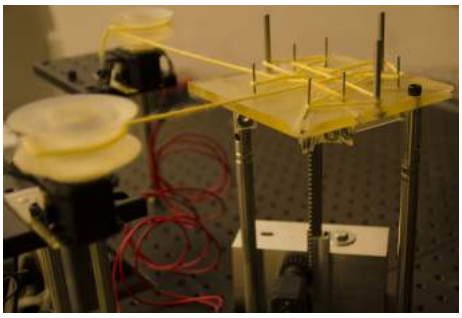


**(a)** Top view of tracks for a cloverleaf knot.

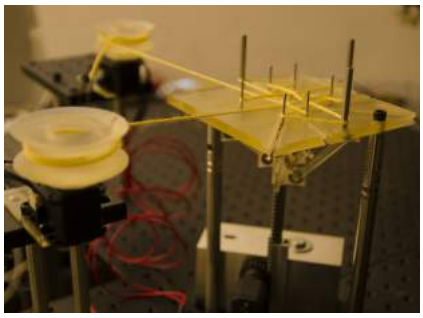


**(b)** Side view of the bar-linkage system. There are two different link lengths, to move different rods with different speeds.

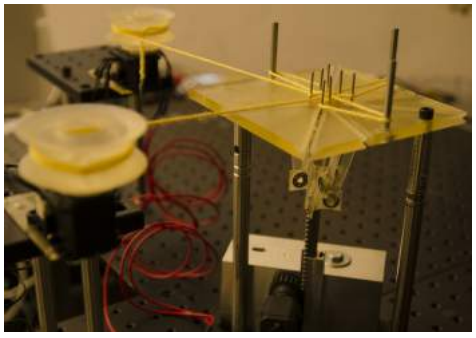
**Figure 4.34:** Model of the track and bar-linkage system for a cloverleaf knot.



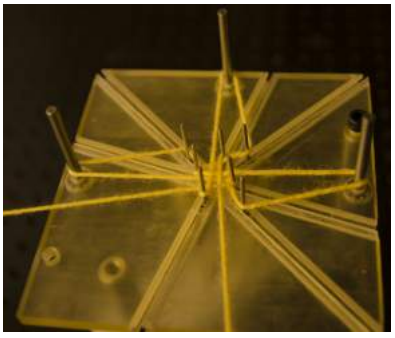
**(a)** Start of tightening with active fixture.



**(b)** During tightening with active fixture.



**(c)** Tight around active fixture.



**(d)** Tightening with active fixture; top view.

**Figure 4.35:** Tightening cloverleaf knot using active fixture.

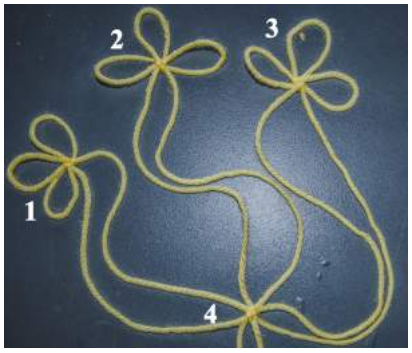



**Figure 4.36:** A hand tied Ruyi knot.

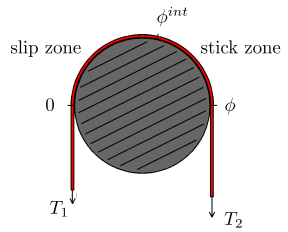



**Figure 4.37:** An illustration of the capstan equation.

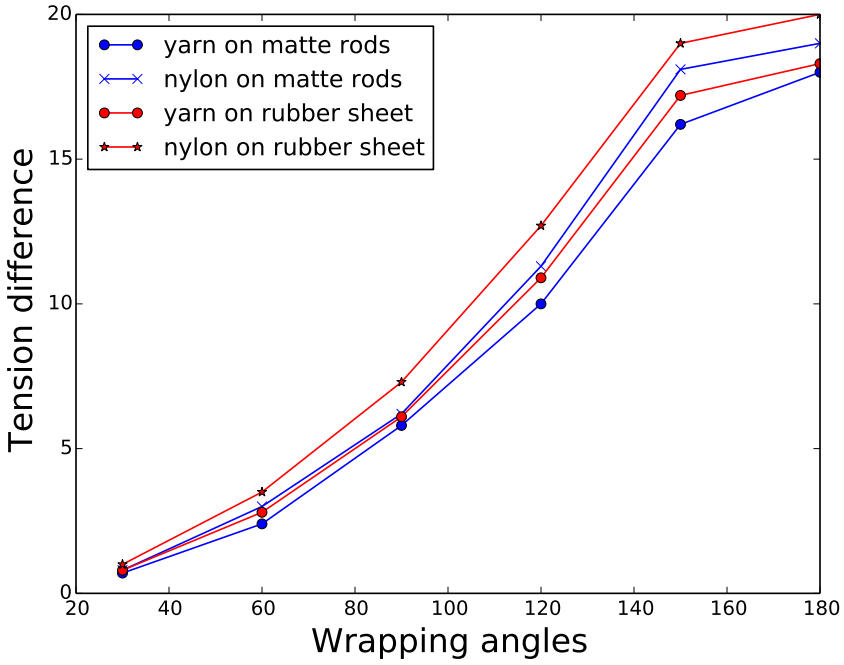



**Figure 4.38:** Friction provided by the rotating rods between different string wrap around different rod surfaces.

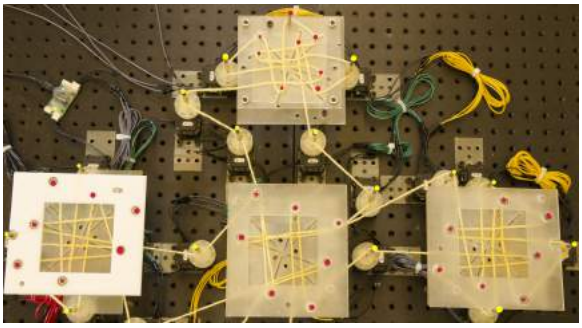


**Figure 4.39:** The initial layout of the fixtures and the rotating rods for tying Ruyi knot. Powered rods are labeled with yellow dots, and all other rods are labeled with red dots.

# Chapter 5

# String manipulation

Previous proposed fixture based methods approached the knot tying problem with passive controls. The string is partitioned into a sequence of line segments, so that we can model and control the string easily.

However, to force the string into line segments, many contact points are needed, and for each new knot we need to tie, a new fixture needs to be build. For generalized string manipulation tasks, such specialized devices may not be applicable in all scenarios.

Most robotic knot tying research conducted in the past three decades uses the classic set up: generic robot arms with sensing feedback. Such setup do have the advantage of being general, so that the knot tying skill and the capability of the manipulating string can be transferred into other applications.

In this chapter, we explore a simplified string manipulation control strategy, without simulating the string deformation. At each time step, we compute a Jacobian matrix that roughly represents the string deformation, and use the matrix to compute the appropriate controls for the robot arm.

We first applied this control strategy in threading string through small openings in work space, demonstrating the robustness of the approach under faulty sensing feedbacks.

A knot tying task can sometimes be thought of as controlling string to form different loops, and threading the tip of the string through these loops formed by the string. Using the proposed control strategy, we expect the knot tying task can be simplified into a few subroutines that can be achieved with high success rate.

In this chapter, we first introduce the control strategy, and apply the strategy to thread string through small workspace openings, both in simulation and using a Da Vinci surgical robot arm. Then, a few simple knots are arranged using the Da Vinci robot arm, with the same set of generic fixtures. The proposed control strategy combined with generic fixtures serves as the fundamental blocks for autonomous knot tying with minimum sensing, which is part of the future work.

## 5.1 Introduction

When we look at how humans tie knots, the majority of the procedure seems to include threading a segment of string through some established loops. These insertions forms over- and under-crossings. When we decompose the task, it seems the elemental task is to thread a piece of string through a tight tolerance workspace loop. The same description fits the task of threading a needle.

A tight tolerance insertion task resembles the classic robotics problem, "peg-in-hole" [Don90, LPMT84]. There are elegant solutions for solving "peg-in-hole" problem. However, when we deal with deformable objects, the deformations were not characterized in previous research about "peg-in-hole". Common approaches to deal with deformation objects usually includes simulation of the deformation. The simulations evolves complex string model are usually slow, and still not as accurate as we want them to be.

Instead, we propose an online control based approach that does not rely on simulation to insert strings into tight-tolerance spaces. An approximate Jacobian is used to compute the appropriate control that should be applied to the string based on the desired motion of the string. Desired motion of the string is calculated using a controller that is calculated using artificial vector fields. Combine the use of approximate Jacobian and the vector fields, our approach conservatively applies control to achieve desired motion. The motion of the string is assumed to be quasi-static, so the equilibrium configuration of the flexible object is completely determined by the external forces (gravity, and the configuration of the grippers which are used to control the string).

## 5.2 Diminishing Rigidity Jacobian

Let us assume the string is controlled by a group of $k \geq 1$ grippers with configuration(s) $q$, which will lead a group of reference points along the string to move in desired directions. The locations of these reference points in world frame are represented by a vector $\mathcal{P}$. Given a function $F$ that computes the location of reference points based on the configuration of the grippers, we have

$$\mathcal{P} = F(q), \tag{5.1}$$

Since the intent is to control the string locally, we will omit the fact that there might be multiple equilibrium configurations of the string for a given $q$.

A Jacobian of $F$ may locally relate the changes in the gripper configuration and the change of the reference points' locations,

$$\dot{\mathcal{P}} = J(q)\dot{q}. \tag{5.2}$$

If sufficient grippers are used to precisely control each of the reference points, then a set of gripper velocities (configuration space) exists for a particular velocity of the reference points. We could compute such a gripper velocity directly by solving Equation 5.7—a equation we will derive later based on Equation 5.2—for $\dot{q}$ if we knot $J$. However, if the string is under controlled (not sufficient grippers), the pseudo inverse $J(q)^+$ can be used to find a best-possible motion (in least-squares sense).

Most known techniques to compute $F$ or $J$ accurately require detailed modeling. However, it is possible to *approximate* the Jacobian of them *without* a model. In 2013, Berenson [Ber13] proposed an approximation technique that assumes that the deformable object behave "more rigid" near the grasping points. In his experiments, he discovered that the motion of a reference point due to a gripper movement diminishes exponentially with its distance from the gripper. Let the Jacobian calculated using this assumption be a *diminishing rigidity Jacobian*, denoted by $\tilde{J}(q)$. A short review of the calculation for $\tilde{J}(q)$ is presented. More details can be found in [Ber13].

Let us start with two points $p_i, p_j \in \mathcal{P}$ with geodesic distance $d(p_i, p_j)$ in between. By geodesic

distance, we mean the distance between these two points when the string (or other deformable object) is undeformed (a straight line). Denote the point grasped by the gripper $g$ on the string that is closest to the $i$th point of $\mathcal{P}$ by $c(i,g) \in \mathcal{P}$. Let $w(i,g)$ be the rigidity weight of the $i$th point with respect to gripper $g$, which is calculated by:

$$w(i,g) = e^{-k(d(p_i,c(i,g)))}, \tag{5.3}$$

where $k$ is a constant determined experimentally for the system (for details, see [Ber13]).

We take the translation component of $\tilde{J}(q)$ for the $i$th point with respect to gripper $g$ to be simply

$$\tilde{J}_{trans}(q,i,g) = w(i,g)\mathbf{I}_{3\times 3}. \tag{5.4}$$

The configuration of a gripper in the world frame can be represented by a translation vector $v^g$ and a $3 \times 3$ rotation matrix $\mathbf{R}^g$. We can then define the rotation component of the approximate Jacobian:

$$\tilde{J}_{rot}(q,i,g) = w(i,g)[\mathbf{R}^g[1] \times r, \mathbf{R}^g[2] \times r, \mathbf{R}^g[3] \times r], \tag{5.5}$$

where $r = (c(i,g) - v^g)$.

Then $\tilde{J}(q,i,g)$ for point $i$ with respect to gripper $g$ is

$$\tilde{J}(q,i,g) = w(i,g)[\tilde{J}_{trans}(q,i,g), \tilde{J}_{rot}(q,i,g)]. \tag{5.6}$$

We can then define $\tilde{J}(q)$ by combining the Jacobians for all points and all grippers into a single matrix. For a given desired motion $\dot{\mathcal{P}}$ of the reference points, the corresponding $\dot{q}$ can be calculated by multiplying the pseudo-inverse of the diminishing rigidity Jacobian $\tilde{J}(q)$:

$$\dot{q} = \tilde{J}(q)^{+}\dot{\mathcal{P}}. \tag{5.7}$$

Additionally, the calculated term of $\dot{q}$ can be combined with obstacle avoidance terms, calculated in Equations 11 to 13 in [Ber13]; when multiple grippers are used, a stretch avoidance term can also be added to automatically correct excessive stretching. We will omit the details of these terms, which can be found in [Ber13].

A desired motion of the reference points for the insertion task can be generated using a sequence of vector fields. These vector fields, if designed correctly, should guide the reference points on the string through these tight-tolerance spaces. The first reference point selected is the tip of the string, since if the tip can go through the tight tolerance space, the rest of the string can follow and complete the insertion task.

The vector fields that are used to guide the string, however, should satisfy certain properties. First, we expect all integral lines of the vector field go through the small opening, such that following the integral curve can successfully insert the string regardless of the initial configuration. These integral curves should also go through the small opening in the same direction, which coincide with the insertion direction. A closed form solution everywhere for the vector field would be preferred to ensure the fast calculation of $\dot{\mathcal{P}}$.

One field that obeys these properties is a magnetic field induced by a current carrying loop. For simplicity, we let the wire be a planar circle (referred as a loop in the following context). An example of an induced vector field is shown in Figure 5.1.

For an arbitrary point $\mathrm{p}$ in the workspace, and a fixed current-carrying loop $S$, the magnetic force at $\mathrm{p}$

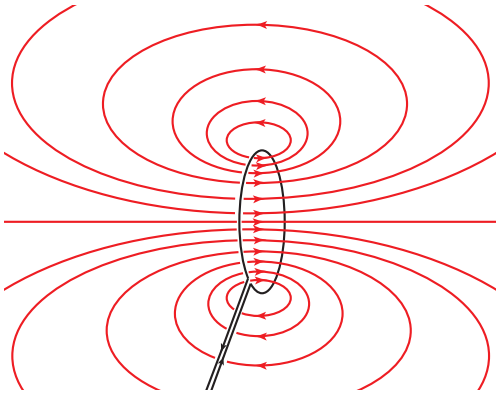


**Figure 5.1:** An example of magnetic field induced by a current carrying planar circle loop.

can be calculated as follows:

$$F(\mathrm{p}) = \frac{1}{4\pi} \int_S \frac{(x - \mathrm{p}) \times dx}{||x - \mathrm{p}||^2}. \tag{5.8}$$

This vector is used as the desired motion for the tip of the string, which we denote as $v_1$.

It is convenient that the magnetic field shown in Figure 5.1 describes reasonable motions for the tip of the string globally. Even if the tip reaches the wrong side of the insertion loop, the field lines drive the tip of the string back around for another attempt using the same vector field. In some sense, the length of the integrated field line (from arbitrary point to the intersection with the interior of the loop) approximates the real distance to the goal.

In this approach, we use these current-carrying rings to induce the magnet fields. Currently, the rings are still manually placed. However, a dynamic placement is possible, and will be explored in the future.

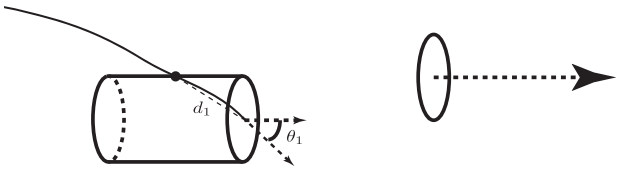
To ensure the success of the insertion, we also expect the orientation of the tip of the string should line up with the insertion direction. To control the orientation, a second reference point must be selected. Different level of deformation may present when different ropes are used, therefore we select this second reference point dynamically using a cylinder centered at the tip with radius $S$. Let the second point be the first intersection between the string and this cylinder. Figure 5.2 shows an example of dynamically choosing the second reference point.

The motion of this second reference point needs to be calculated "on the fly" to minimize the difference between the string tip orientation and the insertion direction. Let the two reference points have locations $p_1, p_2$, and the distance between them be $d$. Let the calculated motion of the tip be $v_1$, and the normal of the loop to penetrate be $n$. We choose the motion for the second point to be

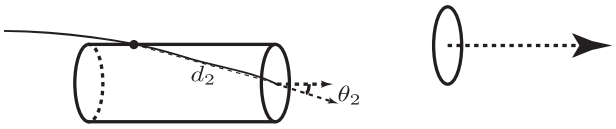
$$v_2 = p_1 + v_1 - d \cdot n - p_2, \tag{5.9}$$

so that if the tip moves exactly $v_1$ in one time step, and the second reference point moves $v_2$, then the orientation of the line segment between the references will be parallel to the insertion direction. Together, $v_1$ and $v_2$ forms $\dot{\mathcal{P}}$.

There are many sources of error in the insertion process. The expected motion is calculated using a




**(a)** A configuration where the second point is close to the tip of the string.




**(b)** A configuration where the second point is further from the tip of the string.

**Figure 5.2:** Dynamic selection of the second reference point; $d_1 < d_2$, and $\theta_2 < \theta_1$.

Jacobian that is only approximate, and we expect errors in sensing and control. Following the designed vector field, even with error, the controller will keep inserting the string. So, given sufficient time, we expect the process to succeed eventually. However, to minimize the number of misses, certain principles apply in the design of the virtual loops.

During experiments, we found that sometime after a failed attempt of insertion, the vector field keeps guiding the string to the edge of the small opening. This is found to be an artifact of the magnetic field, since the integral lines near the boundary of the opening always loops around near the boundary. Notice that in the presence of error, it is easily possible for the string to pass outside of the target loop (miss the target), since some of the field lines pass very close to the boundary of the loop. In order to ensure that the controller does not repeatedly "re-try" the insertion task near the boundary of the loop, we test if the string passes through a larger disk. As long as the small opening is larger than this larger disk, we may in this case consider the insertion to have succeeded. At the same time, the magnetic field is induced using a current carrying loop that is some fraction of the size of the small opening. A smaller loop ensures that the the induced magnetic field lines pass very close to the center of the opening; a larger loop allows more gradually-bending field lines. Figure 5.3 shows such an example; the smaller loop induces the vector field the string follows, while the larger disk is used to detect penetration.

The release and re-grasp of the string may be needed during the insertion task for two reasons. First, one primary source of error during the insertion task is the estimation of the Jacobian. The error increase as the distance between the gripper and the references increases. To reduce the effect of the estimation of the Jacobian, we can re-grasp closer to the reference points when we found that the calculated expected motion

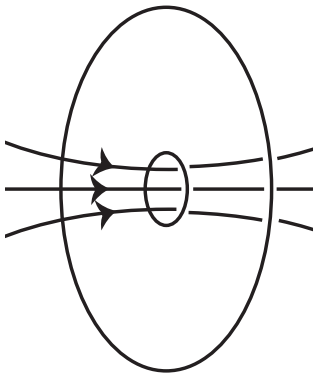


**Figure 5.3:** A loop used for guiding the string, and a disk for detecting successful insertion.

is too far from the desired motion.

Second, since the grippers themselves cannot go through the small openings (otherwise the task would not be challenging), re-grasp may be needed to push or pull the string through the opening completely after the initial contact.

