FLEXIBLE OBJECT MANIPULATION

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

in

Computer Science

by

Matthew Bell

DARTMOUTH COLLEGE

Hanover, New Hampshire

February 2010

Dartmouth Computer Science Technical Report TR2010-663

Examining Committee:

(chair) Devin Balkcom

Scot Drysdale

Tanzeem Choudhury

Daniela Rus

Brian W. Pogue, Ph.D.
Dean of Graduate Studies

Abstract

Flexible objects are a challenge to manipulate. Their motions are hard to predict, and the high number of degrees of freedom makes sensing, control, and planning difficult. Additionally, they have more complex friction and contact issues than rigid bodies, and they may stretch and compress. In this thesis, I explore two major types of flexible materials: cloth and string.

For rigid bodies, one of the most basic problems in manipulation is the development of immobilizing grasps. The same problem exists for flexible objects. I have shown that a simple polygonal piece of cloth can be fully immobilized by grasping all convex vertices and no more than one third of the concave vertices. I also explored simple manipulation methods that make use of gravity to reduce the number of fingers necessary for grasping. I have built a system for folding a T-shirt using a 4 DOF arm and a fixed-length iron bar which simulates two fingers.

The main goal with string manipulation has been to tie knots without the use of any sensing. I have developed single-piece fixtures capable of tying knots in fishing line, solder, and wire, along with a more complex track-based system for autonomously tying a knot in steel wire. I have also developed a series of different fixtures that use compressed air to tie knots in string. Additionally, I have designed four-piece fixtures, which demonstrate a way to fully enclose a knot during the insertion process, while guaranteeing that extraction will always succeed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

From observing humans, we can infer that flexible manipulation skills require a high degree of dexterity, as well as the ability to see and intuitively analyze objects as they are manipulated. Children need help learning to tie their shoes. We can learn to fold paper into an origami flower, but only after practicing with simpler constructions. We must fold many loads of laundry before we learn how to do so efficiently, and in such a way that our clothes are not wrinkled when we want to put them on at a later time. In this thesis, I will show that various simplifications can be made to reduce the complexity of manipulators and the need for sensing when handling flexible materials. I will focus on laundry folding and knot tying.

We can simplify cloth folding by using gravity to supplement a small set of manipulators. We have built a system for folding a T-shirt, using a fixed-length iron bar as a manipulator (Figure 1.1). Magnets on the shirt function as stand-ins for a more complex gripper, such as the one developed by Shibata [75]. Based on this system, we can make some observations on the behavior of cloth in the presence of gravity, and use these observations to build a basic framework that defines the types of grasps that are possible with only two fingers.

Flattening the cloth in preparation for folding is the biggest challenge. We would like to fully immobilize a piece of cloth in a flat configuration, which is equivalent to the problem of finding immobilizing grasps for rigid bodies. For any shape, it is always at least necessary to grasp the convex corners, and it may even seem that this is true for all shapes. However, there exist shapes for which rotational motion is possible even if all convex corners are grasped. Figure 1.2 shows such a polygon, with arrows indicating the direction of rotation. This polygon is part of a class of polygons that we call pinwheels. Using pinwheels, we can show that there exist polygons for which a grasp consisting of the convex vertices plus one third of the concave vertices is necessary.

Figure 1.1: Shirt folded with robot arm

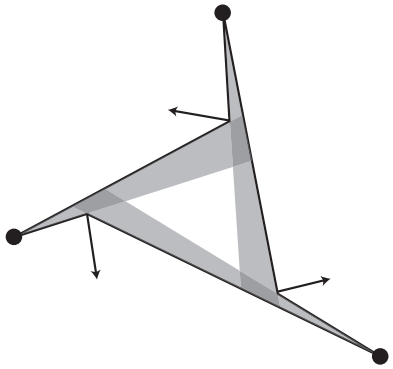


Figure 1.2: Polygon requiring extra fingers

Figure 1.3: Track-based fixture for an overhand knot

We have also shown that this many fingers are sufficient to grasp any polygon, thus giving tight bounds on the number of fingers needed to immobilize a non-stretchable cloth polygon.

String can be thought of as one-dimensional cloth; in fact, we use string as a fundamental unit in our cloth grasping proofs. Our main goal with string manipulation has been to tie knots without the use of any sensing. We have developed single-piece fixtures capable of tying knots in fishing line, solder, and wire. Lessons learned from the single-piece fixtures led to a spectrum of different types of fixtures that work with both wire and string. These fixtures use air and motors for insertion, and range from simple tubes to complex multi-walled tubes and tracks with sliders.

Many of the knot fixtures rely on the fact that when pulled, string tends towards the shortest path between the pulled points. Within a curved tube, string moves to the inside of the curve. If we somehow insert string into a tube in the shape of a knot, we can use this property to make it possible to extract the string while maintaining the knot. We use various methods to accomplish this, from slits to removable pieces of the fixture.

We use two main methods for inserting string or wire into a knot fixture: manual/motorized pushing for wire, and compressed air for string. We built a motorized feeding mechanism for wire that uses rubber rollers to insert wire into our knot fixtures. This feeder has been combined with a track-based fixture to create a system capable of tying simultaneous overhand knots in multiple strands of steel wire (Figure 1.3). Additionally, we developed several different types of fixtures that rely on compressed air for insertion.

1.1 Related Work

Our work on cloth grasping is related to rigid body grasping, capturing and caging rigid bodies, deformable object manipulation, and cloth simulation, manipulation, and sensing. Our knot tying research is related to fixturing, string and linear deformable object models, robotic knot tying, snake robots and steerable needles, and extraction of parts from molds.

1.1.1 Rigid Body Grasping

We are interested in grasping cloth and string; however, most existing work on grasping has focused on rigid bodies. Numerous papers have been written on the subject, as evidenced in Bicchi and Kumar's survey of theoretical work on grasping [7]. The listing here is meant to be a subset of rigid body grasping work that is closest to our work.

Early work on immobilizing grasps was done by Reuleaux in 1876 [64]. Reuleaux devised a geometric method for analyzing a grasp of a planar rigid body by a set of frictionless points. However, it was not possible to easily generate a minimal grasp using Reuleaux's method.

Nguyen developed algorithms for constructing force-closure grasps for a given planar object [50]. Mishra, Schwartz, and Sharir found bounds on the number of fingers needed to grasp a rigid object [49]. Specifically, to hold a rigid object at equilibrium, 4 fingers suffice for a two-dimensional object, and 7 fingers suffice for a three-dimensional object. Rimon and Burdick refined this result for two dimensions by showing that 3 convex fingers suffice to immobilize any smooth or polygonal planar object [66].

Zhu *et al.* gave a metric for analyzing how well a given grasp is capable of withstanding external forces [76], which can be used to compare grasps produced by different algorithms. Jia developed a fast algorithm for computing all antipodal grasps that place a parametric plane curve in force closure [38].

1.1.2 Caging

Perfect immobilization is hard for flexible objects; instead, approximate or partial immobilization may be required. Caging is a weaker form of grasping, where the object might not be immobilized, but is instead prevented from leaving some bounded region (motions to move it to infinity do not exist). Caging may be weaker, but it can also be more useful, as it allows for some uncertainty in the position of the object to be grasped. In some sense, we can state that our knot fixtures are caging the string in a configuration that happens to also tie the string in a knot.

Kuperberg posed the 2D caging problem in 1990 [45]. Rimon and Blake developed an algorithm to answer Kuperberg's problem [65]. They determined the maximal set of arm and manipulator configurations that resulted in the caging of an object. This set could be used to cage an object in an unknown configuration, and to slowly contract the caging grasp to an immobilizing grasp.

Erickson *et al.* examined the use of three disc-shaped robots for capturing an arbitrary convex object in the plane [23, 24]. They fix the locations of two of the robots, and determine the "capture region", which is the set of configurations of the third robot that capture the object. They show that determining the capture region is equivalent to a visibility problem.

1.1.3 Deformable Object Manipulation

For rigid bodies, grasping and manipulation are often separate operations, although they are occasionally combined for the purpose of removing error. Once a part is in an immobilizing grasp (which can be achieved with a small number of fingers), end effector motions result in predictable changes in configuration. However, for deformable objects, grasping and manipulation are closely coupled. A grasp may deform part of an object in a way that causes major changes in other portions of the object. As with rigid bodies, deformable objects are predictable if they are fully immobilized; in practice, such immobilization requires more fingers than are available.

Wakamatsu *et al.* developed the concept of bounded force closure for deformable parts [84]. Bounded force closure is an extension of force closure in which the grasp can withstand external forces up to some bound. They applied this to linear deformable objects as an example.

Howard and Bekey developed a system for generalized handling of 3D deformable objects [34]. They developed a ball and spring model, and used machine learning to compute deformation characteristics for different objects. These characteristics were then used to grasp unknown objects.

Hirai *et al.* worked on combining grasping and manipulation for deformable objects [32]. They developed a robust control system that incorporated tactile and vision data to influence controls, and experimentally verified their control scheme using a planar deformable sponge.

Gopalakrishnan and Goldberg created the concept of deform closure [29]. They define a deformable object to be in deform closure if positive work is needed to release the part, and show that if a given set of frictionless contacts holds a rigid object in form closure, the same set of contacts holds the equivalent deformable object in deform closure.

Cheong *et al.* gave bounds for the number of fingers that immobilize an acyclic flexible chain of hinged polygons [18]. They show that $(p + 2)$ frictionless point contacts suffice to immobilize a chain of $p \neq 3$ polygons if the polygons do not have parallel edges. An arbitrary chain of p polygons can be immobilized by at most $(p + 3)$ contacts.

Rodríguez, Lien, and Amato worked on motion planning in an environment where every object is deformable [67]. They focus on maintaining constant volume for all deformable objects, and use a tree-based planner to explore the system. This type of planning can also be applied to grasping problems in deformable environments.

1.1.4 Cloth Simulation

The ubiquitous presence of cloth has led to extensive research in simulation, such as in the field of computer graphics, where realistic cloth is very desirable.

The earliest cloth model for simulation was developed by Weil in 1986 [86]. Weil's model uses a rectangular grid to represent the cloth, and analyzes a piece of cloth that has been suspended from a set of fixed points. First, the resulting shape is approximated based on the constraints of the fixed points. This is followed by a relaxation process that incorporates distance constraints present in cloth.

In 1987, Feynman developed a physically-based cloth model, which was also designed to compute static configurations of draped cloth [25]. This model also uses a grid of points spaced out over the cloth surface, and tries to minimize the energy of the cloth by relaxing individual points. The cloth energy function is based on physical cloth, and incorporates stretching, bending, and buckling, among other properties.

Breen *et al.* worked on developing a cloth model to analyze the draping behavior of different types of cloth [9]. This model used a particle mesh, where particle behavior is governed by a set of energy functions that represent properties such as gravity, bending, and stretching. They used a standard cloth testing system, the Kawabata Evaluation System, to measure the mechanical properties of different types of cloth, and used these properties as inputs to the cloth model. This allowed them to test draping behavior of the same cloth shape using different materials.

Baraff and Witkin published one of the seminal works on cloth simulation, which focused on modifying cloth simulators to allow for larger time steps [3]. They developed an implicit integration method that uses direct constraint satisfaction, which yields computation times on the order of seconds per frame of a 30 Hz animation.

Choi and Ko developed a cloth simulation method that attempts to realistically model buckling [19]. They focus on correctly displaying the results of buckling, while preventing stability issues caused by buckling. This model is also particle-based, and uses a semi-implicit integrator to enable large time steps.

1.1.5 Cloth Manipulation

Work has also been done on manipulating cloth in the real world. Patton *et al.* developed an adaptive force feedback controller for handling cloth [58]. This controller was used to apply a desired tension to cloth, and to straighten wrinkles in the cloth. They simplified grasping by attaching wooden rods along two sides of the cloth, and grasped one of these rods with the robot's end effector. The other rod was fixed to the table. Additionally, they assumed that the cloth handling only took place in 2D. The primary focus was the use of a force/torque sensor to estimate the tension and configuration of the cloth.

Cloth manipulation has been used in various laundry folding projects; typically, only a few fingers are used for grasping in these projects. Ono *et al.* have worked on a manipulator for cloth handling [53]. This manipulator consists of a pair of fingers that grasp one point of a piece of cloth, and is designed for picking up cloth rectangles from a stack. One finger is a blade that slides between pieces of cloth, and the second is a piece of balsa wood that pins the cloth against the blade. They make use of sensors to verify that only one piece of cloth was grasped.

Ono *et al.* also developed a cooperative system combining touch and vision to unfold cloth [54]. They use the gripper mentioned in their previous work, with the addition of a stationary camera and a second camera attached to a robot arm. They developed a very basic planner for unfolding a rectangular piece of cloth with one fold in it. They analyzed the outline of the folded cloth, estimated the location of a moved corner, tried to grasp it, and then simply used straight line motion to move the grasped corner to its final location.

Hamajima and Kakikura worked on isolating individual pieces of cloth from a mass of washed clothing, and on unfolding the selected item [30]. They segment an image of a mass of clothing based on color, and select a point to grasp within a single piece of cloth. Their system uses a gripper that consists of a pair of rubberized wheels, which rotate inward to pull a small fold of cloth between them. The wheels are then pushed together to hold this fold. They use a pair of robot arms with such grippers to grasp cloth on a hemline, and to spread out the cloth along this line.

Kaneko and Kakikura continued this work by focusing on classifying the type of cloth after an initial grasp [40]. Once the cloth has been hung by a pair of robot arms, they classify the resulting shape as one of three types based on the protruding regions. Given this classification, they would measure portions of the cloth, and try to identify the shape as an example of a towel, shirt, or pants. They state that their classifier had a success rate of about 90%.

Salleh *et al.* developed a system consisting of two robot arms in which they traced cloth boundaries with grippers to flatten towels [71]. They used a camera and standard image processing techniques to find and grasp one corner of the cloth. The second robot arm loosely gripped the towel next to this corner, and dragged the gripper along the towel until an adjacent corner was reached. At this point, the hanging cloth is essentially flattened. Their system has a success rate of about 50%, although they can detect failure and repeat portions of the process as needed.

Salleh *et al.* refined this result by developing an improved gripper to avoid dropping the cloth during the tracing process [72]. The key difference is that the tracing gripper actually consists of a pair of grippers, separated by a linear actuator that allows the distance between them to change. During the tracing motion, one gripper is always holding the cloth securely while the other slides along the cloth. With this method, they were able to improve their success rate to 75%.

Kita *et al.* worked on developing a deformable model for clothing, and used stereo vision to match this model to the position of an observed hanging pullover [43, 44]. Their model consists of 20 nodes interconnected by springs. They suspend this model in gravity, and match it to the observed cloth. At this point, they estimate a grasp location in 3D space for a second manipulator.