Let us consider a simple version of the re-grasping problem. Given a string modeled as a sequence of $n$ links, a gripper may grasp the center of any link.

Since we are only controlling two reference points during the execution in this work, more than one gripper could result in stretching the string, thus violating the constraints; for simplicity, we use a single gripper for the insertion task.

Two approaches were used to find whether re-grasp is needed and where should the gripper re-grasp.

First strategy is very simple: grasp the first link after the reference points that are accessible. This minimizes the error of approximating the Jacobian, however, it may result in frequent re-grasping near the obstacles when inserting the string.

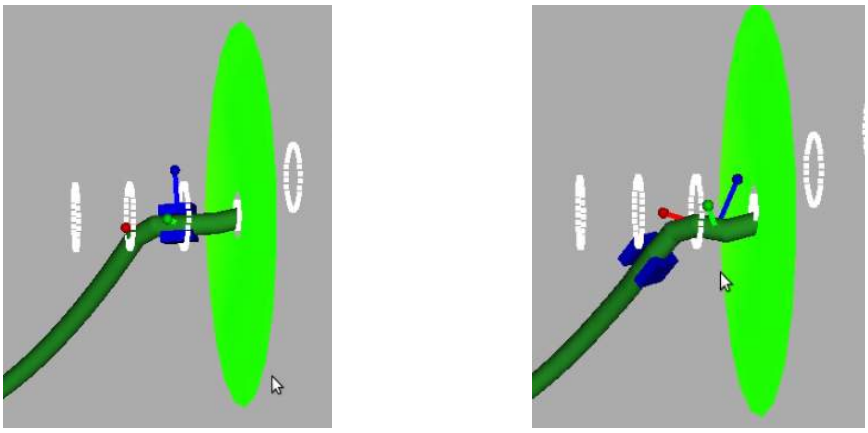
The second approach we tried attempts to take advantage of the fact that the string tends to "droop" in the direction of gravity. If the target loop is lying more flat with respect to the direction of gravity, we expect the insertion task to be easier, and can grasp the string farther away, leading to fewer re-grasps. We used the following heuristic to compute the grasp location. We compute the angle between the normal to the target loop and the direction of gravity, $\theta$. We choose the grasp point to be a distance of $l/\theta$ beyond the final reference point, where $l$ is a constant that might be chosen experimentally. This switch strategy is referred as a *heuristic switch strategy*.

We expect the walls of a physical small opening to have some depth. When should we decide that the string has penetrated the opening? Once the tip of the string has penetrated the target ring, then either the tip may be re-grasped on the other side of the opening, or it may not be (due to occlusion). If the tip is re-graspable, then we do the re-grasp, and declare success. Otherwise, we choose a new first reference point at the intersection of the target ring with the string, and continue pushing the string.

We present the entire online control strategy in Procedure Insertion strategy, which may be used to guide string through one or more small openings. The loop and disk configurations are inputs. We denote the actual motion of the rope as $\tilde{\mathcal{P}}$.

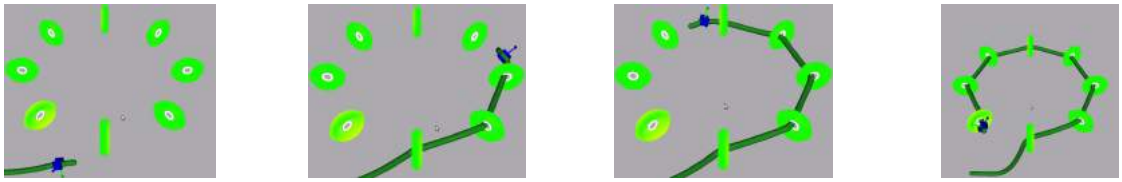
**Procedure** Insertion strategy

Find gripper location using the normal of first target loop and gravity;
**while** *last loop not penetrated* **do**
  Compare the actual motion $\tilde{\mathcal{P}}$ of the reference points and the expected motion $\dot{\mathcal{P}}$;
  **if** $|\dot{\tilde{\mathcal{P}}} - \dot{\mathcal{P}}| >$ *tolerance* **then**
    Re-grasp closer to reference points;
  **if** *string penetrates a target disk* **then**
    Calculate the gripper location $g$ using the normal of next loop and gravity;
    **if** *location* $g$ *is available* **then**
      Re-grasp at $g$;
      Activate next loop;
      Set the first reference point to be the tip;
    **else if** *Gripper is too close to the obstacle* **then**
      Set the first reference point to be the first point on the string that has not yet penetrated current target disk;
      Find gripper location $g'$ using new normal of active loop;
      Re-grasp $g'$ away from the reference point;
  Recalculate second reference point's location;
  Find the expected motion of reference points $\tilde{\mathcal{P}}$;
  Compute the approximate Jacobian $\tilde{J}$;
  Compute desired motion of string $\dot{\mathcal{P}}$;
  Calculate the motion of gripper $\dot{q}$; apply for $\Delta t$ ;



**(a)** Before apply heuristic switching strategy. **(b)** Before apply heuristic switching strategy.

**Figure 5.4:** Re-grasping strategy that considers gravity. Left: The configuration before re-grasp. Right: the configuration after re-grasp.



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

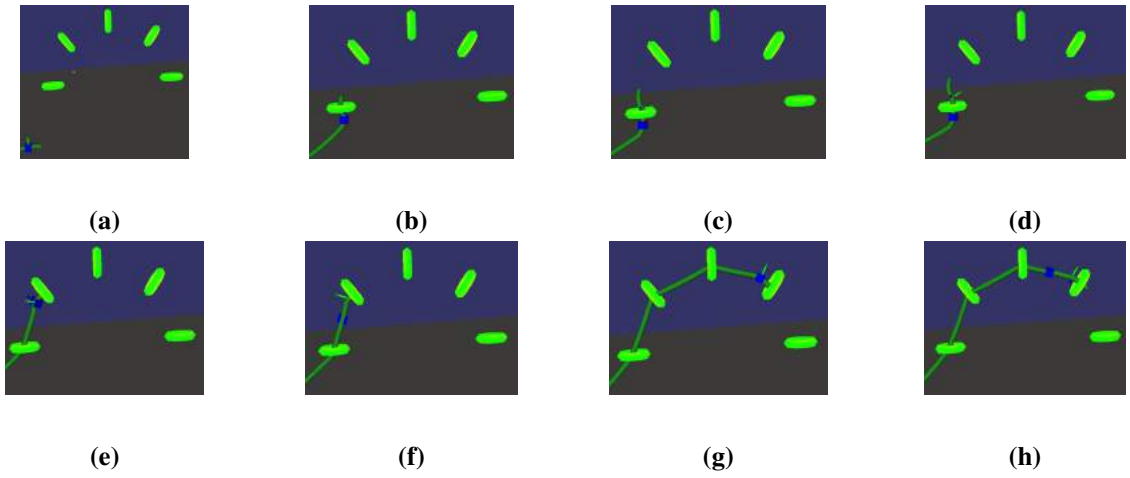
**Figure 5.5:** Simulation results of threading a sequence of loops, Figures 5.5a to 5.5d show several snapshots from the simulation.

To test our approach, we first conducted experiments in simulation using the Bullet Physics library [Bul] and constructed the string using a sequence of capsules (hinged short links). The simulator is used as a black box for our evaluation; no simulation model or parameters were known to our method.

We will first describe several experiments in simulation that we used to test the ability of this method to thread one or more small openings in sequence, with various loop orientations. We also tested our method's ability to deal with noise (added zero-mean Gaussian noise in sensing) and a changing environment (a large move of the target opening during execution) in simulation. Finally, we conducted physical experiments with a Da Vinci surgical robot, to insert yarn into washers of various sizes, with arbitrary initial configuration.

Figure 5.4 shows a close-up of a simulation experiment; white dotted lines show the loops. Figure 5.5 shows an example of threading string through several openings in sequence, with penetration normals parallel to the x-y plane. Figure 5.6 shows an example with normals that are not parallel to the x-y plane.

During simulation experiments, we observed that the gripper tends to grasp near the tip during most



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

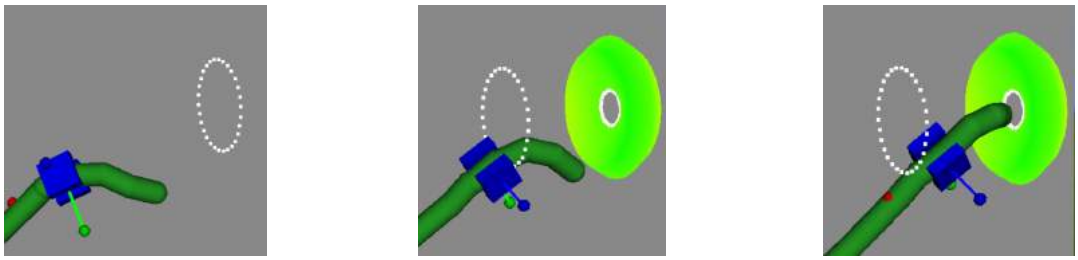
**(e)** **(f)** **(g)** **(h)**

**Figure 5.6:** Simulation results of threading a sequence loops with various ring orientations. Figures 5.6b to 5.6d show the frequent switches near the first loop, due to gravity. The rest of the figures show re-grasping at further and further locations.

of execution, leading the tip of the string to behave almost like a rigid body. Even though this strategy is effective for threading purposes, it does not demonstrate the advantage of our approach for deformable objects.

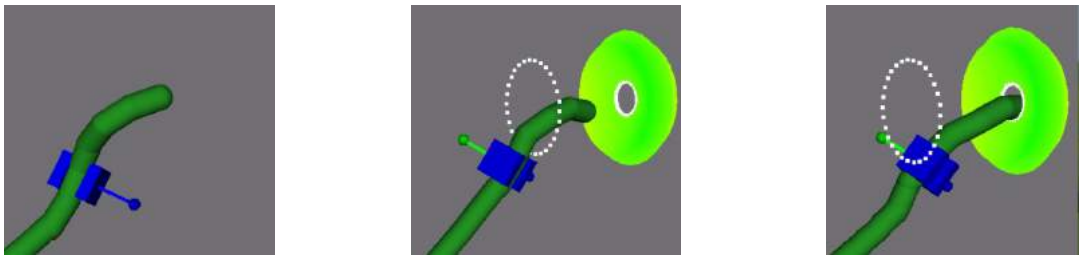
We then enforced that the grasping location be further from the tip during insertion. In such experiments, the string demonstrates clear deformation. The controller was still able to accomplish the insertion task consistently, even though sometimes it missed the opening on several times, perhaps due to un-modeled un-simulated string deformation. Some configurations of the string during experiments are shown in Figures 5.7 and 5.8.

To test robustness to error, we conducted simulation experiments adding zero-mean Gaussian noise,



**(a)** **(b)** **(c)**

**Figure 5.7:** Simulation results of threading while enforced grasping far from reference points, showing the deformation of string during execution.



**(a)** **(b)** **(c)**

**Figure 5.8:** Simulation results of threading while enforcing grasping even further from reference points, showing more deformation of string during the execution.

| noise (variance) | Mean Completion time | completion time std. dev. | Mean # of misses | # Trials Failed |
|---|---|---|---|---|
| 0.01 | 7.82 | 0.39 | 0 | 0/40 |
| 0.03 | 8.03 | 0.41 | 0 | 0/40 |
| 0.05 | 8.92 | 0.44 | 0 | 0/40 |
| 0.1 | 10.24 | 0.37 | 0 | 0/40 |
| 0.3 | 12.47 | 4.36 | 0.025 | 0/40 |
| 0.5 | 14.659 | 5.57 | 0.05 | 0/40 |
| 1 | 29.12 | 16.07 | 0.3 | 0/40 |
| 1.5 | 55.42 | 39.33 | 0.57 | 0/40 |
| 2 | N/A | N/A | N/A | 40/40 |

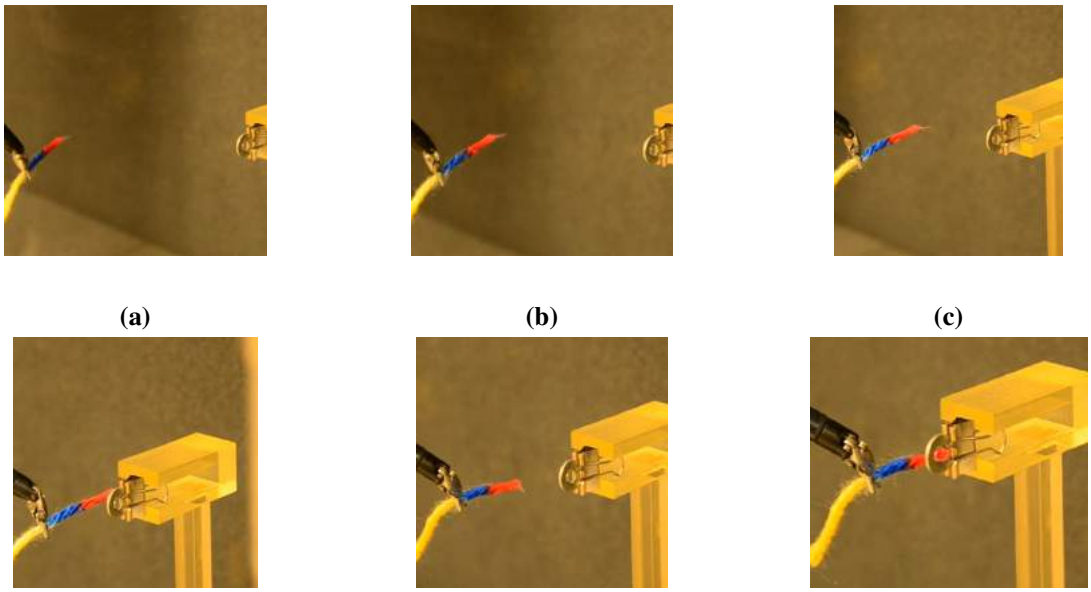
**Table 5.1:** Statistics evaluating the effect of noise on a simulated insertion task.

observed the execution of the task, and time-to-completion for successful insertion with different given noise variances. Average times to completion acquired over 40 trials are shown in Table 5.1. The number of misses was counted by the author watching the simulation. (A failure is reported after 180 seconds without penetrating the target; this only occurs in the presence of extreme noise.) Step size used was 0.05 unit.

Even with sensor noise with 0.3 unit variance applied to every coordinate of the reference points at each time step, our method still succeeded in a mean time of 12.47 seconds. (For comparison, recall that the opening has radius 0.3, and the string has radius 0.2 in our simulation experiments.) As we increased the noise variance, the number of misses increased slowly, but the time to completion increased rapidly. When we added noise with variance 2, the gripper essentially moved randomly and did not penetrate the target.

We also experimented with moving the hole during the execution of the task to study robustness of the approach with respect to unexpected major changes in the environment. Figure 5.10 shows an example of how the controller followed the vector field after the target was moved.

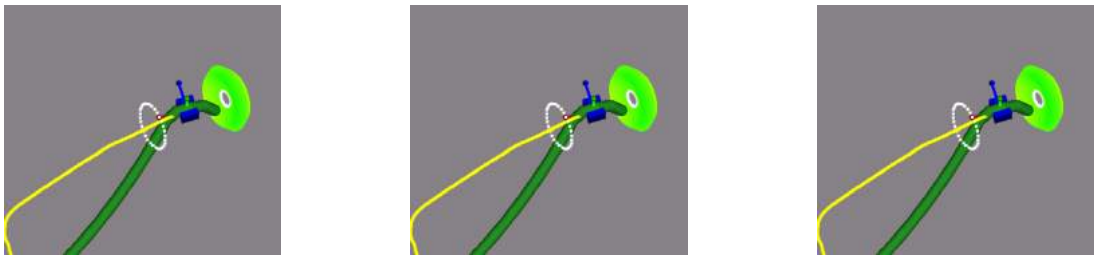
We also conducted physical experiments with one arm of a Da Vinci robot, with seven degrees of freedom. We used two HD webcams to track the location of the string. One webcam was placed over the



**(a)** **(b)** **(c)**

**(d)** **(e)** **(f)**

**Figure 5.9:** Experiment using Da Vinci robot with the smallest washer. Washer diameter was $4.9$ mm, and the yarn diameter was $3.5$ mm. The first three frames show how the controller rotates the gripper to align the yarn with the insertion direction, but the first pass missed the washer, as shown in the fourth frame. Then, the controller made a second attempt (with no intervention) and successfully threaded the washer in the last frame.



**(a)** **(b)** **(c)**

**Figure 5.10:** Example of threading the needle after the target moved during execution (second frame). Gripper trajectory is shown in yellow.

**Figure 5.11:** The three washers used in experiments, the yarn next to a nickel for scale reference.

workspace, while the other was placed facing the $x$-$z$ plane of the workspace. In the experiments, we simply tested the performance of the insertion, without the re-grasping which was shown in previous simulation experiments. The Da Vinci robot is controlled by giving the locations of the end-effector at every time step, and Inverse Kinematics was used to calculate the joint angles.

We used three different washers with different sizes (shown in Figure 5.11), and attempted to thread a segment of yarn through the washers using the Da Vinci robot. The washer was placed on a stand that is not rigidly attached to the ground; thus, any aggressive motion towards the washer would knock the target over.

The Da Vinci robot successfully threaded all three washers. The inner diameter of the smallest washer was $4.9$ mm, while the yarn had diameter $3.5$ mm. Images from one of the experiments are shown in Figure 5.9. The robot successfully threaded even the smallest washer with no more than $4$ passes on average.

## 5.3 Knot tying using Da-Vinci robot

One disadvantage of the previously introduced fixture based approach is the need to build specialized fixture for each new knot. The use of fixtures, however, still have many advantages, such as providing many contact points to support string geometry, and provide anchor points for manipulation, etc.

In this chapter, we show that generalized fixtures can be used to tie different knots when a dexterous arm, such as a Da Vinci robot arm is used. Combining with the simple control strategy that do not re-

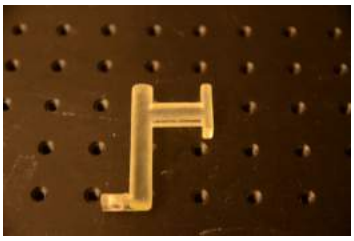


**Figure 5.12:** One unit of the generalized fixture used for tying knots.

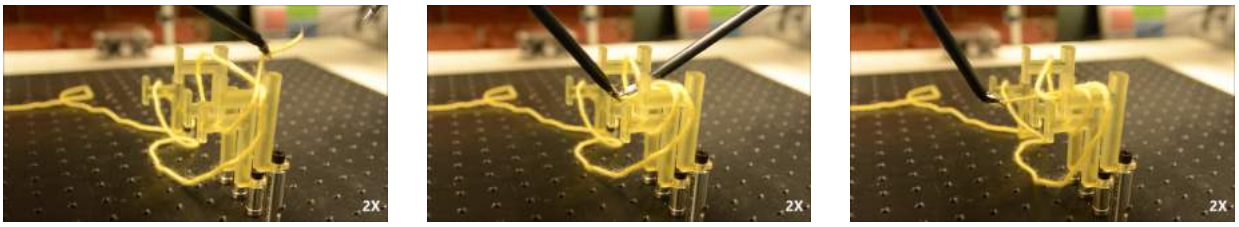



**Figure 5.13:** Arranging an overhand knot using a Da Vinci robot arm, around generalized fixtures.

quire simulation, we hope these two components can lead to simple autonomous knot tying with minimum sensing.