Shibata *et al.* use a single-armed robot with multiple grippers to unfold a hemmed, rectangular piece of cloth [75]. Their gripping mechanism uses four fingers that are capable of grasping along two axes. Pairs of fingers can also be spread apart to grasp two distinct points on the cloth. They successfully pick up a partially folded cloth rectangle and spread out the two pairs of fingers along an edge to flatten the cloth in mid-air.

1.1.6 Cloth Immobilization

Our work on cloth immobilization is dependent on several concepts in computational geometry. In our work, we develop a new type of polygon skeleton, which we refer to as a support tree. There are two major types of polygon skeletons that are similar to our support trees. The first is the medial axis [60], which has the same number of vertices and edges as a support tree. However, medial axes allow for curved edges. The second similar type of skeleton is the straight skeleton [1], which has straight edges, but contains more vertices and edges than are needed for a support tree.

Our cloth immobilizing work also depends on concepts in visibility, such as those surveyed by Ghosh [28], and in triangulation and its applications to the art gallery problem, as explored by O'Rourke [55].

Finally, the problem of immobilizing cloth is related to the problem of trying to determine if a structure consisting only of cables is infinitesimally rigid when it is pinned at a set of points. This type of problem is briefly mentioned in Connelly's work on tensegrities and rigidity theory [20].

1.1.7 Cloth Sensing

One of the major difficulties in cloth manipulation is locating the cloth. As seen in previous sections, most existing systems use vision to infer cloth position, with additional input from strain gauges or other sensors to determine if and how much cloth is grasped between fingers. Other methods are also possible, however.

Nishikawa *et al.* proposed the use of an ultrasonic sensor for measuring a cloth surface in 3D [51]. The use of an ultrasonic sensor removes any problems posed by colored materials. They experimentally found that their sensor gave them an accuracy down to 1 mm. They measured the position of a shirt on a person by manually moving the sensor on an arm around the person's torso at discretized heights, yielding a contour map of the shirt.

Pritchard and Heidrich developed a motion capture system for determining the geometry of a piece of cloth [61]. Their system uses three cameras to produce a partial description of the cloth's geometry. They use the SIFT algorithm to find feature points in an arbitrary pattern on the cloth. This information is combined with the partial geometry to produce a full parametric model of the cloth surface. This system reconstructs each frame individually, which can result in motion discontinuities between frames.

Scholz and Magnor used optical flow between video frames to compute 3D scene flow [73]. This method works best when the cloth has a detailed texture. In regions without detailed texture information, the geometry is interpolated using a grid-based deformable cloth model. To avoid accumulated error over multiple frames, the mesh is constrained to match the silhouette visible in video data.

Scholz *et al.* further developed this work by using cloth with a color-coded pattern [74]. The pattern is an M-array, which has the property that every 3×3 neighborhood of a point uniquely identifies the point. This allows a very accurate reconstruction to be computed, which is filled in as necessary for deep folds that have poor visibility even with multiple cameras.

RFID has the advantage of not suffering from occlusion, and of uniquely identifying each point. As far as we are aware, no prior work exists for using RFID tags to uniquely identify points on a piece of cloth. However, RFID tags have been used for localization. Song, Haas and Caldas developed a method for locating materials at a construction site using RFID tags combined with GPS data [78]. In this system, rather than having each object know its location, a mobile unit (in this case, the construction supervisor) moves around the construction site with a GPS receiver and a RFID reader. When tags are detected, they are known to be within a certain radius of the currently reported location from the GPS receiver. This radius is based on the properties of the RFID tags and reader.

1.1.8 Fixturing

Fixtures present a way to manipulate or grasp flexible materials with little or no sensing, and with simplified motions. The primary complexity lies in designing the fixture to correctly accomplish its task.

Traditionally, fixtures are used to immobilize a part in order to perform some tasks upon it, such as precisely drilling holes. Modular fixturing systems exist that can be reconfigured for different types of parts. Brost and Goldberg developed an algorithm to enumerate all possible fixture designs for an arbitrary polygon silhouette, and to select the optimal design based on some quality metric [10]. The modular fixturing system they work with has a precise lattice of holes as its base. Brost and Goldberg used three round locators as well as one translating clamp to create fixtures.

There has been significant work on reducing the time needed to create fixtures, with a number of computer-aided fixture design systems in existence. For example, Boyle *et al.* designed a system called CAFixD, which uses case-based reasoning to produce a fixture design for an arbitrary 3D part [8].

Lu and Akella developed a method of folding cardboard cartons using fixtures as manipulators [47, 48]. They view the unfolded carton blank as a robot with a series of revolute joints connecting the segments, and develop a motion planning algorithm that generates folding sequences for such robots, provided that there are no kinematic loops. From these sequences, it is possible to generate fixtures that fold the carton blank into a box.

This is also loosely related to work on designing tracks for vibratory bowl feeders. In such devices, spiral tracks are constructed with obstacles that are designed to allow parts to be fed along the track only if they are in one particular configuration. Invalid configurations are rejected back into the bowl, and start up the track again. One example of research in this area is work by Barretty *et al.* on automatically designing traps (polygonal holes cut into the track) that remove undesirable configurations [6].

1.1.9 String Models

Linear deformable objects (LDOs) and string have been analyzed extensively, and many different models have been developed. Pai used Cosserat rods to simulate thin strands, such as sutures [57]. The Cosserat rod model correctly captures behaviors associated with twisting, which are not accounted for in finite element method or ball and spring models. They develop a fast simulator using this model, which is capable of supporting real-time interaction with a virtual suture.

Wakamatsu and Hirai developed a detailed model of linear object deformation based on differential geometry [83]. This model correctly describes flexure, torsion, and extension. They apply their model to path planning with an LDO. Wakamatsu *et al.* also developed a 2D model of LDOs

in contact with obstacles [85]. The basic model is equivalent to their full 3D model, and also uses differential geometry. They extend their model to support dynamic deformation as a result of external forces and moments, as well as geometrical constraints. They also model contact of an LDO with a circular obstacle.

1.1.10 Knot Tying

Reidemeister moves are a major component of knot theory, which we use in developing some of our work [63]. Reidemeister moves are a set of three motions that change a diagram of a knot without changing the topology of a knot. In addition, these three motions suffice to change any diagram of an unknot into a trivial unknot (a circle).

While our robot is not the first to tie knots autonomously, our system is considerably less complex than others. Knot tying was first explored by Inoue and Inaba using a 6+1 DOF robot arm with stereo machine vision [36]. With this system, they successfully insert a rope into a ring, and then tie the rope into a knot around the ring.

Hopcroft *et al.* developed a graph-based language meant for programming knot tying motions at a fairly high level [33]. Since rope motions are unpredictable, it does not make sense to operate on fixed positions, since these will not be known in advance. Instead, the language consists of motions of points on segments along paths, which are located with a vision system. A higher level consisting of segment and crossing operations is built on top of the point and path-based level. Hopcroft *et al.* developed a parser and compiler for this language, and tested it by tying knots with a robot arm.

Phillips, Ladd and Kavraki created a simulator capable of handling realistic rope, with suturing explored as a possible application [59]. Their rope model consists of a spline formed from linear springs, with control points located at the ends of the springs. They employ adaptive subdivision as needed to refine the model. Their simulator includes support for collision detection, which is required for simulated knot tying.

Takamatsu *et al.* created a system that could learn to tie a knot by observing a human tying the same knot [79]. They analyze images of multiple steps of the knot tying process, and recognize the state of the knot in each image. From this sequence of states, they generate a sequence of knot tying motions based on Reidemeister moves. However, they do not seem to have extended this work to actually having a robot tie a knot.

Taylor gives a survey of medical robots in general, with some mention of suturing systems [81]. One such suturing system is the EndoBot, developed by Kang and Wen [41]. EndoBot is a system designed to assist surgeons in minimally invasive surgeries. Their system includes algorithms for autonomously tying knots while suturing, and they have modified a shuttle needle device for use by the robot.

Saha, Isto, and Latombe developed a string model and motion planning algorithms for tying simulated knots [68, 70, 69]. They model LDOs as a cylinder described by a smooth curve, with a coordinate frame at each point along the curve for describing torsion. They analyze a knot, and come up with a sequence of motions for tying it using two robot arms. They have implemented the motions both in simulation and using actual robot arms.

Wakamatsu, Arai and Hirai developed a very detailed description of the theory involved in tying and untying knots with robots [82]. They created a description of the topological state of a knot, and defined a set of four basic operations for transitioning between states. With these operations, they used a tree search to find sequences of motions from an initial state to a goal state. They developed a planner that determines grasp points and uses the state transitions to guide a robot arm in knot tying. They also showed that a SCARA robot (three translational DOF and one

rotational DOF) is sufficient to tie an LDO into a knot.

In addition, there are many patents for various devices to assist in knot tying. The patents all involve complex devices with moving parts, and are used for many applications, from tying fishing line [14] and shoelaces [12], to suturing [77].

1.1.11 Snake Robots and Needle Steering

Snake robots are another area loosely related to string manipulation, since they are in some sense also LDOs with the addition of some amount of actuation.

Burdick *et al.* analyzed the kinematics of a snake modeled as a continuous flexible curve [11]. They developed locomotion algorithms to enable the snake to travel in a uniform direction using sidewinding motions, as well as enabling it to change direction.

Wright *et al.* developed a modular snake robot that is capable of motion in a variety of environments, such as the inside and outside of pipes [88]. They develop a custom servo motor for control, and covered their modules in high-friction skin to assist in locomotion.

Webster *et al.* designed a system that uses nonholonomic controls to steer a flexible, bevel-tipped needle through soft tissue [35]. They observed that the bevel tip of the needle makes it possible to steer the needle by controlling insertion and axial rotation, and developed a model using standard nonholonomic unicycle and bicycle models. They created a planner, and experimentally verified the ability to control the needle.

1.1.12 Mold Parting Directions

In order to build multi-piece knot fixtures, we need to know how to separate the fixture pieces to free the entire knot. This is very similar to the problem of extracting cast or injection-molded parts from their molds. Typically, only 2-moldability is considered, in which there are two mold pieces that are separated with one translation. Early approaches only considered the three principal axes as parting directions; however, they allowed complex (non-planar) parting surfaces [62, 16]. Later approaches allowed for multiple mold pieces. Chen gave criteria for identifying parting faces, which are used with a reverse glue operation to produce a set of mold pieces [17]. Finally, Khardekar *et al.* developed an algorithm running on graphics hardware for computing a feasible parting direction for two mold pieces [42]. On a modern GPU, this is fast enough to allow for real-time highlighting of undercuts (regions that would get stuck during mold separation).

Chapter 2

Cloth Grasping[1]

Cloth manipulation is difficult as a result of the flexibility of cloth. When cloth is suspended from one or two points, it develops buckles in a manner that is hard to predict. Grasps that minimize buckling will therefore make it easier to handle a piece of cloth, such as during the flattening or folding of laundry. One guaranteed way to avoid buckling is to fully immobilize cloth in a flattened state.

Once we have cloth immobilized in a flat state, we have full configuration information with which we can plan further actions. Further, once we have a set of points necessary for an immobilizing grasp, we only need to sense these points to get information about the cloth. If all of the grasp points are in locations corresponding to a flat configuration, we know that the rest of the cloth must be flat without directly sensing it.

We make a few simple assumptions about the cloth grasping problem. The cloth is non-stretchable, and we will place some number of point fingers on the cloth. These fingers are "pinned" to the plane; once they are placed, they do not move and they directly immobilize the point on the cloth underneath them.

The fundamental questions in grasping ask how many fingers are needed for a grasp, and where they should be placed. Figure 2.1 shows three pieces of cloth, all of which are immobilized except for b).

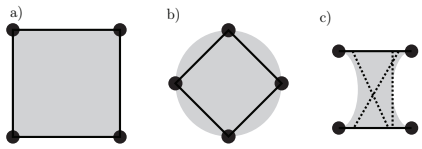


Figure 2.1: Three flat cloth shapes grasped by fingers. All but b) are immobilized.

Fact 2.1. Any line segment with pinned endpoints that is fully contained in a polygon (the endpoints are mutually visible) is immobilized.

First order line segments of this type are indicated by solid lines in Figure 2.1. If a point somewhere in the polygon lies on a line segment between grasp points or first order lines, then it too will be immobilized, since the endpoints of this second order line are immobilized. A few second order lines are shown as dashed lines in the figure. This process can be repeated as needed with higher order line segments until the entire cloth is immobilized.

[1]Portions of this chapter were published as [5]

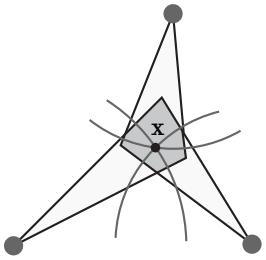
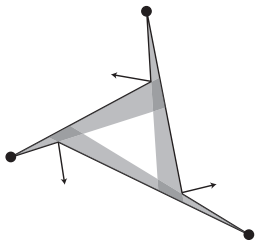
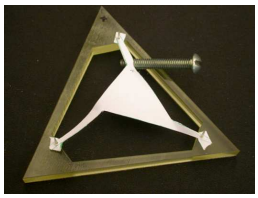


Figure 2.2: Star grasped with three fingers.



(a) Arrows indicate possible instantaneous velocities.



(b) Paper polygon after motion was applied.

Figure 2.3: Polygon that cannot be immobilized by pinning convex vertices (closed circles).

There are some cases where we cannot validate a grasp by drawing immobilized lines between fingers. Consider the polygon shown in Figure 2.2. No finger is visible from another finger. However, no point in the shaded hexagon can move further from any of the fingers, so this region is immobilized; therefore the entire polygon is immobilized.

To immobilize a cloth polygon, there must be a finger at least at each convex vertex; otherwise, that convex vertex will be free to move. In some cases, pinning just the convex vertices is enough. However, the piece of cloth shown in Figure 2.3 cannot be immobilized by pinning the three convex vertices of the shape. We have verified this result experimentally (Figure 2.3(b)) and theoretically (Section 2.3.1).

This polygon is representative of a class of polygons which we call pinwheels. These polygons all require more than n_{convex} fingers for immobilization. In Theorem 2.6, we will show that the upper bound is $n_{\text{convex}} + \left\lfloor \frac{1}{3} n_{\text{concave}} \right\rfloor$ fingers for simple polygons. This bound is tight; there exist polygons that require this many fingers for immobilization.

2.1 Cloth Models and Definitions

Cloth can be modeled in several different ways. In the graphics and simulation worlds, ball and spring models are quite common. However, for our approach, we want the cloth to not stretch, which suggests a developable surface model.

We will use a model that is "almost" a developable surface model. We assume the cloth cannot stretch, but that the cloth may compress slightly. Our upper bound on the maximum number of fingers needed to grasp cloth ($n_{\text{convex}} + \left\lfloor \frac{1}{3} n_{\text{concave}} \right\rfloor$) holds for developable surfaces, but we discuss the existence of polygons requiring this many fingers only for the compressible model.