Each unit of the generalized fixture is an upside down L shaped bracket, with three short extensions located at the turn and the end of this L shape, as shown in Figure 5.12. The short extensions are used to prevent string from sliding away from the fixture due to the tension along the string during arrangement.

For different knots, we use different number of unit fixtures and arrange them at different locations based on the knot geometry. In this preliminary work, the authors manually lays out the fixture, and we consider the automated layout of the fixtures as part of future work.

The location and the size information are collected to compute the path to drag the string along. Following the computed paths, the string is supposed to wrap around the generalized fixtures taut, preventing deformation. The paths are given to a Da Vinci robot arm to carry out. We used the arm to tie three different knots: an overhand knot, a shoelace unknot, and a surgeon's knot.

The experiments showing the arm dragging string around fixtures are shown in Figures 5.13, 5.14, and 5.15. Different number of re-grasps are needed for the knots. The string is stretched around the generalized fixtures with minimum deformation, presenting the loops that the string tip needed to thread through.

Throughout the experiments, we did not use visual sensing feedback, since the generalized fixtures are at known locations and the paths the arm need to follow are computed without ambiguity. The re-grasps can also be achieved without sensing in these specially designed experiments. We utilize the orientation information of each gripper, and assume the string tip is not dropped during the knot arrangement. Then, at the time for re-grasping, two grippers are designed to face the same orientation at nearby locations in the workspace, utilizing the fact that locally the string is minimum deformed near the grasp points. The assumption follows the same idea of the design of the diminishing rigidity Jacobian. At this configuration,

**Figure 5.14:** Arranging a shoelace unknot using a Da Vinci robot arm, around generalized fixtures.

**Figure 5.15:** Arranging a surgeon's knot using a Da Vinci robot arm, around generalized fixtures.

two grippers can perform re-grasps with no-sensing. The re-grasps fails sometime during the experiments, but we did not count the success rate as this experiments only meant to demonstrate the fact that generalized fixtures can be used to tie different knots. We intend to work on fully autonomous knot tying in the future.

As a proof of concept, the experiments demonstrate that the generalized fixtures are useful in tying different knots, and simplifies the knot tying task into a sequence of threading tasks at known locations, where after each threading a re-grasp needs to be performed. As the threading task can be performed with high success rate even under faulty sensing using the previously proposed control strategy, we hope by combining the two components together, we can achieve autonomous knot tying with minimum sensing.

However, problems are observed during the experiments. As we intend to wrap string taut around the generalized fixtures, after a few contacts with the generalized fixtures, the friction becomes so significant so that the cable driven Da Vinci robot arm have difficulty to drag the string without the tip slipping out of the gripper. Even though this is partially due to the fact that the grippers are old and cannot grasp very tight, the existence of friction is clearly observable. As the knots we would like to tie becomes more complex, the problem will cause more trouble.

When humans manipulate string to wrap around rigid bodies in real life, similar problem with friction also exists. Most of the time, human "pokes" the string to loosen it locally to yield extra string between the tip and the nearest contact point with the rigid body. The poking usually repeats many times during the manipulation.

As part of the future work, we also intend to study how to address the friction problem between the string and the fixtures, possible through the poking with the robot arm as when the string is wrapped taut around fixtures, different segments are at known locations so that little sensing is needed for the arm to poke

the string. We are also looking for alternative solutions.

## 5.4 Conclusions

In this chapter, we present a simple active control strategy to manipulate string without simulating the string deformation. A Jacobian matrix is computed to generate the motion of grippers to manipulate the string into desired configurations. The approach shows positive results, and is applied to thread string through small workspace openings. The threading has a high success rate even under noise sensing, as demonstrated in both simulation and physical experiments with Da Vinci robot.

Using the same robot arms, we demonstrated the usefulness of generalized fixtures in assisting knot tying, by reducing string deformation. Future work will combine the two components together to work towards autonomous knot tying with minimum sensing, while addressing several challenges in both components, including the automated placement of virtual vector fields for the control strategy, the placement of the generalized fixtures, and how to reduce the effect of friction between string and generalized fixtures.

# Chapter 6

# Knot folding

In this chapter, we will study the knot tying from a more theoretical point of view: how many grasp points (fingers) are needed to fold a given knot?

In the previous chapters, we used fixtures to assist knot tying process. Even though we can arrange and eventually tighten different knots, we are still unclear about the reason why knot tying has traditionally considered a difficult task in string manipulation. In previous two chapters, we used fixtures to contact string at multiple points to reduce the deformation, and thus tied different knots without the interference of the string deformation. But exactly how many of those contact points do we need to tie a given knot?

In the game of cat's cradle, string is stretched tightly between fingers; as the fingers move, the configuration of the string changes without ambiguity. We adapt similar approach, stretching the string between adjacent grasp points (fingers) to fully control the configuration and motion of the string to reduce string deformation. We assume that all contacts are bilateral and have infinite friction so that they cannot slide along the string. We do not allow any re-grasp for any finger in order to understand how many contacts are needed in total to arrange a knot.

The theoretical complexity of the knot tying process has begun to be studied from the perspective of topological structure (*e.g.*, sequences of *Reidemeister moves* [HKK91]), but the details of the manipulation itself are left to the imagination. By forcing the string to take on polygonal shape, we are able to study complexity, while also providing specific finger placements and motions to achieve grasping and tying. The knots formed by polygonal arcs are called polygonal knots.

Within knot theory, the study of polygonal knots do exist, however, it does not investigate the number of line segments needed to represent the knot, which is the focus of this chapter.

We start with fundamental concepts of polygonal knots, and analyze its properties. We will then present algorithms to first grasp then fold the knot using a bounded number of fingers w.r.t. the number of crossings to the given knot.

## 6.1 Polygonal knots and polygonal knot diagrams

**Definition 2** *[Arm83] A polygonal knot is a knot whose image in $\mathbb{R}^3$ is the union of a finite set of line segments.*

**Definition 3** *[Arm83] A tame knot is any knot that is equivalent to a polygonal knot.*

Since the tame knot is the equivalent to a polygonal knot, it only has a finite number of crossings. Knots that are not tame are called *wild knots*, which also have infinite many crossings. In this work, we are focusing on the knots that are tame. For simplicity, we will drop the word tame in the rest of the text.

A regular projection of a polygonal knot is a set of planar line segments. The corresponding knot diagram will be called *polygonal knot diagram*. Then, if a polygonal knot diagram contains $n$ links, in addition to the Gauss code, the knot can be fully represented by the relative relations between the end points of these $n$ links.

Since a polygonal knot diagram with $k$ crossings is the result of a regular projection, it is a 4-planar graph $G = (V, E)$ with straight line edges where $|V| \geq k$. On the regular projection, we trim the first and the last link of the polygonal arc, such that the first and last crossings become the end points of the polygonal arc. Then among all the vertices of the graph, $k - 2$ vertices have degree four, and two vertices have degree three. These $k$ vertices represent crossings. All other vertices have degree 2.

In order to relate the vertices to the crossings in the Gauss code which are ordered, we will further require the graph to be directed. For a vertex with degree 2, it will not appear in the Gauss code, and it can be fully defined by its adjacent vertices. For the rest of the vertices, each of them appears in the Gauss code twice. In order to distinguish the two different times the edges pass through the same vertices, we will denote the directed edges using the end points (crossings) with the corresponding super script in the Gauss code.

For the projected polygonal knot diagram, it separates the projected plane into different regions. Let us call the region that is connected to infinite far away (which means an open region) the exterior face, and all the other closed region interior faces. Let us refer a connection between two adjacent crossings on the Gauss code a *c-path* (crossing path), which may be one or a sequence of line segments. For a c-path, if it contacts both the exterior face and at least one interior face, it is called an *exterior c-path*.

On every knot, we know that each crossing will appear twice on a Gauss code. Let us define a *knot unit* as a connected subsection of Gauss code where all the labels in the subsection have appeared twice. For example, let us consider a sounding line that contains three overhand knots, with Gauss code $G = \{1^+, 2^-, 3^+, 1^-, 2^+, 3^-, 4^+, 5^-, 6^+, 4^-, 5^+, 6^-, 7^+, 8^-, 9^+, 7^-, 8^+, 9^-\}$. This sounding line contains three knot units, $G_1 = \{1^+, 2^-, 3^+, 1^-, 2^+, 3^-\}$, $G_2 = \{4^+, 5^-, 6^+, 4^-, 5^+, 6^-\}$ and $G_3 = \{7^+, 8^-, 9^+, 7^-, 8^+, 9^-\}$. We call a c-path a *bridge* if it connects two knot units.

**Lemma 1** *All the exterior c-paths of a knot unit form a Jordan curve (simple closed curve).*

*Proof:* On a knot unit, let us consider the first crossing and last crossing, and ignore the links before first and after last crossing. Then, these two crossing vertices have degree 3. Let us denote them as $v_0$ and $v_n$. They correspond to the first and the cast crossing on the Gauss code. Since this is a section of a knot diagram, the graph contains a Eulerian path starting from $v_0$ tot $v_n$.

If a c-path is only adjacent to exterior face, the reader can verify that it must be a bridge. Since the knot unit do not contain a bridge, no c-path on the projected polygonal arc of the knot unit will adjacent only to the exterior face. Therefore, there exist a set of connected c-paths $\partial E$ that are adjacent to both exterior face and an interior face. So, these exterior c-paths are closed.

What is more, since all the 3 or 4 degree vertices are crossings (intersection of edges on the projected plane), and all such intersections have been recorded by these vertices, so $\partial E$ do not self-intersect.

Since all the c-paths in $\partial E$ are closed and have no self-intersection, they form a Jordan curve. ■

**Lemma 2** *Given an arbitrary knot with $k$ crossings, and a corresponding planar polygonal knot diagram, there are $k$ interior faces formed by this polygonal drawing.*

*Proof:* Let us assume there are in total $m + k$ vertices in the planar polygonal drawing of the knot. In this drawing, every crossing is a vertex of degree 4 or 3, and let us add additional $m$ vertices with degree 2. Then, in total, we have $m + 2k - 1$ edges related to this drawing.

For each crossing $k$, it has 3 or 4 degrees, so each of $k$ crossings is associated to 3 or 4 edges. What is more, there are only two vertices with degree 3. For each of the additional $m$ vertices, each of them is associated with two edges. Add all of them together, we have $2 \times m + 2 + 4(k - 2)$ edges associated with all the vertices. However, each edge has two end points, so each of them has been counted twice, so, we have in total $(2 \times m + 2 + 4(k - 2))/2 = m + 2k - 1$ edges.

Let us denote the number of faces on this polygonal drawing as $|f|$, so according to Euler's theorem, $|v| + |f| = |e| + 2$. Then, we have $|f| = m + 2k - 1 + 2 - (m + k) = k + 1$ faces. However, one of all the faces is the exterior face, so there are only $k$ interior faces. ■

## 6.2 Knot grasping

In order to use multiple independent controllers to control the string while forcing the string to take the shape of a polygonal arc, we need to know how many controllers we need, and where to place them. However, this question is too complex to answer directly. Since our assumption is that we use controllers to grasp string and fully span the string between two adjacent controllers, we can then view the string as a polygonal arc.

In the spirit of working with polygonal arcs, we will call the process of manipulating the polygonal arc to form a knot structure as *folding a knot*.

In order to be able to fold a knot, we need to be able to use a polygonal arc to represent a knot. In this section, we will discuss how many line segments are needed on a polygonal arc to represent a knot.

We start with a polygonal knot diagram. If we can represent the polygonal knot diagram for a given knot with certain number of line segments, we can easily project it back to 3D, by using more (but a bound number of) line segments, whereas directly determining the number of line segments needed in 3D is complicated.

Given an arbitrary knot, with Gauss code $G$, we know how many crossings it contains. What is more, there always exist a corresponding polygonal knot diagram, because the original polygonal knot must have such a projection.

First, let us consider given a sequence of polygonal arc of length $n$, how many intersections can we create? Remember, on the polygonal knot diagram, a crossing has degree 3 or 4. Given a sequence of line segments, only intersection creates a vertex with degree more than 2.

**Lemma 3** *For an arbitrary knot with $k$ crossings, at least $\left\lceil \frac{3+\sqrt{8k+1}}{2} \right\rceil$ sequential line segments are needed to plot the corresponding polygonal knot diagram.*

*Proof:* Based on the line arrangement [AS98], $n$ independent line segments (not sequential line segments) can create at most $n(n-1)/2$ intersections. However, for a sequence of $n$ line segments (polygonal arc), there is one common end point between the two adjacent line segments, which results in $n-1$ common end points. For the $n(n-1)/2$ intersections, $n-1$ of them will be the common end points, therefore, the maximum number of intersections for a sequence of $n$ line segments is

$$n(n-1)/2 - n - 1 \tag{6.1}$$
$$\Rightarrow \quad (n^2 - n - 2n + 2)/2 \tag{6.2}$$
$$\Rightarrow \quad (n-1)(n-2)/2. \tag{6.3}$$

Given $n$ sequential line segments, at most $(n-1)(n-2)/2$ intersections (crossings) can be created. In order to make it a knot diagram, we need to have $(n-1)(n-2)/2 \geq k$. By solving the equation, we can find $n \geq \frac{3+\sqrt{8k+1}}{2}$. ■

Essentially, Lemma 3 gives a necessary number of sequential line segments for arbitrary knot with $k$ crossings.

Naturally, the next question we would like to answer is, does there exist a sufficient number of line segments to plot polygonal knot diagram for arbitrary knot with $k$ crossings? It turns out there is.

Before we answer the question, we need to further define the following few concepts besides the exterior c-paths and interior faces. Let us define an *exterior crossing* as a crossing as the common end point of two adjacent exterior c-paths, and all the other crossings as *interior crossings*. If a connection between two crossings are not an exterior c-path, it is defined as an *interior c-path*.

**Lemma 4** *For an arbitrary knot with Gauss code $G$ where $|G| = 2k$, let there be $i$ interior c-paths, $j$ exterior c-paths, $m$ interior crossings and $n$ exterior crossing. Then, the following equations and inequalities hold:*

$$m = i - (k-1) \tag{6.4}$$
$$n = j \tag{6.5}$$
$$m + n = k \tag{6.6}$$
$$i + j = 2k - 1 \tag{6.7}$$

$$j \geq 3 \tag{6.8}$$
$$k - 1 \leq i \leq 2k - 4 \tag{6.9}$$

*Proof:* First, there are total of $k$ crossings, so $m+n = k$. For $k$ crossings, where each crossings appears twice in the Gauss code, there are total of $2k-1$ c-paths between crossings, which lead to $i + j = 2k - 1$.

Each exterior c-path is on the boundary of 1 interior face. Each interior c-path is on the boundary of 2 interior faces. When we consider the crossings. For the first and the last crossing, they can only be adjacent to 2 interior faces. For the rest of the exterior crossings, each of them is adjacent to 3 interior faces (4 degrees cuts the plane into 4 pieces, one of which for the exterior vertex is the exterior face). For an interior

vertex, each vertex is adjacent to $4$ interior faces. Therefore, we have:

$$\begin{aligned} 2i + j &= 4m + 3(n-2) + 4 && (6.10)\\ \Rightarrow i + (2k-1) &= m + 3k - 2 && (6.11)\\ \Rightarrow i - (k-1) &= m, && (6.12) \end{aligned}$$

because for all the closed interior faces, the number of its boundary crossings equals the number of boundary c-paths.

We always have $m \geq 0$, so $i \geq k-1$. We can also have the following equations:

$$\begin{aligned} m + n &= k && (6.13)\\ \Rightarrow i - (k-1) + n &= k && (6.14)\\ \Rightarrow i + n &= 2k - 1 && (6.15)\\ \Rightarrow j &= n. && (6.16) \end{aligned}$$

For arbitrary knot, we know that the boundary of the planar drawing is closed, therefore all the exterior edges form a cycle, which involves three crossings at least.

Let us assume only two crossings $1$ and $2$ are exterior crossings, and crossing $1$ is both the first and the last label on the Gauss code. Then this crossing is formed by a type I Reidemeister move, so it can be removed, which lead to a different knot diagram and Gauss code. Then, the only possibility is if $1$ is the first label and $2$ is the last label on the Gauss code. It can be easily checked that such sequence will result in more crossings on the exterior c-paths. Therefore, $n = j \geq 3$. ■

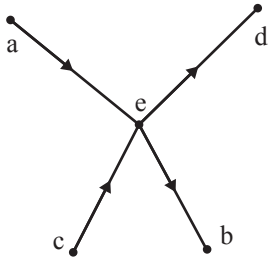
In order to come up with a sufficient number of line segments to draw arbitrary knot diagram given a Gauss code, we introduce an algorithm to place all the crossings and extra vertices, starting from exterior crossings and moving inwards. We will show that by connecting all the placed crossings (and extra vertices) using line segments, no intersection (except at common end points) is introduced. We will then show that this strategy will use a bounded number of line segments with respect to the number of crossings of the given knot.

In the following text, we will use the following notations for simplicity. We represent a crossing using a lower case letter. If the letter is used without any superscript, it represents both the over-crossing and the under-crossing. If the crossing is labeled with a superscript $^s$ (can be either $^+$ or $^-$), it represents the point on the polygonal curve that projects to the over-crossing (under-crossing). A negative sign may appear before a crossing notation with a superscript, then it means the same crossing with the opposite superscript. For example, an over-crossing can be represented as $a^+$, or $-a^-$.

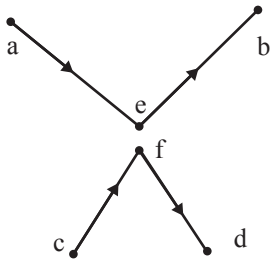
We start by showing how the edges are connected to an exterior crossing.

**Lemma 5** *Given a crossing with label $e$ that is not the first or the last crossing, if one c-path containing $e^s$ is an exterior c-path, then there must be another exterior c-path containing $-e^s$.*

*Proof:* We know that the on the knot diagram, there exist an Eulerian path starting from the first crossing to the last crossing. Each of the crossing is visited twice. A crossing $e$ that is not the first or last crossing has degree $4$, which means it is on a segment of polygonal curve that projects to $e^+$, and also on a segment of polygonal curve that projects to $e^-$. Therefore, on the polygonal knot diagram, $e$ has two




**(a)** Edges related to a crossing




**(b)** If the edges are not correctly structured, the vertex can be separated into two.

**Figure 6.1:** An example of the edges related to a crossing.

incoming edges and two outgoing edges. One pair of incoming and outgoing edge is associated with $e^+$, and the other pair is associated with $e^-$.

Without loss of generality, let us denote the incoming edges as $(a,\ e^s)$ and $(c,\ -e^s)$, and outgoing edges as $(e^s,\ b)$ and $(-e^s,\ d)$. Let we construct two rays starting from $e$ and passing through $c$ and $d$ respectively, denote them as $R_{ec}$ and $R_{ed}$. These two rays separate the plane into two disconnected regions, and $a$ and $b$ must belong to different regions, as shown in Figure 6.1a.

If $a$ and $b$ belong to the same region, then line segment sequence $(a, e, b)$ projects an over-crossing (or under-crossing) over (under) line segment $(c, e, d)$. Then, either these two line segment sequences form a type II Reidemeister move (as shown in Figure 6.1b) where $e$ can be redrawn as two independent vertices, or more than two points projects at $e$, which is a violation of the regular projection.

Without loss of generality, let us assume c-path $(a,\ e^s)$ is an exterior c-path. Then the c-path contacting the same exterior face will be another exterior c-path, which is either $(c,\ -e^s)$ or $(-e^s,\ d)$ since $a$ and $b$ belong to two different region separated by rays $R_{ec}$ and $R_{ed}$. ■

What about the first and last crossings?