2.1.1 Support Graphs

To discuss polygon immobilization, we use a specific type of polygon skeleton called a support graph; an example is shown with dotted lines in Figure 2.4.

Definition 2.1. A **support graph** for a polygon is an embedded planar graph contained within the polygon, such that every point of the polygon falls on a line segment (possibly of length zero) that

- is completely contained within the polygon, and
- has endpoints that are points of the embedded graph (on an edge or at a node).

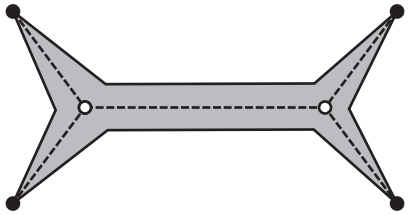


Figure 2.4: Example of a support graph in a cloth polygon.

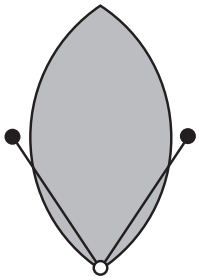


Figure 2.5: Allowed motion of a non-positively-spanned vertex.

A **support tree** is a support graph with no cycles.

It is clear that if a support graph for a polygon is immobilized by some set of fingers, every line segment specified in the definition is immobilized, and therefore the polygon is immobilized. We can examine the immobilization of support graphs by placing fingers at vertices.

Definition 2.2. A **pinned vertex** is a graph or polygon vertex that is held in place by a finger. This is indicated in diagrams by a closed circle. (Unpinned vertices have open circles).

Definition 2.3. A **positively-spanned vertex** is a vertex in a graph whose adjacent edges positively span $\mathbb{R}^2$. (For a definition of positive linear spans, see [22].)

There are many ways to construct a support graph for a polygon. Figure 2.4 shows a support graph constructed by hand, but we can always easily construct a (possibly more complex) support graph by triangulating a polygon. If a triangulation of the polygon is immobilized, the polygon is immobilized.

We assume a model of cloth that allows the cloth to compress. In this case, if a triangulation of the cloth is not immobilized by a set of fingers, the cloth is not immobilized.

2.2 Immobilizing Trees, Graphs, and Polygons

As an approach to specifying the fingers required to grasp a piece of cloth, we can first describe the fingers needed to immobilize a connected, linear network of non-stretchable string embedded in the plane. If this network is a support graph for a polygon, then that polygon is immobilized in 2D and 3D.

At a minimum, all vertices that are not positively-spanned must be pinned in order to immobilize a non-stretchable planar graph. The shaded region in Figure 2.5 illustrates the free motions of an unpinned and non-positively-spanned vertex.

2.2.1 Immobilizing Non-stretchable Graphs

Initially, we consider a non-stretchable tree, and assume that all vertices have degree one or three. Additionally, we will assume that all interior vertices (non-leaves) are positively-spanned vertices. These assumptions will be relaxed later, but they are useful in the first stage of the proof.

In the theorems that follow, we consider only first order constraints on the free motions of vertices, since linear constraints are sufficient for the proofs and simpler to analyze. Using quadratic distance constraints yields the same results. We will use the notation $\overrightarrow{uv}$ to indicate a normalized vector pointing from vertex u to vertex v. Figure 2.6 illustrates the following lemma.

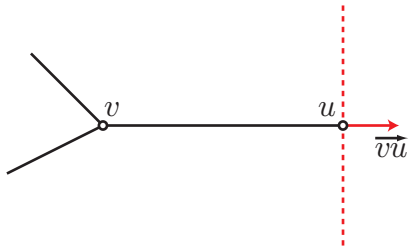


Figure 2.6: Restriction on allowed motions of u.

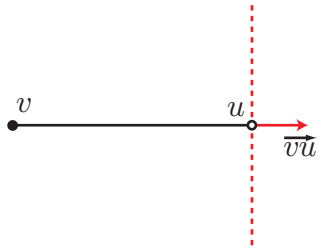


Figure 2.7: Base case (vertex v is pinned, as indicated by the closed circle).

Lemma 2.1. *Consider a planar non-stretchable tree, with all vertices of degree one or three, and with only positively-spanned interior vertices. Let all the leaves (non-positively-spanned vertices) be pinned, except for one leaf, labeled u. Let v be the vertex adjacent to u. Vertex u cannot move into the half plane defined by normal $\overrightarrow{vu}$. (This can also be written as a constraint of the form $\dot{u} \cdot \overrightarrow{vu} \leq 0$.)*

Proof. **Induction Hypothesis**: Consider a tree subject to the assumptions with all leaves pinned except for u, and let v be the vertex adjacent to u. u cannot move into the half plane indicated by the constraint $\dot{u} \cdot \overrightarrow{vu} \leq 0$.

Base Case: The base case is a tree consisting of only vertices v and u (Figure 2.7), with vertex v pinned. The proof is clear for this case.

Inductive step: Given a tree T, break it at vertex v into two trees, T_1 and T_2. Let a be the vertex adjacent to v in T_1, and b be the vertex adjacent to v in T_2 (Figure 2.8). By the induction hypothesis, T_1 imposes the constraint $\dot{v} \cdot \overrightarrow{av} \leq 0$ (equivalent to $\dot{v} \cdot \overrightarrow{va} \geq 0$), and T_2 imposes the constraint $\dot{v} \cdot \overrightarrow{bv} \leq 0$ (equivalent to $\dot{v} \cdot \overrightarrow{vb} \geq 0$).

From our assumptions, we know that $\overrightarrow{va}$, $\overrightarrow{vb}$, and $\overrightarrow{vu}$ positively span $\mathbb{R}^2$. As a result, if both $\dot{v} \cdot \overrightarrow{va} \geq 0$ and $\dot{v} \cdot \overrightarrow{vb} \geq 0$ are satisfied, then $\dot{v} \cdot \overrightarrow{vu} \leq 0$, proving the induction hypothesis. □

This lemma can be extended from restricted motion to immobilization.

Lemma 2.2. *Consider a planar non-stretchable tree, with all vertices of degree one or three, which contains only positively-spanned vertices in its interior. If all the leaves of this tree are pinned, the tree will be immobilized.*

Proof. Consider a tree that satisfies Lemma 2.1, and label its unpinned leaf u. Leaf u cannot move away from its adjacent vertex v ($\dot{u} \cdot \overrightarrow{vu} \leq 0$, which also implies $\dot{v} \cdot \overrightarrow{vu} \leq 0$). If we now pin u, we impose a constraint on v of $\dot{v} \cdot \overrightarrow{uv} \leq 0$. Combined with the previous constraints at v from the other adjacent edges (which we know positively span $\mathbb{R}^2$ if edge vu is included), this completely

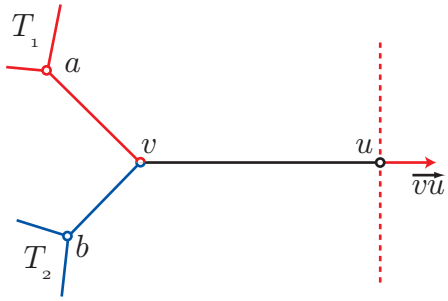


Figure 2.8: Inductive step.

immobilizes v. The immobilization of v can now be used to show that the vertices adjacent to v are also immobilized. This immobilization can be continued throughout the tree, showing that the entire tree is immobilized. □

This result can be strengthened to any non-stretchable planar tree. The next theorems depend on the concept of splitting vertices of a non-stretchable tree or graph by pinning them. If a vertex v has k adjacent edges, and we pin v, then this is equivalent to having k pinned vertices all located at the same point as v, with each vertex adjacent to exactly one of the edges adjacent to v. Physically, the resulting tree or graph is exactly equivalent to the original tree or graph, as constraints do not propagate past pinned vertices.

Theorem 2.3. *Any planar non-stretchable tree embedded in $\mathbb{R}^2$ (with vertices of any degree) that has its non-positively-spanned vertices pinned is immobilized.*

Proof. First, we will remove the assumption that all interior vertices must be spanned vertices, and we allow degree 2 vertices. If any vertex is non-positively-spanned, then it is pinned, as is specified by the theorem statement. Additionally, note that any degree 2 vertices can never have edges positively spanning $\mathbb{R}^2$, and therefore must be pinned. If we break the tree into a forest by splitting it at each non-positively-spanned (and pinned) interior vertex, each component of the forest will be immobilized by Lemma 2.2. When joined, the resulting complete tree is still immobilized.

Finally, we allow vertices of degree greater than 3. If such a vertex is non-positively-spanned, we can simply use the argument above. If it is positively-spanned, then we need to slightly rework the inductive step of Lemma 2.1. If vertex v is of degree $d > 3$, it will be split into $d - 1$ subtrees (along all edges except vu). By the inductive hypothesis, we know there are constraints of the form $\dot{v} \cdot \overrightarrow{va_i} \geq 0$ for each subtree T_i. We can pick a pair of subtrees T_i and T_j, such that $\overrightarrow{va_i}$, $\overrightarrow{va_j}$, and $\overrightarrow{vu}$ positively span $\mathbb{R}^2$. Now, as in the original inductive step, this gives us the desired constraint on u. □

If we split a graph into a tree by adding one finger per cycle (and pinning non-positively-spanned vertices), the graph is immobilized.

Theorem 2.4. *Any planar non-stretchable graph with all non-positively-spanned vertices pinned and at least one vertex pinned within each cycle is immobilized.*

Proof. Pin one vertex per cycle of the graph. This splits the graph at all of these pinned vertices. Splitting each cycle with one finger converts the graph into a tree, with properties satisfying Theorem 2.3. □

If no vertices in a cycle are pinned, the vertices in the cycle can typically move. One case in which the cycle is immobilized occurs if the edges supporting the cycle bisect the exterior angles of the cycle, but this case is rare.

2.2.2 Grasping Polygons

A tree or graph embedded in a cloth polygon can be used to show that the polygon is immobilized.

Theorem 2.5. *If a cloth polygon contains a planar non-stretchable graph G such that non-positively-spanned vertices of the graph correspond exactly to the convex vertices of the polygon, then the graph is a support graph for the polygon, and immobilizing the graph immobilizes the polygon.*

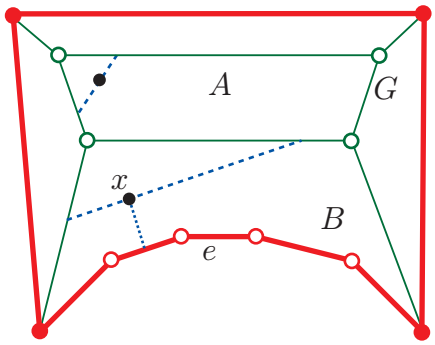


Figure 2.9: Two types of cells (A and B) in a polygon containing a support graph.

Proof. By the definition of a support graph (Definition 2.1), every point in the polygon must lie on a line with endpoints on the support graph.

Consider the polygon and graph shown in Figure 2.9. Since non-positively-spanned vertices of the graph (thin line) exactly map to all convex vertices, the polygon (thick line) is divided up into two types of cells. Cells that are contained within cycles of the graph are trivial to handle (indicated by A in the figure). For any point within a cycle, any line through the point has endpoints on the graph, and thus is immobilized if the graph is immobilized.

The other type of cell is enclosed by graph edges and a chain of (possibly zero) concave vertices on the polygon boundary (B in the figure). The polygon boundary must consist purely of concave vertices, as a convex vertex would have a non-positively-spanned graph vertex located at it, splitting the cell. Now, consider any point x in the cell. Find the closest polygon edge e. Extend a line through x parallel to e until both ends of the line hit the boundary of the cell. The endpoints must both lie on graph edges; if this were not the case, the polygon boundary would contain a convex vertex, and it does not. Therefore, for any point in this type of cell, there exists a line with both endpoints on the graph.

Since both types of cells satisfy the definition of a support graph, G is a support graph. By definition of a support graph, if G is immobilized, the polygon is immobilized.

□

We can now show that $n_{\text{convex}} + \lfloor \frac{1}{3} n_{\text{concave}} \rfloor$ fingers are always sufficient to immobilize a polygon. In the following proof, we view a triangulation of a polygon as a graph embedded in the polygon.

Theorem 2.6. *A simple cloth polygon can always be immobilized by pinning* $n_{\text{convex}} + \lfloor \frac{1}{3} n_{\text{concave}} \rfloor$ *vertices.*

Proof. Portions of this proof are similar to Fisk's proof that an art gallery requires $\lfloor \frac{n}{3} \rfloor$ guards [26]. In both proofs, the main problem is placing one item (a guard or a pinned vertex) per triangle.

As in Fisk's proof, we begin by considering a triangulation $T = (V, E)$ of the polygon P. We consider the most strict form of a triangulation, in which triangle vertices must also be polygon vertices. In this type of triangulation, concave polygons vertices will be positively-spanned by incident graph edges, and convex vertices will not be. Concave vertices must be positively-spanned because each exterior angle at a concave vertex is less than $\frac{\pi}{2}$, and the interior angle is split into angles of less than $\frac{\pi}{2}$ by the triangulation.

Let all convex vertices of the polygon (and thus all non-positively-spanned vertices of T) be pinned. By Theorem 2.4, T is immobilized if we also pin one vertex per cycle (which, for a triangulation, means one pinned vertex per triangle).

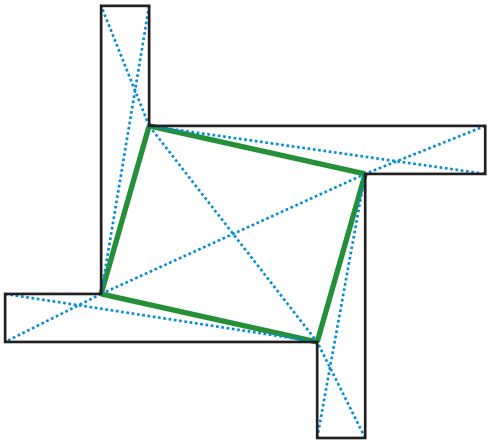


Figure 2.10: Convex quadrilateralization of an orthogonal polygon, with extra edges (dashed lines) necessary for 4-colorability.

Convex vertices must always be pinned, so we can ignore any edges that are adjacent to them, and we can construct a $T^* = (V^*, E^*)$ that removes these edges. Specifically, T^* contains only the concave vertices of P, and only edges that are between pairs of concave vertices. Since any triangulation can be 3-colored [55], and since T^* is a subset of a (3-colorable) triangulation T, T^* is also 3-colorable. As in Fisk's proof, one of the three colors must be used no more than $\lfloor \frac{1}{3}|V^*| \rfloor = \lfloor \frac{1}{3} n_{\text{concave}} \rfloor$ times. Now, pin each vertex labeled with the least frequently used color. Since each triangle must have one vertex of each color, each triangle (and therefore cycle) of T^* has one pinned vertex, and therefore each cycle of T has one pinned vertex. As a result, T (and thus P) is immobilized. □

The above proof does not hold for non-simple polygons, as triangulations of such polygons are not necessarily 3-colorable. However, we can use the same idea to give a bound for polygons with holes as well.