**Lemma 6** *First crossing and last crossing are both exterior crossings. If we denote the first crossing as* $1^s$ *and last crossing as* $l^t$*, then the two c-paths with* $-1^s$ *as end points and two c-paths with* $-l^t$ *as end points are exterior c-paths.*

*Proof:* The polygonal arc starts from the first crossing, and ends at the last crossing, and the polygonal knot diagram is achieved by trimming the first and last link, making the first and last crossing the new end points of the polygonal arc, they must be adjacent to the exterior face, making them exterior crossings.

Let us consider the first crossing. Without loss of generality, let the first crossing be $1^s$. On the polygonal knot diagram, the first crossing has only degree 3, with only one outgoing edge associated with $1^s$, denoted as $(1^s, a)$, and two edges associated with $-1^s$, denoted as $(i, -1^s)$ and $(-1^s-, j)$ respectively.

If we draw two rays starting from $-1^s$ and pass through $i$ and $j$, then the two rays separate the plane into two regions with one of them containing edge $(1^s, a)$. Therefore, $(1^s, a)$ cannot be connected to the exterior face, so it cannot be on the boundary. Same argument can be made for the last crossing. ■

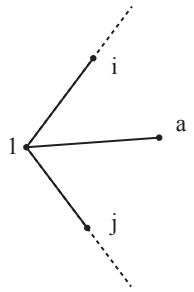



**Figure 6.2:** The line segments that pass through the first crossing.

Now we have ways to identify exterior c-paths and know two exterior crossings, we can identify all the exterior crossings using the following procedure. Without loss of generality, let the first crossing be $1^a$ and last crossing be $l^b$.

**Procedure 6.1:** Find exterior crossings and c-paths

1. Add two c-paths with $1^a$ as end points (denote as $(p,\ -1^a)$ and $(-1^a,\ q)$) into the exterior c-path list;
2. Add two c-paths with $l^b$ as end points (denote as $(s,\ -l^b)$ and $(-l^b,\ t)$) into the exterior c-path list;
3. Add crossings $s^c$, $t^d$, $p^e$ and $q^f$ into a queue $Q$, and add 1, $l$, $p$, $q$, $s$ and $t$ to exterior crossings list;
4. Check if any label appears twice ini the exterior c-path list with both superscript; if there is any, delete the corresponding labels from $Q$;
5. Pop a crossing $x^g$ from the queue, find previous crossing $u^h$ and next crossing $v^r$ of $-x^g$ on Gauss code, check if $u$ and $v$ are in $Q$ (might with different superscripts);
6. If $u$ (or $v$) is in $Q$, add $(u^h, -x^g)$ (or $(v^r, -x^g)$) to exterior c-path list, add $u$ (or $v$) to exterior crossing list (note, at most one of $u$ and $v$ can be in the queue);
7. If both $u$ and $v$ are not in $Q$, add both of them in $Q$, add both c-paths into exterior c-path list, and add both of them to exterior crossing list;
8. Check if a subset of exterior c-paths in the list form a cycle; if a cycle is formed, then delete all c-paths not used in the cycle, delete all crossings not used in cycle from $Q$ and exterior crossing list;
9. If $Q$ is not empty, go to step 5;

For example, let us consider a double coin knot, with Gauss code $G = \{1^+, 2^-, 3^+, 4^-, 5^+, 6^-, 2^+, 7^-, 4^+, 8^-, 6^+, 1^-, 7^+, 3^-, 8^+, 5^-\}$.

We first find $1^-$ and $5^+$ are related to the exterior c-paths, first add $(4^-, 5^+)$, $(5^+, 6^-)$, $(6^+, 1^-)$ and

$(1^-, 7^+)$ in the exterior c-path list, and put $4^-$, $6^+$, $6^-$ and $7^-$ in the queue. Since 6 has appeared twice, delete it from the queue. Let us start with crossing 4. Note that $4^+$ has previous crossing $7^-$ and next crossing $8^-$ on the Gauss code. We find that 7 is already in the queue, we add c-path $(7^-, 4^+)$ to the exterior c-path list, delete 7 from the queue, and notice the queue is already empty. Then, we terminate the process.

So far, we have c-paths $(4^-, 5^+)$, $(5^+, 6^-)$, $(6^+, 1^-)$, $(1^-, 7^+)$ and $(7^-, 4^+)$ in the exterior c-path list for double coin knot, and exterior crossings include 1, 4, 5, 6 and 7. Notice that the labels form a cycle.

**Procedure 6.2:** Place all exterior crossings

1. Place crossing 1 at $(-1, 0)$, denote the origin $o = (0, 0)$;
2. In the exterior c-paths list, we find an c-path $(1^a, b)$ that has crossing 1 as an end point, delete it from the list.
3. Place crossing $b$ at $(\cos(\pi - t), \sin(\pi - t))$ where $t = 2\pi/j$;
4. Label the crossing just labeled as $p$, find an exterior c-path $(p, q)$ in the list that has $p$ as an end point, delete it form the exterior c-path list; if $op$ has angle $\alpha$, then place $q$ at $(\cos(\alpha - t), \sin(\alpha - t))$;
5. If exterior c-path list has more than 1 c-path, go to step 4;

**Lemma 7** *If we place all the exterior crossings using Procedure 6.2, and connect two exterior crossings if they are adjacent on the Gauss code, there is no intersection between line segments other than the common end points.*

*Proof:* If we place all the exterior crossings based on Procedure 6.2 and connect them using line segments, all exterior c-paths form a regular convex polygon $P$. Because no chord of the regular convex polygon will intersect the boundary except at the common end points, no exterior c-path will intersect with non-exterior c-paths between exterior crossings.

We will then show that there is no intersection between c-paths (that are not exterior c-paths) between exterior crossings exists.

Let us then consider four exterior crossings $a, b, c, d$, such that $a$ and $c$ are adjacent on Gauss code, and $b$ and $d$ are adjacent on Gauss code. A line segment is used to connect $a$ and $c$. Line segment $ac$ separates the circle into two regions. If $b$ and $d$ belongs to the same region, then the connection between $ac$ and $bd$ will not create intersection.

If $b$ and $d$ belong to different region w.r.t. $ac$, then the only way to avoid intersection between $ac$ and $bd$ using arbitrary curve is if $b$ and $d$ is connected outside $P$, which means c-path $(b,\ d)$ is an exterior c-path. Since $(b,\ d)$ is not an exterior c-path, this case will not happen. ■

Now, we have placed all the exterior crossings and connected them if they are adjacent on the Gauss code. So far, no undesired intersection is introduced. An example of placing all exterior crossings and connecting them is shown in Figure 6.3.

Then, let us work inwards to place all the crossings that are directly connected to the exterior crossings.

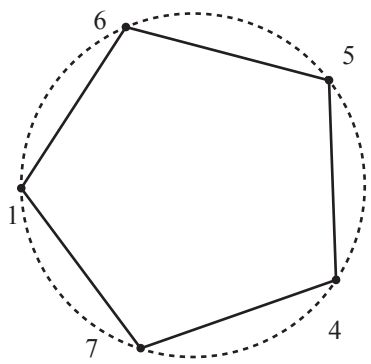



**Figure 6.3:** Place all exterior crossings for double coin knot.

**Procedure 6.3:** Place interior crossings

1. Find all crossings $V$ that directly connect to exterior crossings;
2. For a $v \in V$, find all its adjacent exterior crossings $E_v$, delete it from $V$;
3. Find the smallest circular sector that contains all the crossings in $E_v$, denote it as $aob$; let $oa$ and $ob$ intersect a circle centered at $o$ with radius $1/2$ at $a'$ and $b'$, place $v$ at the midpoint of $a'b'$ within $aob$;
4. If $V$ is not empty, go to step 2;

For the placement of a vertex $v$, consider the example shown in Figure 6.4. Given $v$ connects to $a$, $b$ and $c$ (or even $d$), we find the circular sector (wedge) with $oa$ and $ob$ as boundary containing all the crossings $v$ connects to. We can find the placement for $v$ on the smaller circle with radius $1/2$.

**Lemma 8** *If we place $v \in V$ based on Procedure 6.3, and connect all $v \in V$ to their corresponding exterior crossings using line segments, there exist an $\epsilon \geq 0$ perturbation of each $v$ along $ov$ away from $o$ such that these c-paths do not intersect with each other or with previously connected c-paths.*

*Proof:* Since all the c-paths that we introduce will be inside the regular convex polygon $P$ formed by exterior c-paths, they will not intersect exterior c-paths that are the boundary of $P$. We will first show that the connection between $v \in V$ will not intersect c-paths between exterior crossings that are not exterior c-paths.

Let us consider three exterior crossings $a, b, c$ and a crossing $v \in V$. Without loss of generality, let $a$ connect to $b$ and $v$ connect to $c$. The c-path $(a,\ b)$ separates $P$ into two regions. If $v$ and $c$ belong to the same region, their connection will not intersect $(a,\ b)$. Therefore, if for a given $v \in V$, all its adjacent exterior crossings are located in the same region w.r.t. $(a,\ b)$, as long as $v$ is placed in the same region, these c-paths will not intersect with c-paths between exterior crossings. Based on our strategy, $v$ is placed within

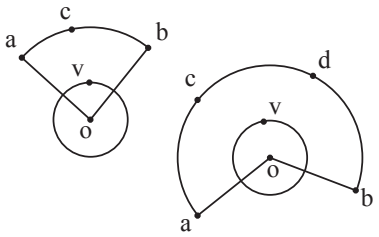



**Figure 6.4:** Finding placement for $v$ based on the exterior crossings it connects to.

the circular region of $aob$. However, as $v$ is placed on the circle with radius $1/2$, $ab$ might not intersect with this circle. In this case, we move $v$ along $ov$ to $v'$ to be within the same region as its adjacent exterior crossings, then the c-paths will not intersect with $ab$.

If the exterior crossings connected to $v$ belong to different regions w.r.t. $ab$, then there exist a c-path between $v$ and one of its adjacent exterior crossings that will intersect with $ab$, and cannot be avoided inside $P$. Since $v$ is not an exterior crossing, such situation cannot happen, because there exist a polygonal knot diagram corresponding to this Gauss code. Therefore, c-paths connecting $v \in V$ and exterior crossings does not intersect with any previously placed c-paths.

We will then show the c-paths between $v$ and corresponding exterior crossings will not intersect each other.

Let us consider two exterior crossings $a, b$ and two crossings $v_1, v_2 \in V$. Without loss of generality, let $a$ and $v_1$ be adjacent on Gauss code, and let $b$ and $v_2$ be adjacent on Gauss code. Let us connect them using straight line segments. If both $v_1$ and $v_2$ only connects to one exterior crossing, line segments $av_1$ and $bv_2$ cannot intersect. If $v_1$ also connects to another exterior crossing $c$, let us assume the connection is also a straight line segment. Curve $av_1c$ separates the regular convex polygon $P$ into two different regions. If $b$ and $v_2$ belong to the same region, then $bv_2$ should not intersect with $av_1$ and $v_1c$.

It is possible that for a crossing $v$, even though all the exterior crossing connected to $v$ belong to a connected region, a straight line connection may still create undesired intersection between interior-exterior crossings connections, since we place all crossings in $V$ on a circle. We may change the location of $v$ to $v'$ along $ov$, which will avoid any such type of intersection which is caused by occlusion. For example, consider Figure 6.5, when $v$ is placed on the circle, the connection between $v$ and exterior vertices $a$ and $b$ intersects with $pc$ connection, even though $p$ and $c$ belongs to the same connected region w.r.t. $v$, $a$ and $b$. By moving $v$ (or possibly $p$, not shown in this Figure) will avoid such collision. Therefore, from now on, we consider if two crossings belong to the same closed region, their connection will not intersect with any other interior-exterior connection between crossings.

If $c$ and $v_2$ belong to two different regions, then the intersection is unavoidable inside $P$. We will show that our placement scheme will not place two crossings in two different regions. Let us consider two crossings $v_1$ and $v_2$ that belong to $V$. First, consider a circular section $av_1b$ such that no exterior crossings inside $av_1b$ is connected to $v_1$. Then, if all the crossings connected to $v_2$ belongs to $av_1b$, then our placement will place $v_2$ inside $av_1b$ (maybe the previously mentioned perturbation is needed to finalize the location of $v_2$), so $v_2$ and all its neighbors will be located inside one region, so no intersections will be introduced.

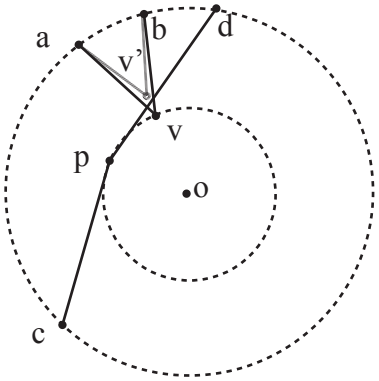



**Figure 6.5:** Example of intersection of line segment connection even end points belong to the same connected region.

Let us then consider a circular sector $av_1b$ such that all crossings connected to $v_1$ is on or inside $av_1b$. If all the crossings connected to $v_2$ are outside $av_1b$, then $v_2$ will be placed outside $av_1b$ based on our approach. Similarly, $v_2$ and its adjacent exterior crossings are in the same region w.r.t. $v_1$, so no intersection will be introduced.

Let us finally consider the circular sector $av_1b$ such that no crossing connected to $v_1$ is inside $av_1b$. The only case let is if for $v_2$, at least one of its adjacent exterior crossing is inside $av_1b$, and at least one of its adjacent exterior crossing is outside $av_1b$. In this case, our placement can place $v_2$ either inside or outside $av_1b$. However, no matter where $v_2$ is placed, it is in a different region with at least one of its adjacent exterior crossing. What is more, the exterior crossings cannot be reordered along the boundary (all the exterior c-path are defined on the Gauss code, so a change in order will introduce intersection), so there is no way $v_2$ and all its adjacent exterior crossing will be in the same region and a intersection will be introduced no matter what kind of curve is used to connect the crossings. Since the Gauss code did not record such intersection as a crossing, this case conflict with the fact that there exist a polygonal knot diagram correspond to this Gauss code. Therefore, this case will not happen. ■

An example of connecting all crossings in $V$ to exterior crossings for double coin knot is shown in Figure 6.6.

We then connect all the crossings inside $V$. We will show that these c-paths does not create intersections. First, they will not intersect with any previously placed c-paths, because these c-paths are in a smaller circle (or at least a convex shape), while all other c-paths are outside this circle. We will show that connections between crossings in $V$ will not introduce intersection.

If connection $v_1v_2$ intersects with $v_3v_4$, then $v_3$ and $v_4$ must belong to different regions, w.r.t. $v_1v_2$ (a curve pass through $v_1$ and $v_2$ and connect two exterior crossings $a$ and $b$). Since $v_3$ and $v_4$ are both connected to exterior crossings, then consider a curve pass through $v_3$ and $v_4$ that connects two exterior crossings $c$ and $d$. Two exterior crossings $c$ and $d$ belongs to two different regions w.r.t. the curve connecting $a$ and $b$ that pass through $v_1$ and $v_2$. Then, these two curves cannot avoid intersection inside $P$, which conflict with the fact that there exist a polygonal knot diagram. Therefore, no edges between crossings in $V$ will intersect.

For example, the final laid out polygonal knot diagram for double coin knot is shown in Figure 6.7. No extra intersection is introduced, and 15 line segments are used.

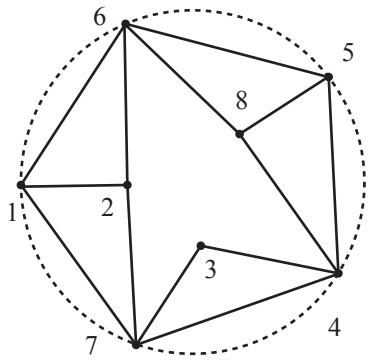



**Figure 6.6:** Connect crossings in $V$ with the exterior crossings based on the Gauss code.

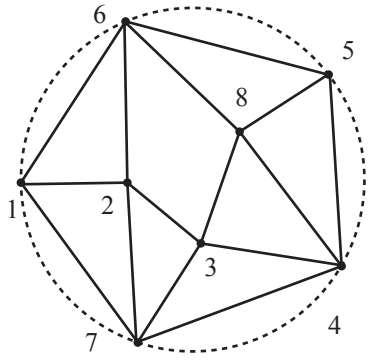



**Figure 6.7:** Connect crossings in $V$ to other crossings in $V$ for double coin knot.

If there are still crossings not placed, we recursively find crossings $V_1$ that directly connects to crossings in $V$, and place them on the circle with radius $1/4$ (and along with small perturbation). Similarly, the connection will not intersect with each other. We then recursively place all crossings, so no intersection will be created.

**Lemma 9** *For an arbitrary knot unit with Gauss code $G$, where $|G| = 2k < \infty$, there always exists a polygonal knot diagram of the knot with no more than $3k - 2$ line segments.*

*Proof:* We place all the crossings based on the Procedures 6.2 to 6.3, connect all of them using line segments, and no intersection in created. There are $k$ crossings, each visited twice, so $2k - 1$ line segments are used.

However, for two adjacent crossings $a$ and $b$ on Gauss code, the connection between them may appear twice such that the two line segments overlaps. For the second time the same connection appears, we will use one extra line segment passing through $p$ to avoid the overlapping. Let us denote the midpoint of $ab$ as $m_{a,b}$. For the second time $a$ connects to $b$, if it is an exterior c-path, we place $p$ at $(1 + \epsilon)o\vec{m}_{a,b}$, otherwise, we place $p$ at $(1 - \epsilon)o\vec{m}_{a,b}$, where $\epsilon > 0$.

Therefore, there are $k$ crossings, and the overlapping may appear at most $k - 1$ times. Therefore, we need at most $2k - 1 + k - 1 = 3k - 2$ line segments to draw any planar polygonal knot diagram. ■

In the proof, we choose to layout the exterior crossings on a circle. We choose a circle as a special and simple example of a convex shape. It turns out that if we arrange all the exterior crossings on a convex shape and intends to connect adjacent crossings (on Gauss code) using line segments, we cannot change the order of the crossings along the convex shape.

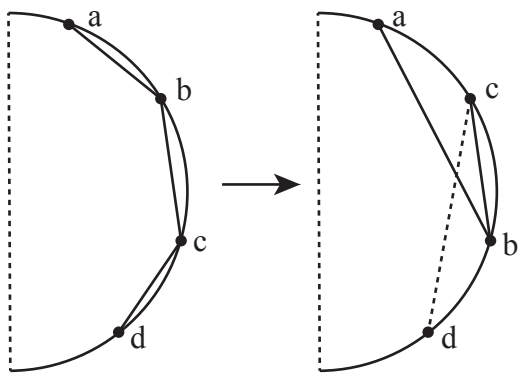



**Figure 6.8:** Perturb the order of the exterior crossings along the boundary will create undesired intersection.

If the order is changed, undesired intersection will be created. For example, consider Figure 6.8. Based on our arrangement, $a$, $b$, $c$ and $d$ should be in exact order on the circle clockwise, meaning $a$ and $b$ are adjacent on the Gauss code, $b$ (with a different sign) and $c$ are adjacent on the Gauss code, and so are $c$ (also with a different sign compared to previous $c$) and $d$. If $b$ and $c$ change their order along the circle, then connection $ab$ and $cd$ will intersect and also violates the assumption that they are exterior c-paths. It is easy

to verify that change order of non-adjacent crossings on the circle will also not work. Therefore, the order of exterior crossings cannot change along the boundary.

Obviously, the exterior crossings can be laid out on a concave boundary. However, in this case, the connections between interior crossings and exterior crossings have a higher chance of creating undesired crossings.

Combine Lemma 3 and Lemma 9, we have

**Theorem 4** *For an arbitrary knot with Gauss code $G$ where $|G| = 2k$, to draw a polygonal knot diagram of this knot, at least $\left\lceil \frac{3+\sqrt{8k+1}}{2} \right\rceil$ and no more than $3k-2$ sequential line segments are required.*

*Proof:* It follows directly from Lemma 3 and Lemma 9. ■

Based on the algorithm presented earlier, there do exist knots that require $3k-2$ line segments to represent their polygonal knot diagram.

Now we know how to draw the polygonal knot projection, we can project it back into 3D to get the number of line segments needed to represent a polygonal knot.

The necessary number of line segments in 2D will still be necessary. However, due to the lack of knowledge about the crossing sequence, we are not able to find a larger necessary number of line segments required in 3D to span a polygonal knot.

Then, how many sequential line segments are required to project to the polygonal knot diagram presented in the proof of Lemma 9? It turns out, it is still $3k-2$.

**Theorem 5** *For arbitrary polygonal knot with $k$ crossings, at least $\left\lceil \frac{3+\sqrt{8k+1}}{2} \right\rceil$ and no more than $3k-2$ sequential line segments are required to represent the polygonal knot.*

*Proof:* In order to project to $n$ line segments on the plane, at least $n$ line segments are needed in 3D to form such a projection. As for the polygonal knot diagram constructed in the proof of Lemma 9, since all the crossing are happening at the vertices of the projected planar grah, we can actually use the exact same number of line segments, though the number of vertices on the polygonal arc may close to double.

The idea is, given the crossing location $(x_i, y_i)$, if it is an over-crossing, we place the crossing at $(x_i, y_i, z_i^+)$, where $z_i^+ > 0$, and for the same crossing number that is the under-crossing, we place it at $(x_i, y_i, z_i^-)$ where $z_i^- < 0$. For simplicity, let us assume for all $i$, $z_i^+$ are equal, and all $z_i^-$ are equal. If the vertex is an added vertex to avoid overlapping in the projected graph, we put it at $(x_i, y_i, 0)$. Since the projected line segments are not intersecting, the original 3D line segments are not intersecting providing that the $z$ locations of the crossings are different for over- and under-crossings. Therefore, we can use no more than $3k-2$ line segments to describe arbitrary polygonal knot with $k$ crossings. ■

## 6.3 Knot folding

Now that we have the number of line segments needed to grasp any knot, we can start to think about how to fold it using the polygonal arc. If a knot can be represented with $n$ line segments, it may be folded with no less than $n$ line segments. However, how much more line segments are required is the question we would like to answer. Is there a bound to the additional line segments needed?

Let us assume in the context of knot folding, a polygonal knot $K_p$ is approximated by a polygonal arc $P$ if the projection of $P$ on $x$-$y$ plane is a polygonal knot diagram of $K_p$. A link $l$ on a polygonal arc is approximated by a link $l'$ if the projection of $l'$ overlaps with the projection of $l$ on $x$-$y$ plane.

**Theorem 6** *For an arbitrary knot with Gauss code $G$ where $|G| = 2k$, it can always be folded by a sequence of no more than $9k - 6$ line segments.*

*Proof:*

Let us assume we will use a polygonal arc $P$ to approximate the configuration of the polygonal arc $K_p$ constructed in Theorem 5. Let us denote the two end points on the $K_p$ as $v_0$ and $v_n$.

Let us assume that for two vertices $v_i$ and $v_j$ on $K_p$, if $i < j$, then $v_i$ appears earlier than $v_j$ when traversing along the $K_p$ from the end point $v_0$. In addition, let us assume that once we have approximated a link $v_iv_{i+1}$ on $K_p$, we fix the link(s) on $P$ used to approximate $v_iv_{i+1}$. In addition, we require that when approximating a link $v_iv_{i+1}$ using links from vertex $p_a$ to $p_b$ on $P$, where $a < b$, $p_a$ and $v_i$ have the same coordinates, and $p_b$ and $v_{i+1}$ have the same coordinates.

Let us start with $n$ links on $P$. For the $i$th link ($l_i$) between vertex $p_{i-1}$ and $p_i$ where $1 \leq i \leq n$, the length of the link $d(p_{i-1}, p_i)$ ($d(l_i)$) equals the distance between $v_{i-1}$ and $v_i$, the $i-1$th and $i$th vertex on $K_p$, $d(v_{i-1}, v_i)$.

Let us start with the first link, $v_0v_1$. Without loss of generality, let $v_1$ be an over-crossing. We use the link $p_0p_1$ to approximate the first link, while keeping all the links from $p_1$ to $p_m$ ($m \geq n$) above the plane $z = z(v_1)$. Then we approximate $v_1v_2$ using link $p_1p_2$. During approximation, we move all the links to be below the plane $z = z(v_2)$ before moving $p_2$ to the location of $v_2$, by rotating link $j$ (starting from the last link) around vertex $p_{j-1}$ ($j \geq 1$) on plane $\mathbb{P}_{j-1}$ that is perpendicular to $x$-$y$ plane, and pass through $v_{i-1}$ and $v_i$. We repeat this process, if we approximate an over-crossing (under-crossing) using the links up to $p_i$, then move all the remaining links to be above (below) the plane of $z = z(p_i)$.

When approximating a vertex $v_i$ (where we used up to $p_j$ to approximate $v_{i-1}$, without loss of generality, let $v_i$ be an under-crossing and $v_{i-1}$ be an under-crossing), since all the previous links are fixed in space, it is possible that their projection forms a loop on $x$-$y$ plane. Let $v'_{i-1}$ and $v'_i$ denote the projected points on $x$-$y$ plane, and the over-crossing of $v_i$ (denote as $v_i^o$) has been placed, then we know $d(p_j, p_{j+1}) > d(v'_{i-1}, v'_i)$.

In this situation, before link $l_{j+2}$ and after link $l_{j+1}$ (remember, links $l_1$ to $l_i$ are fixed already, and we were supposed to move $l_{i+1}$ to let $p_{j+1}$ occupy the same location as $v_i$), we add two links $l^1_{j+1}$ and $l^2_{j+1}$, and change the length of $l_{i+1}$. Let $d(l^1_{j+1}) = d(l_{j+1}) \geq \max(l^a_b)$ where $b \geq j+2$ and $a = \{0, 1, 2\}$ (with an abuse of notations, let $l^0_i$ be the same link as $l_i$), and let $l^2_{j+1} = \sqrt{d(v_i, v_i^o)^2 + \epsilon^2}$. This way, for arbitrary small $d(v'_{i-1}, v'_i)$, all the links $l_i$ (except the last link) can be folded such that $p_m$ for $m \geq i+1$ is along the line that is perpendicular to $x$-$y$ plane and pass through $v'_{i-1}$, by folding two or more links before $p_m$ above the plane $z = z(v_{i-1})$ to form a triangle with part of $l_{m+1}$. After the folding, link $p_{m-1}$ is also perpendicular to $x$-$y$ plane, and directly above $v'_{i-1}$, and $p_{m-1}$ is above $p_m$.

An example of the folding is shown in Figure 6.9, as long as we maintain the length $l(AB) \geq l(BC)$. If the link before $p_{m-1}$ does not satisfy the situation, we let $AB$ be the set of links that is first longer than $BC$. Since we have set $d(l^1_{j+1}) = d(l_{j+1}) \geq \max(l^a_b)$, the situation will always be satisfied near $p_j$.

After all the links are folded (including $l^1_{j+1}$, denote the end points as $p^1_j$ and $p^1_{j+1}$), we move $p^1_{j+1}$ on the plane $z = z(v_{i-1})$ along $\vec{v'_{i-1}v'_i}$ and stop when $d(p^1_{j+1}, v_i^o) = \epsilon > 0$. We then move link $l^2_{j+1}$ to let $p^2_{j+1}$ occupy the location of $v_i$. Repeat this process, we can approximate $K_p$ using $P$.

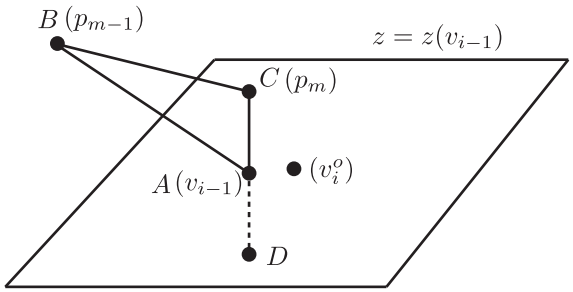



**Figure 6.9:** Folding links given arbitrary small clearance.

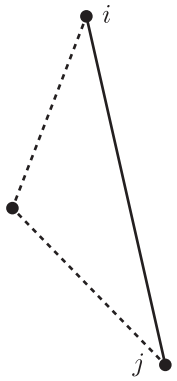



**Figure 6.10:** The method to avoid overlapping in Lemma 9.

Since there are at most $3k-2$ links on original layout, we may at most need to add two links every time we approximate a new link. Therefore, we need at most $9k-6$ links. ■

We are able to find a sufficient number of links to approximate arbitrary polygonal knot with $k$ crossings. However, even though the number of links only grows linearly with the number of crossings, we may still not be satisfied with the large number of links needed to fold such knots.

**Theorem 7** *By re-arranging the locations of the crossings computed in Lemma 9, the number of line segments needed to represent the polygonal knot diagram can be reduced to* $2k-1$*, and the number of line segments needed to fold the knot is reduced to* $6k-3$*.*

*Proof:* Compare to $2k-1$, the extra $k-1$ line segments comes from the overlap line segments (repeated adjacent crossings on the Gauss code). We originally added one vertex near the center of the connection, to detour the second connection to avoid overlapping, as shown in Figure 6.10.

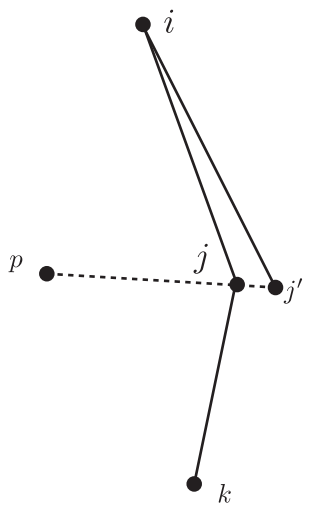



**Figure 6.11:** The method to avoid overlapping in the first case.

Let us consider two crossings $i$ and $j$ that appeared twice in the Gauss code. Let us first consider the case where $ij$ is adjacent to $k$ the first time, and adjacent to $p$ the second time on the Gauss code, and $k$ and $p$ are different crossings. Let us add a vertex $j'$ on the extension of $pj$, where $d(j, j') < \epsilon$ where $\epsilon > 0$ is a small number. When the second time $i$ is adjacent to $j$ on the Gauss code, let $i$ connects to $j'$ instead of $i$, then the crossing $j$ still happens at $j$, but no collision will happen, as shown in Figure 6.11.

Now let us consider the case where $i$ and $j$ are connected to $k$ both time on the Gauss code. Let there be $j'$ and $j''$, where $j'$ is on the extension of $ij$, and $j''$ is on the extension of $kj$. Let $d(j, j') < \epsilon$ and $d(j, j'') < \epsilon$ When the first time $ij$ appears adjacent on the Gauss code, let $i$ connects to $j'$ instead of $j$, and let $i$ connects to $j''$ when the second time they appear adjacent on the Gauss code, as shown in Figure 6.12. The crossing $j$ is still happening at the location of $j$, but $j$ is a projected intersection instead of an end point of a line segment.

By re-arranging the crossings following the principles stated earlier, only $2k - 1$ line segments are needed (because we avoided using $k - 1$ extra line segments to avoid collision).

When projecting back to 3D, in the first case, $j$ and $j'$ have different $z$ coordinates, so the two line segments will not intersect. In the second case, because $j'$ and $j''$ must have different $z$ coordinates, the two connecting line segments will not intersect. Therefore, $2k - 1$ line segments are needed to represent the polygonal knot. And with the same argument in Theorem **??**, at most two line segments are needed to add at each crossing, so $6k - 3$ line segments are sufficient to fold the knot. ■

### 6.3.1 Loose complexity bound

Although the preceding theorem gives a sufficient number of links, analysis of particular knots may allow tighter bounds. For example, to fold the configuration calculated using the algorithms from the previous section, 8 fingers are sufficient to fold an overhand knot, while Theorem 5 requires 8 fingers to immobilize

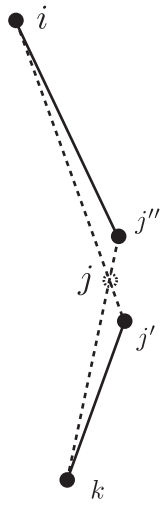



**Figure 6.12:** The method to avoid overlapping in the second case.

it; and 19 fingers are sufficient to fold the double coin knot, and Theorem 5 requires 16 fingers to immobilize it.

**Theorem 8** *At least 5 links are necessary to fold an overhand polygonal knot.*

*Proof:* For any polygonal arc to project to a diagram with 1 crossings, three links are needed. Following Lemma 3, four links are required to achieve 3 intersections, which is the crossing number of an overhand knot.

For the first three links $v_0v_1$, $v_1v_2$ and $v_2v_3$, let assume that $z(v_0) = z(v_1) = z(v_2) = 0$, and $z(v_3) < 0$. In order to achieve the three crossings with four links, the projection of $v_0v_1$ must intersect with the projection of $v_2v_3$, let us assume the projected intersection is $I_1$.

At $I_1$, since $v_0v_1$ is above $v_2v_3$, this crossing is an over-crossing, correspond to the $1^+$ on the Gauss code. So far so good. In order to create additional two crossings with link $v_3v_4$, the projection of $v_3v_4$ need to intersect $v_0v_1$ and $v_1v_2$. Let us denote the intersection between the projection of $v_3v_4$ with $v_0v_1$ as $I_2$, and the intersection between the projection of $v_3v_4$ and $v_1v_2$ as $I_3$.

If $z(v_4) < 0$, then at $I_2$, $v_0v_1$ is above $v_3v_4$, then the second crossing when walking from $v_0$ along the links is an over-crossing again, which is conflict with the $2^-$ on the Gauss code. Therefore, $z(v_4) > 0$, and at $I_2$, $v_3v_4$ must be above $v_0v_1$. Since $v_3v_4$ is a straight line segment, let us denote $p$ as the point projected to $I_3$ on $v_3v_4$. It can be verified that the line segment from $p$ to $v_4$ is all above the $z = 0$ plane. Therefore, at $I_3$, $v_3v_4$ is above $v_1v_2$. Then, the reader can check that after satisfying the second crossing to be an under-crossing on $v_0v_1$, the third crossing on $v_1v_2$ is also an under-crossing. Then, the resulting Gauss code will be $1^+2^-3^-1^-2^+3^+$.

The previous polygonal arc does not form an overhand knot, and this is the only possible arrangement for the polygonal arc. Therefore, one additional link is needed to represent the overhand knot, so at least 5 links are necessary to fold an overhand knot.

The reader can check in order to fold an overhand knot, $v_4$ needs to project inside the triangle $I_1v_1v_2$, where $z(v_4) > 0$. Then the link from $v_4$ to $v_5$ must projects to intersect with either $v_1v_2$ or $v_2v_3$, with $z(v_5) < 0$. ■

**Theorem 9** *There exist* $5$ *links that cannot fold into an overhand polygonal knot.*

*Proof:* Let us consider a $5$ link that has the following length in some given unit: $(100, 1, 1, 1, 100)$. t is easy to verify that no matter how to move the links, these five links are not able to fold into an overhand knot.

In order to form an overhand knot, a loop needs to be formed, and one end of the link needs to go through the link. However, with the given link, the loop has maximum diameter $1$, and either end contains a link of length $100$. No motion permits the link with length $100$ to fold through the loop with diameter at most $1$. Therefore, this particular $5$-link cannot fold an overhand knot, even though it is possible to represent an overhand knot with these five links. ■

## 6.4 Conclusions

In this chapter, we give the first bound on how many contacts are need to arrange an arbitrary knot. A generic polygonal knot configuration is first computed, and then extra links are added to fold the knot into this polygonal knot configuration sequentially. However, the bound we computed are quite loose, leading to a large number of contacts needed to arrange an arbitrary knot. An important part of future work is to reduce the number of contacts needed to arrange a given knot.

# Chapter 7

# Knot tying with few re-grasps

In the previous chapter, we derived the first bound on the number of contacts needed to arrange an aribtrary knot. The loose bound yields large number of contacts to arrange any knot. The strategy then is not very suitable in practice as the state of arm robot arms and robot hands cannot provide the number of contacts needed.

With human arms, or traditional robot arms that are used in practice, some re-grasping is unavoidable during tasks such as knot tying, due to the topology of the arms and the target topology of the string. In this chapter, we investigate the use of re-grasps in a knot tying task, and attempt to reduce the use of re-grasps, as re-grasp traditionally requires precise configuration information that is hard to acquire especially for flexible objects. And mis-grasps can lead to total failure.

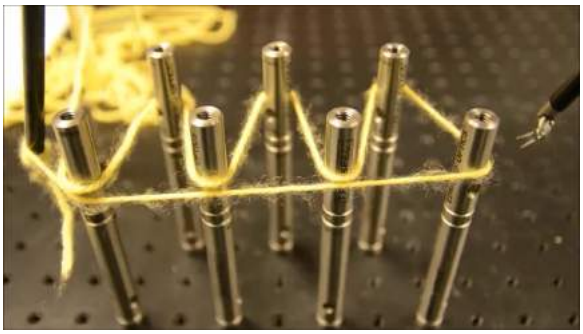
Inspired by the mechanical looms and weaving, we will show that by arranging knots like a loom forms weaving, only a small number of re-grasps are needed to arrange many knots.

Through theoretical analysis on the *Gauss code*, a common text based description of knots commonly used in mathematical knot theory, we derive an algorithm to compute a small sufficient number of re-grasps for tying a given knot. The use of a small number of re-grasps lead to a novel way to tie many different knots both for robots and for humans. Some knots, including the *double coin knot* which has $8$ crossings, can be tied with a single re-grasp.

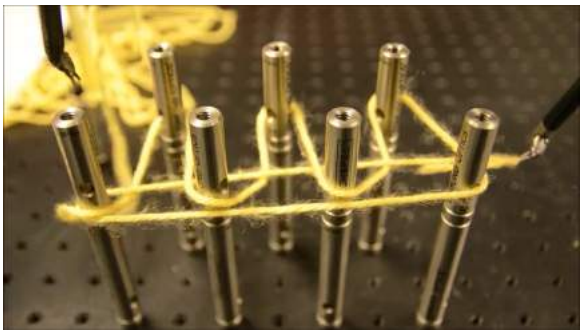
## 7.1 Introduction

We can divide the crossings on the Gauss code into two different categories. The formation of the crossings in the first category do not change string structure topologically, even when the ends of the string are grasped. Figures 7.1a and 7.3a shows the arrangement of crossings in the first category for a decorative double coin knot, and the knot labelled as $7_1$ on the standard knot table. The formation of the crossings of the second category changes the topological structure of the string. Theoretically, the crossings in the first category can be formed in parallel, while the crossings in the second category can only be formed sequentially.