Corollary 2.7. *A cloth polygon with n_{holes} holes can always be immobilized by pinning $n_{\text{convex}} + \lfloor \frac{1}{3} n_{\text{concave}} \rfloor + 2n_{\text{holes}}$ vertices, where both n_{concave} and n_{convex} include the concave and convex vertices in the polygon's holes.*

Proof. If we place cuts in the polygon such that each hole is open to the region outside the polygon (either directly through a single cut, or by a chain of cuts through other holes), then we have turned the polygon with holes into a simple polygon with at most $4n_{\text{holes}}$ new vertices (careful cutting can reduce this to $2n_{\text{holes}}$ new vertices if the cuts go between existing vertices). These new vertices will be convex vertices; however, since the two sides of the cut are in the same place, we can use one finger to pin each pair of new convex vertices. As in Theorem 2.6, we now triangulate the simple polygon, which requires us to pin up to $\lfloor \frac{1}{3} n_{\text{concave}} \rfloor$ concave vertices, plus the original convex vertices, plus two fingers per cut (equivalent to two fingers per hole). Therefore, a polygon with holes will be immobilized with $n_{\text{convex}} + \lfloor \frac{1}{3} n_{\text{concave}} \rfloor + 2n_{\text{holes}}$ vertices. □

We can also refine this proof for orthogonal polygons. Figure 2.10 illustrates the following theorem.

Theorem 2.8. *A simple orthogonal cloth polygon can always be immobilized by pinning $n_{\text{convex}} + \lfloor \frac{1}{4} n_{\text{concave}} \rfloor$ vertices.*

Proof. As with the proof for general simple polygons, this proof is similar to the proof that an orthogonal art gallery requires $\lfloor \frac{n}{4} \rfloor$ guards, given by Kahn [39].

We begin by taking a convex quadrilateralization Q of the orthogonal polygon P. A convex quadrilateralization subdivides the polygon into convex quadrilaterals by placing non-intersecting diagonals between polygon vertices; the existence of such a decomposition for orthogonal polygons is proved by Kahn. Additionally, in order to force the quadrilaterals to be 4-colorable, we add a pair of diagonals to each quadrilateral, and consider the result as a graph. Figure 2.10 illustrates this structure; the thin black lines indicate the polygon, the medium green lines indicate the quadrilateralization, and the thin, dotted lines show the extra diagonals added to each quadrilateral. Concave polygon vertices become positively-spanned vertices of the graph as a result of the requirement that the quadrilaterals be convex.

The remainder of the proof is essentially identical to Theorem 2.6. We pin all convex vertices, 4-color the graph, and discard edges adjacent to pinned vertices to produce Q^*. In the resulting colored graph, one of the four colors is used no more than $\lfloor \frac{1}{4} n_{\text{concave}} \rfloor$ times. If we pin the vertices corresponding to this color, Q^* has one pinned vertex per cycle (or per quadrilateral), and thus Q has one pinned vertex per cycle, indicating that Q and P are immobilized. □

Although some of the proof techniques and bounds are related to similar techniques and bounds for the art gallery problem, there are polygons for which a solution to the art gallery problem is not also a solution to the immobilization problem, and vice versa. For example, in Figure 2.10, after culling edges adjacent to convex vertices, we are left with the central quadrilateral. A guard in the middle of this quadrilateral (at the intersection of the middle diagonals) would be enough to watch the entire quadrilateral, but a finger placed at this vertex does not immobilize the polygon when combined with the additional fingers gripping the convex vertices. With this grasp, the polygon would still be able to rotate about the central finger. For the reverse statement, an immobilization of the shape in Figure 2.4 is not a solution to the art gallery problem.

An important distinction is that for cloth, we are operating only on the concave vertices, whereas for the art gallery, we are using all polygon vertices. The similarity between the problems comes up when we remove edges adjacent to convex vertices, and are left with a smaller polygon (or potentially a set of polygons). Within this smaller polygon, we are trying to place one finger on the edges of each subdivision (triangle or quadrilateral), which is the same as the goal for an art gallery.

2.3 When Convex Vertices Are Not Enough

We have shown that $n_{\text{convex}} + \lfloor \frac{1}{3} n_{\text{concave}} \rfloor$ fingers are always sufficient to immobilize a simple polygon, but in order to show that this bound is also necessary, we must first show that there are polygons for which a convex vertex grasp is insufficient for immobilization.

2.3.1 Determining Free Motions

If we can compute possible free motions of a grasped cloth polygon, then the grasp is clearly insufficient. We can verify a grasp by constructing an appropriate linear program, and by testing to see if it has any nonzero solutions. This LP is built from distance constraints, which require that the endpoints of an edge cannot move apart beyond their initial stretched distance. We use the standard notion of polygon visibility in this section.

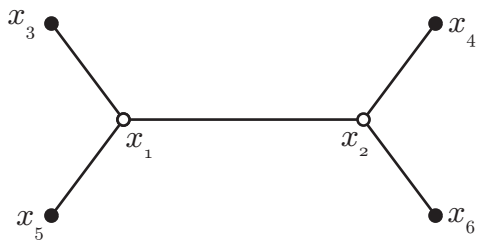


Figure 2.11: A network of points connected by strings (closed circles are pinned).

If x_i and x_j are mutually visible, then at every time t, the distance between the points must not be greater than the initial (fully stretched) distance:

$$||\overrightarrow{x_i x_j}(t)||^2 \leq ||\overrightarrow{x_i x_j}(0)||^2. \tag{2.1}$$

At time 0, the time derivative of every distance between pairs of mutually visible points must be non-positive.

$$\dot{x}_i \cdot \overrightarrow{x_j x_i} + \dot{x}_j \cdot \overrightarrow{x_i x_j} \leq 0 \tag{2.2}$$

A simple example is a network of points attached by strings as shown in Figure 2.11. Let x_1 and x_2 be unpinned points, and let x_3 through x_6 be pinned. There are five distance constraints, corresponding to the edges. Using the constraints from equation 2.2, we have

$$\begin{bmatrix} \overrightarrow{x_3 x_1} & 0 \\ \overrightarrow{x_5 x_1} & 0 \\ 0 & \overrightarrow{x_4 x_2} \\ 0 & \overrightarrow{x_6 x_2} \\ \overrightarrow{x_2 x_1} & \overrightarrow{x_1 x_2} \end{bmatrix} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} \leq 0. \tag{2.3}$$

We can rewrite this as

$$J\dot{x} \leq 0 \tag{2.4}$$

This is in the form of constraints for a linear program, and therefore we can use a solver to see if there are any solutions other than $\dot{x} = 0$. If such solutions exist, then the line network can move as described by one of these solutions.