Our approach is to first lay out the type-1 crossings amongst a set of vertical rods, using a single motion without re-grasping. Then we use a simple weaving motion, either along a straight line, or with only vertical displacement to allow an alternating over-under pattern, to complete the type-2 crossings. Figures 7.1b and 7.3b show examples.



**(a)** Arranging the type-1 crossings around a set of straight rods.



**(b)** Completing the arrangement of a knot $7_1$ with one re-grasp.

**Figure 7.1:** Arranging a knot $7_1$ with Da Vinci robot arm.

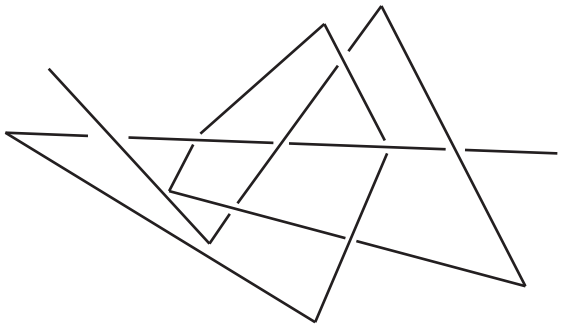


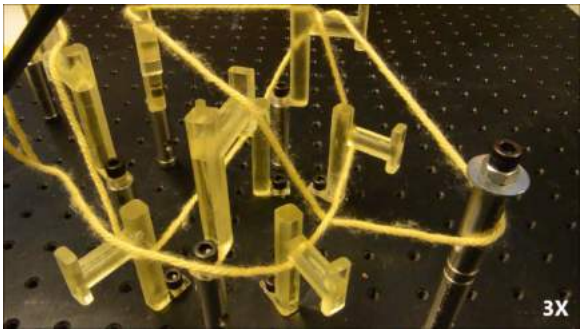
**Figure 7.2:** The resulting configuration of a double coin knot so that the last five crossings are on a straight line.

We conducted experiments with a Da Vinci surgical robot to show autonomous knot tying with a small number of re-grasps. In a second set of experiments, we used an Adept Cobra industrial arm to arrange the type-1 crossings, while a human arranged the type-2 crossings.

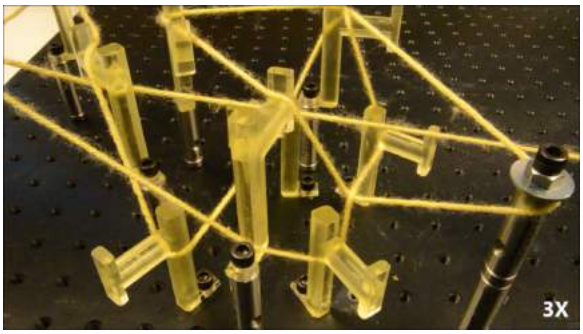
## 7.2 Knots and weaving

In a weaving loom, the warp is the set of strings that form the basic structure around which the weft (the string pulled by the shuttle) is woven. We can think of the first arrangement of string using only type-1 crossings as forming the *warp* for knot tying, and the weaving of the type-2 crossings is analogous to the motion of the weft.

For example, on a double coin knot with Gauss code $G = \{1^+, 2^-, 3^+, 4^-, 5^+, 6^-, 2^+, 7^-, 4^+, 8^-, 6^+, 1^-, 7^+, 3^-, 8^+, 5^-\}$, the last five crossings counting from the right open end are $5$, $8$, $3$, $7$, and $1$. These five crossings can be formed by dragging the right open end along a straight line simulating the motion of a shuttle on a loom (*weaving motion*), provided that the other segments of string are arranged appropriately.




**(a)** Arranging the type-1 crossings of a double coin knot.




**(b)** Completing the arrangement of a double coin knot with one re-grasp.

**Figure 7.3:** Arranging a double coin with Da Vinci robot arm.

An example of the rearranged polygonal configuration of a double coin knot is shown in Figure 7.2. We implemented the knot arrangement using the proposed layout with a Da Vinci robot arm. Figure 7.3 shows the results of implementation.

## 7.3 Decoding the Gauss code

Which parts of knot tying tying can be accomplished without re-grasps? This section shows that by deleting type-2 crossings from the Gauss code, simple structures containing only type-1 crossings can be discovered. Remember that each crossing appears twice on a Gauss code, one with a $^+$ superscript (over-crossing), and one with a $^-$ superscript (under-crossing).

### 7.3.1 Tying knots: forming or removing crossings

We define a *minimal Gauss code* as a Gauss code that does not contain a bridge, and cannot be simplified by performing Reidemeister moves. The Gauss code for an unknot or a compound knot is not a minimum Gauss code. Adding or removing of a crossing from a structure with minimal Gauss code through physical manipulation of the string can only be achieved if one of the two appearance of the crossing number is at the beginning or the end of the Gauss code. What happens if we remove crossings one by one from the open ends? Intuitively, we know that after a certain number of removals, the remaining crossing pattern becomes an unknot, because eventually the knot is untied if we remove all crossings. The knotting and unknotting process are symmetric, so for simplicity, we choose the unknotting process for analysis. It is easier to see the pattern when you remove crossings from a existing sequence of crossings.

### 7.3.2 Counting the number of weaving motions

Consider the example of unknotting a double coin knot, whose Gauss code is $G = \{1^+, 2^-, 3^+, 4^-, 5^+, 6^-, 2^+, 7^-, 4^+, 8^-, 6^+, 1^-, 7^+, 3^-, 8^+, 5^-\}$. Let us start from the right end. Crossing $5^-$ is adjacent to $8^+$

on the Gauss code, and the crossings have *different* superscript signs. Therefore, we can identify these two crossings as part of a *weaving pattern*, and delete crossing $5$ from the code. Here, we define a weaving pattern as a sequence of alternating over- and under-crossings that has to be formed or deleted in the given order indicated by the Gauss code.

We continue to remove crossings that are part of a weaving pattern from the right end, including crossings $8$, $3$, $7$, and $1$, in order. After we remove the last five crossings on the Gauss code, we have $G = \{2^-, 4^-, 6^-, 2^+, 4^+, 6^+\}$. Now, the next two crossings from the right have the same superscripts, so they are no longer part of a weaving pattern. We know that the five deleted crossings can be formed by a single weaving motion; we continue searching for weaving patterns from right to left. In this example, there are none, and the remaining structure consists only of type-1 crossings.

The algorithm continues to remove crossings. No new weaving motion is registered until the current crossing to remove has different sign as its adjacent crossing. The new weaving motion terminates again when the current crossing to remove has the same sign as its adjacent crossing. Repeat the process until only one crossing is left. This last crossing will be appended to the ongoing pattern. The algorithm outputs $m$ sequence of crossings that can be formed by weaving motions.

For a Gauss code with $k$ crossings where $|G| = 2k$, we can find $O(k^2)$ different sub-strings of Gauss code that are the results of removing the crossings at the beginning or the end of the Gauss code. However, since a weaving motion is achieved by weaving with one end of the string, we only need to check the sub-strings that remove crossings from solely the left or right end. The total number of such sub-strings is $2k$.

The proposed algorithm only checks if the current crossing has a different sign as the adjacent one. This approach may overlook some structures that are unknotted but still contain adjacent crossings that have different signs, such as $\{1^+, 2^-, 3^+, 4^-, 4^+, 3^-, 2^+, 1^-\}$. This structure is unknotted, but can be formed by one weaving motion.

It is possible that the algorithm finds that the current crossing has the same label as the next crossing (but different sign, $i^+$ and $i^-$). In this case, we compare the sign of the current crossing to the sign of the first crossing that has a different label than the current.

Knots such as the overhand knot can be arranged by only one weaving motion to form the last two crossings, while the first crossing can be formed by a type I Reidemeister move. Similarly, figure eight knot can be arranged with one weaving motion to achieve the last three crossings where the first crossing is achieved by a type I Reidemeister move. A double coin knot as shown earlier, can be arranged with one weaving motion, with the crossings $2$, $4$ and $6$ form an unknotted structure.

**Lemma 10** *If a knot can be arranged with only one weaving motion, then the crossings formed before the weaving motion is unknotted.*

*Proof:* Based on the definition, the weaving motion only happens once. Therefore, the rest of the structure may contain a sequence of crossings with monotone superscripts; or a pattern like $\{i^-, i^+, j^-, k^-, j^+, k^+\}$, which is also a monotone superscript pattern, in the sense that when we remove the crossings from the right end, every crossing will be an over-crossing near the open end. The string projecting these crossings can be arranged on two different planes (one plane contains only the over-crossings, while the other plane contains all the under-crossings) with finitely many vertical line segments connecting two planes, where the

vertical line segments do not project inside any cell. This structure has the topology of a circle when the ends of the string are connected to each other; an *unknot*. ■

If we refer the motion that do not change the topology of the string as a *folding motion*, we can have the following lemma.

**Lemma 11** *All knots can be tied with alternating folding motions and weaving motions.*

*Proof:* A knot contains either sequences of alternating over- and under-crossing patterns, or consecutive over- or under crossings. The former pattern can be achieved by weaving motion, and the later pattern can be achieved by folding motion. No other crossing pattern presents, therefore all knots can be tied with folding and weaving motions. The motions must alternate, as two consecutive weaving motion form a weaving motion, and two consecutive folding motions can be executed in parallel as a single folding motion. ■

## 7.4 Knot weaving

A straight line motion is easy to achieve even for simple robotic devices. This section will show that weaving crossings can always be aligned on a single straight line.

### 7.4.1 Aligning crossings on a straight line for weaving

**Theorem 10** *A sequence of crossings that can be formed by a weaving motion are connected by a continuous curve in the order of their formation, and this curve does not project self-intersections.*

*Proof:* Since a weaving motion forms an alternating over- and under-crossing pattern, tracing along one end of the string after the weaving motion, all the crossings on the weaving pattern are sequential. Therefore, there exist a curve on which are all the crossings formed by a weaving motion.

Let the curve connecting the crossings project a self-intersection, at crossing $j$. Let crossings $i^-$ and $k^-$ be the two crossings on the Gauss code adjacent to $j^+$, and let $s^+$ and $t^+$ be the two crossings adjacent to $j^-$. The crossings $i$, $k$, $s$ and $t$ have the corresponding signs because they are adjacent crossings to $j$, and they are formed by a weaving motion; by definition they have different signs from their adjacent crossings. Without loss of generality, let $j^-$ be closer to the open end. After the deletion of the crossing $j^-$, crossings $i$ and $k$ are now adjacent on the Gauss code, and they have the same sign, so they cannot be on the weaving motion. Therefore, the curve connecting the crossings formed by weaving motions in order cannot project self-intersection. ■

**Lemma 12** *Starting from an arbitrary knot diagram configuration, there exists a reconfiguration of the string that can align all crossings of a weaving motion onto a straight line.*

*Proof:* Let us define an *extremal segment* of the weaving pattern as the segment between two exterior crossings. We will show that there is a simple way to arrange the crossings on each external segment onto a straight line. Then we will show that there exists a method to align two adjacent extremal segments on the same line.

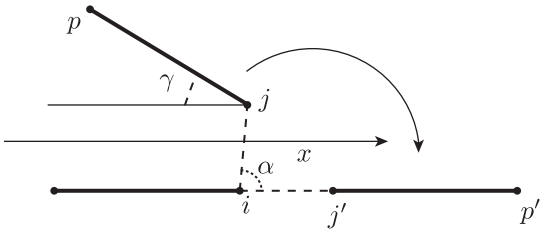
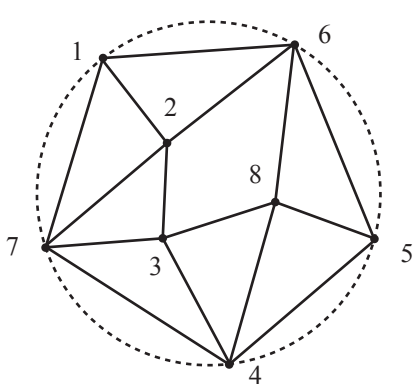
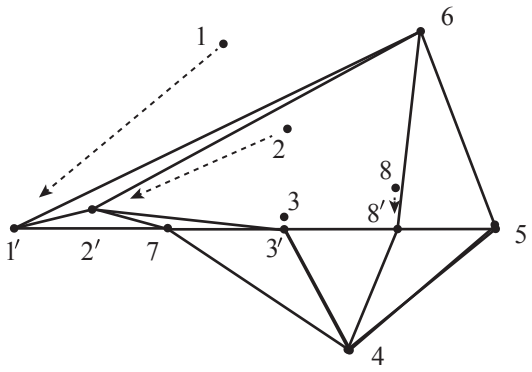



**Figure 7.4:** Rotating extremal segments to align all weaving crossings on a straight line.




**(a)** A polygonal knot diagram for a double coin knot, computed using methods proposed in [WB16a].




**(b)** The rearranged configuration for the double coin knot, by rotating extremal segments using the proposed method indicated in the proof of Lemma 12.

**Figure 7.5:** Reconfiguration of a double coin knot to align type-2 crossings onto a straight line.

For each extremal segment, we can connect a line between two exterior crossings (or an interior to an exterior crossing), and move all crossings on the extremal segment to their projections on the line. No new intersections will be created, because every projected intersection is a crossing. Since we are adjusting the extremal segment, every crossing on it must be moved. We can do this for all the extremal segments following the order along the weaving sequence.

Between two adjusted extremal segments, without loss of generality, let crossings $i$ with coordinates $(x_i, y_i)$ and crossing $j$ with coordinates $(x_j, y_j)$ be the adjacent two exterior crossings on the weaving sequence. Let there be $k$ connections between two extremal segments with $i$ and $j$ as endpoints respectively. Also, without loss of generality, let $y_j > y_i$, and on the extremal segment with endpoint crossing $j$, denote the other endpoint as $p$. We rotate all the points above extremal segments between $p$ and $j$ around crossing $j$, then around crossing $i$ in the stated order, so that for the new location $(x'_j, y'_j)$ of crossing $j$, $y'_j = y_i$, as shown in Figure 7.4. The angle rotated around crossing $i$ can be calculated as the acute angle $\alpha$ between the $x$ axis and the vector $\vec{ij}$, and the rotation angle around crossing $j$ is $\beta = \pi - \alpha - \gamma$, where $\gamma$ is the angle between $\vec{pj}$ and $x$ axis. Along the line $x = (x_i + x'_i)/2$, find $k$ points above (or under) the line $y = y_i$ if $x'_i \geq x_i$ ($x'_i < x_i$) with equal distance. For the end points of the $k$ pair of connection between two extremal segments, connect to $k$ points along the line $x = (x_i + x'_i)/2$ in order based on the distance of the end point on segment $pj$ to the exterior crossing $j$, such that no additional intersection is introduced. Finally, we adjust the $z$ coordinates of all crossings so that the crossings on the rearranged weaving pattern lie on a straight line in three dimensions. The result of applying the process to a double coin knot is shown in Figure 7.5. ■

We have shown in previous work [WB16a] how to construct a polygonal knot diagram directly from the Gauss code; the method implied by the previous proof can then be applied to this knot diagram.

### 7.4.2 Re-grasping and weaving

This section analyzes the number of re-grasps needed to arrange each of the two types of crossings.

**Lemma 13** *Let $i$ be a crossing label. If on a sequence of crossings, $i^-$ always appears before $i^+$, then the crossings can be laid out by a robot arm without re-grasping.*

*Proof:* The structure is unknotted, and belongs to two layers. The arm alternates laying out string in the bottom layer, and creating crossings by completing layout in the top layer, since for each crossing, the under crossing appears before the over crossing. ■

**Lemma 14** *There exist sequences of crossings of the form $i^a, j^b, k^a$, where $a$ and $b$ have opposite signs, such that to arrange such crossings with a robot arm, re-grasping is required.*

*Proof:* Without loss of generality, let the robot arm end effector be located below the arm in the $z$ direction, and let $a = +$ and $b = -$. Let crossings $i$, $j$, and $k$ be the second time these crossings appear on the Gauss code, so that there are segments of string under crossing $i$ and $k$, and segments of string over crossing $j$. Then, at crossings $i^a$, the robot effector must be grasping the end of the string on top of another segment of string. In the next step, the robot arm must arrange crossings $j^b$ and $k^a$. Let point $A$ be the highest point on the robot arm, and $E$ be the location of the effector. The closed shape $Ai^ak^a$ intersects with a segment of string $(s(j))$ on which $j^a$ lies. The crossing $j^b$ is on the connection $i^ak^a$. Since the string is

**(a)** Arranging type-1 crossings of a double coin knot at the same height.

**(b)** Completing the arrangement of a double coin knot with a single re-grasp.

**Figure 7.6:** Arranging a double coin by laying out the type-1 crossings on the same height.

continuous, if one open end is attached to $E$, the sweeping of $AE$ from $i^a$ to $k^a$ must intersect with $s(j)$. The robot end effector needs to release before $j^b$ and re-grasp after $j^b$ to arrange this structure. ■

**Lemma 15** *To perform a weaving motion with a robot arm grasping the ends of the string, at least one re-grasp is needed to change the topology of the formed structure.*

*Proof:* Let us consider at the end of the weaving motion, before re-grasping and after re-grasping. If we keep the endpoint of the string the same, then the two configurations of the robot arm belong to two different homotopy class with respect to the string (inside vs. outside the convex hull of the string geometry), and the configuration of the entire system after the re-grasping is the one containing the correct knot topology. ■

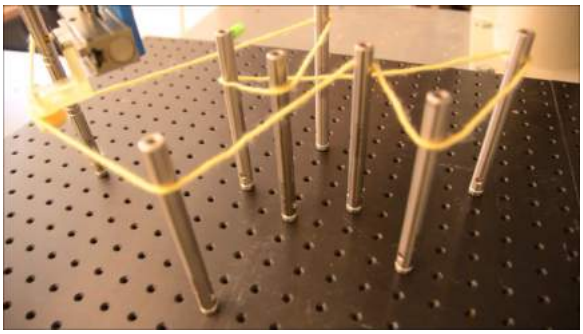
The number $m$ outputted by the algorithm described previously gives the sufficient number of re-grasps needed to tie a given knot. For many knots, including the double coin knot, the number is 1. For these knots, since they are in a different topology class as the straight line segment, at least one re-grasp is needed.

### 7.4.3 Knot weaving with Da Vinci

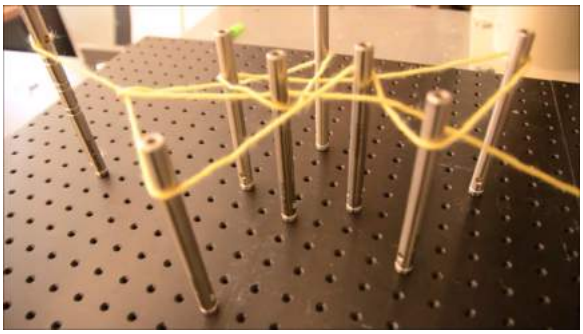
We conducted experiments with Da Vinci surgical robots, which have two symmetric high precision arms. We computed knot layouts and built fixtures to support string arranged at various heights, allowing weaving to be implemented with a single translation.

Figure 7.3a shows the layout of the type-1 crossings of a double-coin knot, arranged without re-grasps. After layout, the effector of the second arm grasped the tip of the string, and used a pure translation to complete the knot, as shown in Figure 7.3b and in the multimedia attachments.

Such arrangement requires many support structures laid out in the workspace of the robot. For simplicity, we also programmed the robot to just arrange the type-1 crossings of the string at the same heights around simpler fixtures. Figures 7.1a and 7.1b show the arrangement of a $7_1$ knot. Figures 7.6a and 7.6b shows the arrangement of a double coin knot.

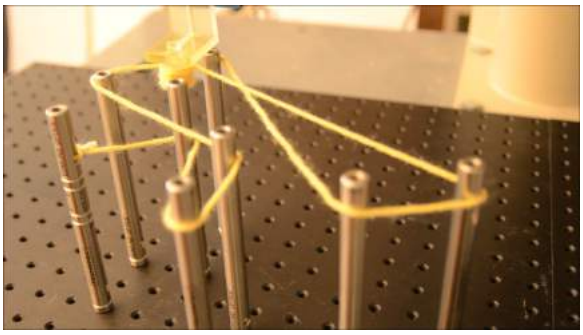


**(a)** Arranging type-1 crossings of a knot $8_{10}$ at the same height.

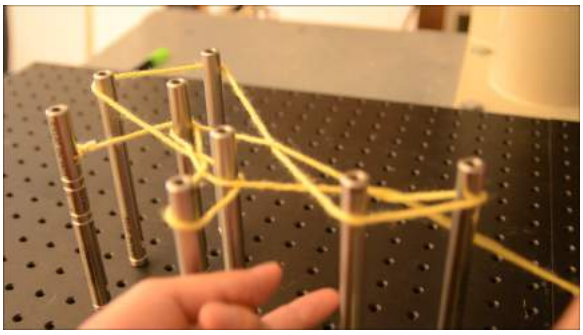


**(b)** Completing the arrangement of a knot $8_{10}$ by a human weaving the string.

**Figure 7.7:** Arranging a knot $8_{10}$ by robot and human collaborating together.

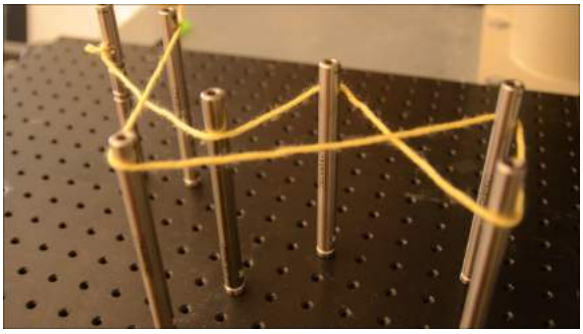


**(a)** Arranging type-1 crossings of a knot $9_{31}$ at the same height.

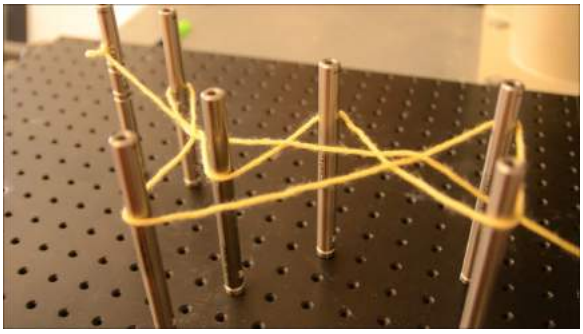


**(b)** Completing the arrangement of a knot $9_{31}$ by a human weaving the string.

**Figure 7.8:** Arranging a knot $9_{31}$ by robot and human collaborating together.



**(a)** Arranging type-1 crossings of a knot $9_{32}$ at the same height.



**(b)** Completing the arrangement of a knot $9_{32}$ by a human weaving the string.

**Figure 7.9:** Arranging a knot $9_{32}$ by robot and human collaborating together.

### 7.4.4 Robot human collaboration

When we arrange the segments of string that are not part of a weaving pattern at the same height, the robot arm weaves around arranged segments of string. Even though the locations of the string segments are known, the motion still may not be easy to perform for a robot. Humans, however, can perform that weaving motion easily.

For example, a double coin knot can be tied using the robot to lay out the structure, and allowing the human to finish the type-2 crossings. We used this technique to tie figure-eight knots, and knots $7_1$, $8_2$, $8_5$, $8_{10}$, $9_{31}$, $9_{32}$ from the standard knot table. All the listed knots can be tied with one re-grasp using a robot arm. Figures 7.7, 7.8, and 7.9 show the examples of a human collaborating with an Adept Cobra industrial arm to tie knot $8_{10}$, knot $9_{31}$ and knot $9_{32}$.

## 7.5 Complexity of knot tying

In the previous chapter, we analyzed how many fingers are needed to fold an arbitrary knot with polygonal constraint. And now we can also know the sufficient number of re-grasps needed to tie an arbitrary knot. Using the information, we can compile the following Table 7.1 and 7.2.

Many knots analyzed here are from a standard knot table, an example of which can be found here. We can see that the sufficient number of fingers needed to tie an arbitrary know grows as the number of crossings grow, while the sufficient number of re-grasps does not. For some knots that have more crossings, it may require fewer re-grasps to tie.

The results are not totally surprising. We can compute the sufficient number of fingers from only knowing the number of crossings of a given knot, without analyze the detail structure of the Gauss code. However, the number of re-grasps can only be computed if the detail of the Gauss code is known. In the last section of the previous chapter, we showed that the sufficient number of fingers needed to tie an arbitrary knot is not tight.

## 7.6 Conclusions

In this chapter, we derived a novel knot tying approach for both robots and human, by reducing the use of re-grasps by simulating the motion of a shuttle on a mechanical loom. An algorithm is presented to analyze the Gauss code to compute this small number of re-grasps. The strategy is demonstrated by a Da Vinci robot, and by human collaborating with an Adept Cobra industrial arm. Many knots can be tied with a single re-grasp, which is both an upper bound and a lower bound.

For future work, we are particularly interested in knots like the shoelace and sheepshank (see Figure 7.10); humans tie these knots by pulling loops through loops. We can identify these structures from the Gauss code, as Type II Reidemeister moves: for each adjacent appearance, crossings $i$ and $j$ have the same sign, and the sign is different from the crossings adjacent to the $ij$ (or $ji$) sequence. We would like to better understand how motions can be designed to mechanically simplify tying such knots.

**Table 7.1:** Knots examples and corresponding complexity.

| Knot | Example | No. of crossings | Grasping complexity | Re-grasping complexity |
|---|---|---|---|---|
| Overhand |  | 3 | 21 | 1 |
| Figure-eight |  | 4 | 30 | 1 |
| Knot $6_2$ |  | 6 | 48 | 1 |
| Knot $7_7$ |  | 7 | 57 | 2 |
| Double Coin ($8_{18}$) |  | 8 | 66 | 1 |
| Knot $8_2$ |  | 8 | 66 | 1 |
| Knot $8_3$ |  | 8 | 66 | 1 |
| Knot $8_4$ |  | 8 | 66 | 2 |

**Table 7.2:** Knots examples and corresponding complexity (continue).

| Knot | Example | No. of crossings | Grasping complexity | Re-grasping complexity |
|---|---|---|---|---|
| Knot $8_5$ |  | 8 | 66 | 1 |
| Knot $8_{10}$ |  | 8 | 66 | 1 |
| Knot $8_{17}$ |  | 8 | 66 | 2 |
| Knot $9_1$ |  | 9 | 75 | 1 |
| Knot $9_{30}$ |  | 9 | 75 | 1 |
| Knot $9_{31}$ |  | 9 | 75 | 1 |
| Knot $9_{32}$ |  | 9 | 75 | 1 |
| River knot |  | 25 | 219 | 4 |

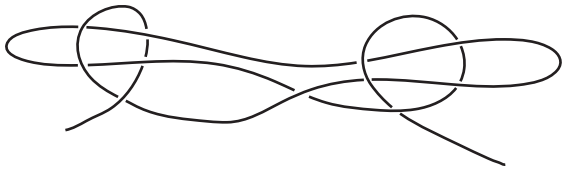


**Figure 7.10:** A sheepshank unknot.

# Chapter 8

# Conclusions and Future work

In this thesis, we study how the different forms of constraint can be applied in robot knot tying. We attacked the problem from several directions, and were able to tie many different knots using different generalized approaches. We discovered that by forcing the string to take a polygonal arc configuration, string achieves minimum deformation and is easy to describe and control. We believe this principle can be applied to general string manipulation scenarios, and can lead to simple string manipulation with minimum sensing feedbacks.

We first designed fixtures to tie different knots. Common characteristics of different knots and fixtures are extracted so that automated design processes can be presented. Knot tying process is separated into arrangement phase and tightening phase, and fixtures are used in both phases to automated the entire knot tying process. Several iterations of design are presented with different complexity, to tie different knots satisfying different constraints. Eventually, we were able to tie knots with great complexity such as Ruyi knots, that are even challenging for human to tie.

Fixtures are designed and built using rapid prototyping machines. Better engineering capabilities are desirable to improve the details of the fixtures so that the knot tying speed and reliability can be vastly improved. We expect better engineered fixtures to be developed to push the limit of autonomous knot tying.

We also explored the possibility to actively control the string to tie knots, rather than passively relying on the mechanisms on fixtures to lead the string to desired knot configuration. A simple control strategy that does not need to simulate string deformation was proposed in collaboration with Berenson, and was first applied to thread string through small workspace openings. The strategy was robust even under faulty sensing. The strategy was physically demonstrated with a Da Vinci surgical robot. Using the same robot, different knots are tied around generalized fixtures to demonstrate the benefits of fixtures.

We intend to combined approaches to push towards autonomous knot tying with robot arms under the assist of generalized fixtures. Several challenges need to be addressed to achieve autonomous knot tying. We first need to automate the placement of virtual vector fields for the control strategy. We also need to automate the placement of the generalized fixtures for different knots. We also need to figure out how to reduce the effect of the friction between the string and the fixtures. The first proposed idea is to poke the string loose through out the manipulation process to yield extra string near the grasped end of the string.

We analyzed the knot tying process from the perspective of how many contacts are needed to complete the task, and presented the first bound on the necessary and sufficient number of contacts to arrange an

arbitrary knot. First, a general polygonal configuration for the given knot is computed using the proved correct algorithm proposed. Then, extra links are added to fold arbitrary knots sequentially. The bound, however, is loose if we do not analyze the Gauss code for the given knot specifically. And the proposed number of contacts are too large to provide by current available robotic devices.

We eventually studied the knot tying process from the perspective of re-grasps. During knot tying process, re-grasps cannot be avoided due to the robot / human arm topology and the desired string topology. Inspired by the loom and the weaving, we derived an algorithm to analyze the Gauss code of each knot and yield a small sufficient number of re-grasps to arrange an arbitrary given knot. The tying strategy simulates the motion of the shuttle on the loom, and encourage a novel knot tying approach for both robots and human. The strategy is demonstrated with a Da Vinci robot, and also through the collaboration between a human and an Adept Cobra industrial arm.

Sometimes, knots need to be tied fast without error, such as during search and rescure. We believe devices can be designed, based on the knowledge we have acquired so far, so that they can quickly guide human operators to tie specific knots fast without error.

We observe similar simplified knot tying strategy when human tie unknots such as shoelace and sheepshank. We intend to better understand how motions can be designed to mechanically simplify tying such knots. We also would like to develop grippers that can allow knot arrangement without the use of re-grasping, starting with the use of magnets, and eventually lead to contactless manipulation.

We would like also to derive an algorithm that can analyze the Gauss code, and generate a much tighter bound on the number of fingers needed to tie an arbitrary knot. Through the analysis, we would like to answering several questions about knot tying, starting with the following few questions.

1. Did we overlook the importance of the flexibility of string in knot tying? If we do not require string to be line segment all the time, but can be stretched by infinite friction fingers if needed, can that reduce the sufficient number of fingers needed to tie an arbitrary knot?

2. Human tie knots using fingers, but our fingers do not serve as point contacts. Our fingers usually contact string at more than one point, acting like a rod. If we associate fingers as if they are on a rod, reduce the individual finger's independence, can we reduce the number of rods used to tie an arbitrary knot?

3. Can we mesh the analysis of the number of fingers needed to tie knots, and the number of re-grasps to tie knots together, and generate a small overall number that contains the use of least fingers and re-grasps combined? Is the derived strategy easier for robots to execute?

In conclusion, this work studies the how to tie knots with robots, pushing towards fast and reliable autonomous knot tying. Different analysis of the knot tying problem yields the following simple principle: forcing string into polygonal arc simplifies the description and control of the string. We believe this principle extends beyond the context of knot tying, and is applicable to general string manipulation, and even to cloth manipulation. We hope this work provides good insights into the problem of general flexible object manipulation, and motivates the use of simple mechanisms and support structures in flexible object manipulation.

# Bibliography


[ACPR11] Ted Ashton, Jason Cantarella, Michael Piatek, and Eric Rawdon. Knot tightening by constrained gradient descent. *Experimental Mathematics*, 20(1):57–90, 2011.

[Ada04a] C.C. Adams. *The Knot Book: An Elementary Introduction to the Mathematical Theory of Knots*. American Mathematical Society, 2004.

[Ada04b] Colin Adams. *The knot book: an elementary introduction to the mathematical theory of knots*. American Mathematical Society, 2004.

[ADG09] Timothy G. Abbott, Erik D. Demaine, and Blaise Gassend. A generalized carpenter's rule theorem for self-touching linkages. *CoRR*, abs/0901.1322, 2009.

[Adl14] Tim Adler. Knot theory and topology of 3-spaces. a triangulation for the figure-8 complement. Seminar: Knot Theory and its Applications, July 2014.

[Aex23] J. W. Aexander. Topological invariants of knots and links. *Trans. Amer. Math. Soc.*, 20:275–306, 1923.

[AGP+03] Ron Alterovitz, Ken Goldberg, Jean Pouliot, Richard Tascherau, and I-Chow Hsu. Sensorless planning for medical needle insertion procedures. In *IROS*, pages 3337–3343. IEEE, 2003.

[ALG+05] Ron Alterovitz, Andrew Lim, Kenneth Y. Goldberg, Gregory S. Chirikjian, and Allison M. Okamura. Steering flexible needles under markov motion uncertainty. In *IROS*, pages 1570–1575. IEEE, 2005.

[Ane99] C.N. Aneziris. *The Mystery of Knots: Computer Programming for Knot Tabulation*. K & E series on knots and everything. World Scientific, 1999.

[Arm83] M. A. Armstrong. *Basic Topology*. Springer-Verlag, New York, 1983.

[AS98] Pankaj K. Agarwal and Micha Sharir. Arrangements and their applications. In *Handbook of Computational Geometry*, pages 49–119. Elsevier Science Publishers B.V. North-Holland, 1998.

[Att99] Stephen W. Attaway. The mechanics of friction in rope rescue. *International Technical Rescue Symposium*, 1999.

[BcfaKB14] Matthew P. Bell, Weifu Wang (co-first author), Jordan Kunzika, and Devin Balkcom. Knot-tying with four-piece fixtures. *International Journal of Robotics Research (IJRR)*, vol 33, no. 11:1481–1489, Sep, 2014.

[BDD+99] Therese C. Biedl, Erik D. Demaine, Martin L. Demaine, Sylvain Lazard, Anna Lubiw, Joseph O'Rourke, Mark H. Overmars, Steve Robbins, Ileana Streinu, Godfried T. Toussaint, and Sue Whitesides. Locked and unlocked polygonal chains in 3d. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, 17-19 January 1999, Baltimore, Maryland.*, pages 866–867, 1999.

[Bel10] Matthew P. Bell. Flexible Object Manipulation. Technical Report TR2010-663, Dartmouth College, Computer Science, Hanover, NH, February 2010.

[Ber13] Dmitry Berenson. Manipulation of deformable objects without modeling and simulating deformation. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 4525–4532, Nov 2013.

[Bes03] Sergei Bespamyatnikh. Computing homotopic shortest paths in the plane. In *SODA*, pages 609–617. ACM/SIAM, 2003.

[BET04] Stephen Berard, Kevin Egan, and Jeffrey C. Trinkle. Contact modes and complementary cones. In *ICRA*, pages 5280–5286, 2004.

[BF06] James Burns and Andrew Fung. Shoelace knot assisting device, May 2006.

[BLGK13] Subhrajit Bhattacharya, David Lipsky, Robert Ghrist, and Vijay Kumar. Invariants for homology classes with application to optimal search and planning problem in robotics. *Ann. Math. Artif. Intell.*, 67(3-4):251–281, 2013.

[BLK11] Subhrajit Bhattacharya, Maxim Likhachev, and Vijay Kumar. Identification and representation of homotopy classes of trajectories for search-based path planning in 3d. In Hugh F. Durrant-Whyte, Nicholas Roy, and Pieter Abbeel, editors, *Robotics: Science and Systems*, 2011.

[BLK12] Subhrajit Bhattacharya, Maxim Likhachev, and V. Kumar. Topological constraints in search-based robot path planning. *Auton. Robots*, 33(3):273–290, 2012.

[BM08] Devin J. Balkcom and Matthew T. Mason. Robotic origami folding. *I. J. Robotic Res.*, 27(5):613–627, 2008.

[BM13] T. Bretl and Z. McCarthy. Quasi-static manipulation of a Kirchhoff elastic rod based on a geometric analysis of equilibrium configurations. *The International Journal of Robotics Research*, 33(1):48–68, June 2013.

[BMAP01] Sebastien J. Blind, Christopher C. McCullough, Srinivas Akella, and Jean Ponce. Manipulating parts with an array of pins: A method and a machine. *I. J. Robotic Res.*, 20(10):808–818, 2001.

[BPP08] J. Baranska, S. Przybyl, and P. Pieranski. Curvature and torsion of the tight closed trefoil knot. *The European Physical Journal B - Condensed Matter and Complex Systems*, 66(4):547–556, 2008.

[BRK99] Robert R. Burridge, Alfred A. Rizzi, and Daniel E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *I. J. Robotic Res.*, 18(6):534–555, 1999.

[BTG02] Devin J. Balkcom, Jeffrey C. Trinkle, and E. J. Gottlieb. Computing wrench cones for planar contact tasks. In *ICRA*, pages 869–875. IEEE, 2002.

[Bul] Bullet Physics library. http://bulletphysics.org/wordpress/.

[CDD+08] David Charlton, Erik D. Demaine, Martin L. Demaine, Gregory N. Price, and Yaa-Lirng Tu. A locked orthogonal tree. *CoRR*, abs/0801.4405, 2008.

[CDD+10] Robert Connelly, Erik D. Demaine, Martin L. Demaine, Sándor P. Fekete, Stefan Langerman, Joseph S. B. Mitchell, Ares Ribó, and Günter Rote. Locked and unlocked chains of planar shapes. *Discrete & Computational Geometry*, 44(2):439–462, 2010.

[CDIO04] Jason H. Cantarella, Erik D. Demaine, Hayley N. Iben, and James F. O'Brien. An energy-driven approach to linkage unfolding. In *Proceedings of the 20th ACM Symposium on Computational Geometry, Brooklyn, New York, USA, June 8-11, 2004*, pages 134–143, 2004.

[CDR00] Robert Connelly, Erik D. Demaine, and Günter Rote. Straighting polygonal arcs and convexifying polygonal cycles. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 432–442, 2000.

[CF77] Richard H. Crowell and Ralph H. Fox. *Introduction to knot theory*. Springer-Verlag, New York - Heidelberg - Berlin, 1977.

[Cha04] Mark Champion. Knot tying device, November 2004.

[Che97] Y. H. Chen. Determining parting direction based on minimum bounding box and fuzzy logics. *International Journal of Machine Tools and Manufacture*, 37(9):1189 – 1199, 1997.

[CR03] Yong Chen and David W. Rosen. A reverse glue approach to automated construction of multi-piece molds. *Journal of Computing and Information Science in Engineering*, 3(3):219–230, 2003.

[Cro04] Peter Cromwell. *Knots and Links*. Cambridge UP, 2004.

[DiY14] Mohd Azuwan Mat Dzahir and Shin ichiroh Yamamoto. Recent trends in lower-limb robotic rehabilitation orthosis: Control scheme and strategy for pneumatic muscle actuated gait trainers. *Robotics*, 3(2):120–148, 2014.

[Don90] Bruce Randall Donald. Planning multi-step error detection and recovery strategies. *I. J. Robotic Res.*, 9(1):3–60, 1990.

[EKL06] Alon Efrat, Stephen G. Kobourov, and Anna Lubiw. Computing homotopic shortest paths efficiently. *Comput. Geom.*, 35(3):162–172, 2006.

[FG92] M.L. Furst and K.Y. Goldberg. Low friction gripper, March 24 1992. US Patent 5,098,145.

[FSAB11] Barbara Frank, Cyrill Stachniss, Nichola Abdo, and Wolfram Burgard. Efficient motion planning for manipulation robots in environments with deformable objects. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2011, San Francisco, CA, USA, September 25-30, 2011*, pages 2180–2185. IEEE, 2011.

[GJS02] Herbert Goldstein, Charles P. Poole Jr., and John L. Safko. *Classical Mechanics*. Pearson Education, 2002.

[GRC98] Mariano Garcia, Andy Ruina, and Michael Coleman. Some results in passive-dynamic walking. In *Proceedings of the Euromech 375: Biology and Technology of Walking*, pages 268–275, Technical University of Munich, 1998.

[GS97] Dima Grigoriev and Anatol Slissenko. Computing minimum-link path in a homotopy class amidst semi-algebraic obstacles in the plane. In Teo Mora and Harold F. Mattson, editors, *AAECC*, volume 1255 of *Lecture Notes in Computer Science*, pages 114–129. Springer, 1997.

[GS98] Dima Grigoriev and Anatol Slissenko. Polytime algorithm for the shortest path in a homotopy class amidst semi-algebraic obstacles in the plane. In Volker Weispfenning and Barry M. Trager, editors, *ISSAC*, pages 17–24. ACM, 1998.

[GT96] Ashim Garg and Roberto Tamassia. A new minimum cost flow algorithm with applications to graph drawing. In Stephen C. North, editor, *Graph Drawing*, volume 1190 of *Lecture Notes in Computer Science*, pages 201–216. Springer, 1996.

[Hag06] Tobias Hagge. Every reidemeister move is needed for each knot type. In *Proceedings of the American Mathematical Society*, volume 134, No. 1, pages 295–301, 2006.

[Hat02] Allen Hatcher. *Algebraic topology*. Cambridge University Press, Cambridge, New York, 2002. Autre(s) tirage(s) : 2003,2004,2005,2006.

[HBAS11] Sami Haddadin, Bico Belder, and Alin Albu-Schaeffer. Reactive motion generation for robots in dynamic environments. In *Proceedings. IFAC 2011, World Congress, 28. Aug. - 02. Sep. 2011, Milano, Italy*, 2011.

[HH10] C. Hayashi and M. Hayashi. Unknotting number and number of reidemeister moves needed for unlinking. *arxiv:1021.4131*, pages 1–10, 2010.

[HHC98] Wade Henning, Frank Hickman, and Howie Choset. Motion planning for serpentine robots. In *ASCE Space and Robotics*, Albuquerque, NM, 1998.

[HIIH04] Minoru Hashimoto, Tomoaki Ichikawa, Shigeru Inui, and Yosuke Horiba. Dynamic manipulation of a string using a manipulator-parameter identification and control based on a rigid body link model. In *IEEE International Conference on Robotics and Automation*, pages 4815–4820, 2004.

[HK10] A. Henrich and L. H. Kauffman. Unknotting unknots. *ArXiv e-prints*, June 2010.

[HKK91] John E. Hopcroft, Joseph K. Kearney, and Dean B. Krafft. A case study of flexible object manipulation. *International Journal of Robotic Research*, 10(1):41–50, 1991.

[HL01] J. Hass and J. Lagarias. The number of Reidemeister moves needed for unknotting. *J. Amer. Math. Soc.*, 14:399,428, 2001.

[HN10] J. Hass and T. Nowik. Unknot diagrams requiring a quadratic number of Reidemeister moves to untangle. *Discrete Compute. Geom.*, 44:91–95, 2010.

[HS94] John Hershberger and Jack Snoeyink. Computing minimum length paths of a given homotopy class. *Comput. Geom.*, 4:63–97, 1994.

[HW00a] Dominik Henrich and Heinz Wörn. *Robot manipulation of deformable objects*. Springer, 2000.

[HW00b] S. Hirai and T. Wada. Indirect simultaneous positioning of deformable objects with multi-pinching fingers based on an uncertain model. *Robotica*, 18(1):3–11, January 2000.

[II85] H. Inoue and M. Inaba. Hand-eye coordination in rope handling. *Robotics Research: The first International Symposium*, pages 163–174, 1985.

[Kau91] L.H. Kauffman. *Knots and Physics*. K & E series on knots and everything. World Scientific, 1991.

[Kau95] L.H. Kauffman. *Knots and Applications*. K & E series on knots and everything. World Scientific, 1995.

[KBM06] Rahul Khardekar, Greg Burton, and Sara McMains. Finding feasible mold parting directions using graphics hardware. *Computer-Aided Design*, 38(4):327 – 341, 2006. Symposium on Solid and Physical Modeling 2005.

[KNMB00] Makoto Kudo, Yasuo Nasu, Kazuhisa Mitobe, and Branislav Borovac. Multi-arm robot control system for manipulation of flexible materials in sewing operation. *Mechatronics*, 10(3):371 – 402, 2000.

[KP10] F Khalil and P Payeur. Dexterous robotic manipulation of deformable objects with multi-sensory feedback – a review. In In-Teh, editor, *Robot Manipulators, Trends and Development*, chapter 28, pages 587–621. 2010.

[KW01a] Hyosig Kang and John T. Wen. Endobot: a robotic assistant in minimally invasive surgeries. In *Proc. IEEE International Conference on Robotics and Automation*, volume 2, pages 2031–2036, 2001.

[KW01b] Hyosig Kang and J.T. Wen. Robotic assistants aid surgeons during minimally invasive procedures. *Engineering in Medicine and Biology Magazine, IEEE*, 20(1):94–104, Jan 2001.

[L.98] Ricca R. L. Applications of knot theory in fluid mechanics. *Knot Theory*, 42:321–346, 1998.

[LA99] Liang Lu and Srinivas Akella. Folding cartons with fixtures: A motion planning approach. In *IEEE International conference on Robotics and Automation*, May 1999.

[LA00] Liang Lu and Srinivas Akella. Folding cartons with fixtures: A motion planning approach. *IEEE Transactions on Robotics and Automation*, 16(4):346–356, August 2000.

[Lic97] W.B.R. Lickorish. *An Introduction to Knot Theory*. Graduate Texts in Mathematics. Springer New York, 1997.

[Lic04] W.B. Raymond Lickorish. *An Introduction to Knot Theory*. Springer, 2004.

[Liv93] Charles Livingston. *Knot theory*. The carus mathematical monographs, Volume Twenty Four. The mathematical Association of America, Washington D.C., 1993.

[LPMT84] Tomas Lozano-Perez, Matthew Mason, and Russell H. Taylor. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1), 1984.

[LS96] Joel Langer and David A. Singer. Lagrangian aspects of the Kirchhoff elastic rod. *SIAM Rev.*, 38(4):605–618, December 1996.

[LT05] Guanfeng Liu and Jeffrey C. Trinkle. Complete path planning for planar closed chains among point obstacles. In *Robotics: Science and Systems I, June 8-11, 2005, Massachusetts Institute of Technology, Cambridge, Massachusetts*, pages 33–40, 2005.

[Man04] Vassily Manturov. *Knot theory*. Chapman & Hall/CRC, Boca Raton, FL, 2004.

[Mas01] Matthew T. Mason. *Mechanics of Robotic Manipulation*. MIT Press, Cambridge, MA, August 2001.

[McG90a] Tad McGeer. Passive dynamic walking. *I. J. Robotic Res.*, 9(2):62–82, 1990.

[McG90b] Tad McGeer. Passive walking with knees. In *IEEE International Conference on Robotics and Automation, 1990*, pages 1640–1645 vol.3, May 1990.

[McG91] Tad McGeer. Passive dynamic biped catalogue, 1991. In *Experimental Robotics II, The 2nd International Symposium, Toulouse, France, June 25-27, 1991*, pages 465–490, 1991.

[MCS05] J.H.Maddocks M. Carlen, B. Laurie and J. Smutny. Biarcs, global radius of curvature, and the computation of ideal knot shapes. *Physical and numerical models in knot theory*, 36 of Ser. Knots Everything:75–108, 2005.

[MEGH+14] Pawe Maciejasz, Jörg Eschweiler, Kurt Gerlach-Hahn, Arne Jansen-Troy, and Steffen Leonhardt. A survey on robotic devices for upper limb rehabilitation. *Journal of NeuroEngineering and Rehabilitation*, 11:3 doi:10.1186/1743-0003-11-3, 2014.

[MK04] Mark Moll and Lydia E. Kavraki. Path planning for minimal energy curves of constant length. In *ICRA*, pages 2826–2831, 2004.

[MK06] Mark Moll and Lydia E Kavraki. Path Planning for Deformable Linear Objects. *IEEE Transactions on Robotics*, 22(4):625–636, 2006.

[MTO+03] T. Morita, J. Takamatsu, K. Ogawara, H. Kimura, and K. Ikeuchi. Knot planning from observation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[NALRL13] D. Navarro-Alarcon, Y. Liu, J.G. Romero, and P. Li. Visually Servoed Deformation Control by Robot Manipulators. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[O'H03] J. O'Hara. *Energy of Knots and Conformal Geometry*. K & E series on knots and everything. World Scientific, 2003.

[Öst01] Olof-Petter Östlund. Invariants of knot diagrams and relations among reidemeister moves. *Journal of Knot Theory Remifications 10*, No. 8:1215–1227, 2001.

[Pai02] Dinesh K. Pai. Strands: Interactive Simulation of Thin Solids using Cosserat Models. *Comput. Graph. Forum*, 21(3):347–352, 2002.

[PGB14] Calder Phillips-Grafflin and Dmitry Berenson. A representation of deformable objects for motion planning with no physical simulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 2014.

[PLK02] Jeff Phillips, Andrew M. Ladd, and Lydia E. Kavraki. Simulated knot tying. In *IEEE International conference on Robotics and Automation*, pages 841–846, May 2002.

[PLPC04] Peng Pan, Kevin M. Lynch, Michael A. Peshkin, and J. Edward Colgate. Static single-arm force generation with kinematic constraints. In *ICRA*, pages 2794–2800, 2004.

[PPAG14] Sachin Patil, Jia Pan, Pieter Abbeel, and Ken Goldberg. Planning curvature and torsion constrained ribbons in 3d with application to intracavitary brachytherapy. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, August, 2014.

[Raw98] Eric J. Rawdon. Approximating the thickness of a knot. *Ideal knots*, 19 of Ser. Knots Everything:143–150, 1998.

[RB95] Elon Rimon and Joel Burdick. New bounds on the number of frictionless fingers required to immobilize 2D objects. In *IEEE International conference on Robotics and Automation*, pages 751–757, Nagoya, Aichi, Japan, May 1995.

[RB96] Elon Rimon and Andrew Blake. Caging 2D bodies by 1-parameter two-fingered gripping systems. In *IEEE International conference on Robotics and Automation*, volume 2, pages 1458–1464, Minneapolis, MN, April 1996.

[RB99] Elon Rimon and Andrew Blake. Caging planar bodies by one-parameter two-fingered gripping systems. *I. J. Robotic Res.*, 18(3):299–318, 1999.

[RBC12] David Rollinson, Austin Buchan, and Howie Choset. Virtual chassis for snake robots: Definition and applications. *Advanced Robotics*, October 2012.

[RCR+11] David Rojas, Sayra Cristancho, Claudia Rueda, Lawrence E. M. Grierson, Alex Monclou, and Adam Dubrowski. The validation of an instrumented simulator for the assessment of performance and outcome of knot tying skill: A pilot study. In *Medicine Meets Virtual Reality 18 - NextMed, MMVR 2011*, pages 517–523, 2011.

[Rei27] Kurt Reidemeister. Elementare begrndung der knotentheorie. *Abhandlungen aus dem Mathematischen Seminar der Universitt Hamburg*, 5(1):24–32, 1927.

[RLA06a] Samuel Rodríguez, Jyh-Ming Lien, and Nancy M. Amato. Planning motion in completely deformable environments. In *IEEE International conference on Robotics and Automation*, pages 2466–2471, Orlando, FL, May 2006.

[RLA06b] Samuel Rodríguez, Jyh-Ming Lien, and Nancy M. Amato. Planning motion in completely deformable environments. In *ICRA*, pages 2466–2471. IEEE, 2006.

[Rol76] D. Rolfsen. *Knots and Links*. AMS/Chelsea Publication Series. AMS Chelsea Pub., 1976.

[RS90] B. Ravi and M. N. Srinivasan. Decision criteria for computer-aided parting surface design. *Computer-Aided Design*, 22(1):11–18, 1990.

[RSBH12] Matthias Rambow, Thomas Schauß, Martin Buss, and Sandra Hirche. Autonomous Manipulation of Deformable Objects based on Teleoperated Demonstrations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[SGV+13] John Schulman, Ankush Gupta, Sibi Venkatesan, Mallory Tayson-Frederick, and Pieter Abbeel. A case study of trajectory transfer through non-rigid registration for a simplified suturing scenario. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2013.

[SHLA13] John Schulman, Jonathan Ho, Cameron Lee, and Pieter Abbeel. Learning from Demonstrations Through the Use of Non-Rigid Registration. In *International Symposium on Robotics Research (ISRR)*, 2013.

[sho] Shoe tying robot. http://www.youtube.com/watch?v=XrA7DR0u0uI. Accessed: 2014-02-17.

[SI06] Mitul Saha and Pekka Isto. Motion planning for robotic manipulation of deformable linear objects. In *Proc. IEEE International Conference on Robotics and Automation*, pages 2478–2484, May 2006.

[SI07] Mitul Saha and Pekka Isto. Manipulation planning for deformable linear objects. *IEEE Transaction on Robotics*, 23(6):1141–1150, December 2007.

[SIL06] Mitul Saha, Pekka Isto, and J.-C. Latombe. Motion planning for robotic knot tying. In *Proc. International Symposium on Experimental Robotics*, July 2006.

[Sim01] Jonathan Simon. Physical knots. *Series on Knots and Everything*, 1:1–30, 2001.

[Sin04] Wamis Singhatat. Intracorporeal knot tier, April 2004.

[SP09] Jerzy Smolen and Alexandru Patriciu. Deformation Planning for Robotic Soft Tissue Manipulation. In *2009 Second International Conferences on Advances in Computer-Human Interactions*, pages 199–204, February 2009.

[SSLT07] Nir Shvalb, Moshe Shoham, Guanfeng Liu, and Jeffrey C. Trinkle. Motion planning for a class of planar closed-chain manipulators. *I. J. Robotic Res.*, 26(5):457–473, 2007.

[Tam87] Roberto Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444, 1987.

[Ted09] Russ Tedrake. LQR-trees: Feedback motion planning on sparse randomized trees. In Jeff Trinkle, Yoky Matsuoka, and José A. Castellanos, editors, *Robotics: Science and Systems V, University of Washington, Seattle, USA, June 28 - July 1, 2009*. The MIT Press, 2009.

[THL+02] Tanya Tickel, David Hannon, Kevin M. Lynch, Michael A. Peshkin, and J. Edward Colgate. Kinematic constraints for assisted single-arm manipulation. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002, May 11-15, 2002, Washington, DC, USA*, pages 2034–2041, 2002.

[TM02] Jeffrey C. Trinkle and R. James Milgram. Complete path planning for closed kinematic chains with spherical joints. *I. J. Robotic Res.*, 21(9):773–790, 2002.

[TMTR10] Russ Tedrake, Ian R. Manchester, Mark M. Tobenkin, and John W. Roberts. LQR-trees: Feedback Motion Planning via Sums-of-Squares Verification. *I. J. Robotic Res.*, 29(8):1038–1052, 2010.

[Tra83] Bruce Trace. On the reidemeister moves of a classical knot. In *Proceedings of the American Mathematical Society*, volume 89, No. 4, page 722724, 1983.

[TS03] R. H. Taylor and D. Stoianovici. Medical robotics in computer-integrated surgery. *IEEE Transactions on Robotics and Automation*, 19(5):765–781, October 2003.

[WAH06] Hidefumi Wakamatsu, Eiji Arai, and Shinichi Hirai. Knotting/unknotting manipulation of deformable linear objects. *International Journal of Robotics Research*, 25:371–395, 2006.

[WB16a] Weifu Wang and Devin Balkcom. Grasping and folding knots. In *IEEE International Conference on Intelligent Robots and Systems, to appear*, 2016.

[WB16b] Weifu Wang and Devin Balkcom. Towards tying knots precisely. In *IEEE International Conference on Intelligent Robots and Systems, to appear*, 2016.

[WBB14] Weifu Wang, Matthew P. Bell, and Devin J. Balkcom. Towards arranging and tightening knots and unknots with fixtures. In *Algorithmic Foundations of Robotics XI - Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics, WAFR 2014, 3-5 August 2014, Boğaziçi University, İstanbul, Turkey*, pages 677–694, 2014.

[WBB15a] Weifu Wang, Matthew P. Bell, and Devin J. Balkcom. Towards arranging and tightening knots and unknots with fixtures. *IEEE T. Automation Science and Engineering*, 12(4):1318–1331, 2015.

[WBB15b] Weifu Wang, Dmitry Berenson, and Devin J. Balkcom. An online method for tight-tolerance insertion tasks for string and rope. In *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, pages 2488–2495, 2015.

[WD09] Guowu Wei and J.S. Dai. Geometry and kinematic analysis of an origami-evolved mechanism based on artmimetics. In *Reconfigurable Mechanisms and Robots, 2009. ReMAR 2009. ASME/IFToMM International Conference on*, pages 450–455, June 2009.

[WH04] Hidefumi Wakamatsu and Shinichi Hirai. Static modeling of linear object deformation based on differential geometry. *International Journal of Robotic Research*, 23(3):293–311, March 2004.

[Whi49a] J. H. C. Whitehead. Combinatorial homotopy. i. *Bull. Amer. Math. Soc.*, 55:213–245, 03 1949.

[Whi49b] J. H. C. Whitehead. Combinatorial homotopy. ii. *Bull. Amer. Math. Soc.*, 55(5):453–496, 05 1949.

[WHKK01] T Wada, S Hirai, S Kawarnura, and N Kamiji. Robust manipulation of deformable objects by a simple PID feedback. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2001.

[WWDL10] Huijuan Wang, Shuxin Wang, Jienan Ding, and Haifeng Luo. Suturing and tying knots assisted by a surgical robot system in laryngeal mis. *Robotica*, 28(2):241–252, 2010.

[WYT+06] Hidefumi Wakamatsu, Tatsuya Yamasaki, Akira Tsumaya, Eiji Arai, and Shinichi Hirai. Dynamic modeling of linear object deformation considering contact with obstacles. In *Proc. of 9th International Conference on Control, Automation, Robotics, and Vision*, Singapore, December 2006.