Theorem 2.9. *If the only solution to $J\dot{x} \leq 0$ is $\dot{x} = 0$ for a triangulation of a polygon, then the corresponding cloth polygon is immobilized.*

Proof. Consider any point p on the cloth, and let this point be contained in the triangle T_p. Any motion of p will pull on at least one of the edges e_p of T_p. Since we know that the triangulation network is immobilized, it is not possible for e_p to move, and therefore p cannot move. □

This gives us an algorithm for verifying a grasp for a cloth polygon. To do this, we take any triangulation of the polygon, and consider this as our line network. We then build J, which has one row for every edge of the triangulation (with the exception of any edges between pinned points, since the coefficients would all be zero in this case). If $J\dot{x} \leq 0$ only has the solution $\dot{x} = 0$, then the triangulation network is immobilized by the given grasp. We have implemented this algorithm in Matlab, using CGAL [13] to construct triangulations and lp_solve to check for nonzero solutions

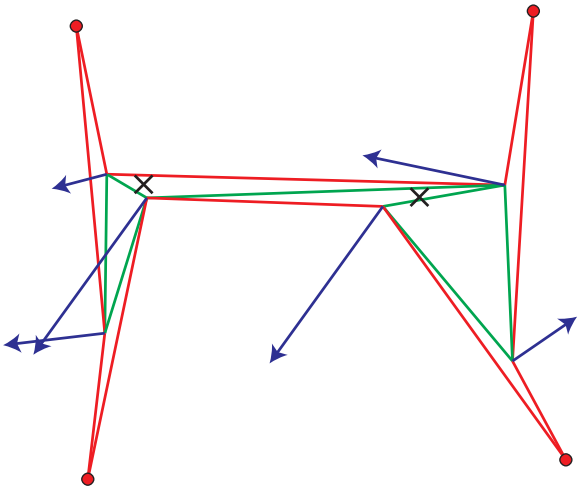


Figure 2.12: A dual pinwheel, with free motions as shown. Adding fingers at the X's immobilizes the polygon.

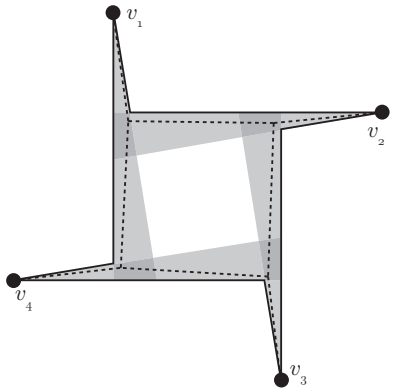


Figure 2.13: A 4-pinwheel, with its cyclic support graph and first-order visibility polygons.

given the constraints. An example run of this algorithm for a non-immobilized polygon is shown in Figure 2.12, with X's indicating one possible set of additional fingers that immobilize the polygon.

If nonzero solutions exist for the lines of a triangulation, we believe that this means that the cloth can move within the given grasp. However, this statement may depend on the cloth model that we use. If we assume that the cloth can simply compress into itself, then it is clear that a nonzero solution will allow movement of the cloth. It is less clear as to what happens if a more realistic model that involves buckling is used, or if the cloth is a developable surface.

2.3.2 Pinwheels

As shown with the example in Figure 2.12, there are polygons for which a convex vertex grasp is insufficient. All such polygons that we have found fall into a class that we refer to as pinwheels.

Definition 2.4. An ***n*-pinwheel** is a polygon with a cyclic first order visibility structure, where a first order visibility structure is defined as the set of visibility polygons from all of the convex vertices of the polygon. The number n refers to the number of points in the pinwheel.

In an n-pinwheel, the visibility polygon from a vertex v_2 first intersects its clockwise neighbor's (v_1) visibility polygon, followed by its counter-clockwise neighbor's (v_3) visibility polygon (See

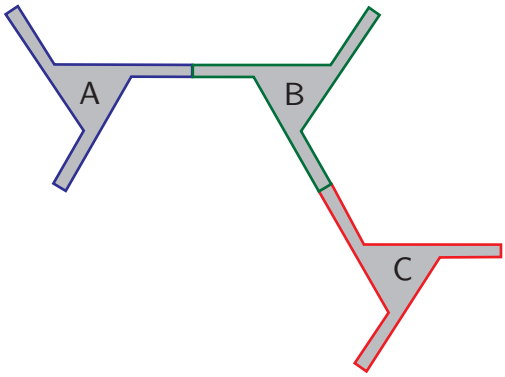


Figure 2.14: Multiple pinwheels.

Figure 2.13 for an example of a 4-pinwheel). The directions can be reversed; if a vertex first sees its counter-clockwise neighbor's visibility polygon, followed by that of its clockwise neighbor, then the polygon also has a pinwheel structure. In order to actually be a pinwheel, this type of visibility intersection must be repeated for all vertices, leading to a cycle of visibility intersections.

Theorem 2.10. *A non-stretchable cloth n-pinwheel can always be immobilized with* $n_{\text{convex}} + 1$ *fingers.*

Proof. A support graph with one cycle and n_{convex} non-positively-spanned vertices located at the convex vertices of the pinwheel can be constructed from the cyclic visibility intersections present in a pinwheel (Figure 2.13). We already know that all n_{convex} convex vertices must be pinned. By Theorem 2.4, pinning any one vertex of the cycle immobilizes the graph, and therefore, pinning the corresponding point in the pinwheel immobilizes the pinwheel. □

We will use pinwheels to show that our upper bound on the number of fingers needed for immobilization is a tight bound.

Theorem 2.11. *There exist non-stretchable cloth polygons that require a grasp of* $n_{\text{convex}} + \lfloor \frac{1}{3} n_{\text{concave}} \rfloor$ *fingers to be immobilized.*

Proof. The class of polygons that we will use to satisfy the statement is based on 3-pinwheels. Consider the triple 3-pinwheel shown in Figure 2.14. The points have been expanded to two vertices to simplify the edge that is common to pairs of 3-pinwheels. As discussed in Section 2.3.1, we can build a linear program that gives the possible motions of a 3-pinwheel. From this, we can easily show that only pinning the six convex vertices does not suffice to immobilize one of the modified 3-pinwheels by itself. It is possible to immobilize a single pinwheel by adding one additional finger.

Now, consider attaching pinwheel B to pinwheel A, with all convex vertices pinned. Let us assume that the dual A-B pinwheel can be immobilized with just one additional finger. If this finger is on the boundary between A and B, then neither pinwheel will be immobilized, as this single finger will provide no more support than would have existed had we pinned the convex vertices of each pinwheel. Next, assume that we have placed the extra finger in such a way that all of A is immobilized (note that this is not actually possible). If this is the case, the boundary line between A and B will also be immobilized. However, as we have already stated, this is not enough to immobilize B. The same situation exists in reverse if we put a finger in B that immobilizes B.

Finally, we can extend this chain by adding pinwheel C, followed by another pinwheel attached to C's right point, and so on (Figure 2.15). There must be one finger per pinwheel to be able to immobilize the entire shape, as fingers outside the boundaries of a pinwheel do not suffice to

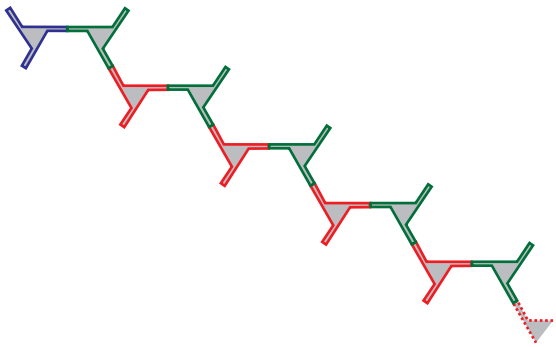
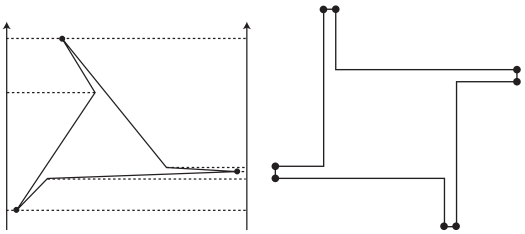


Figure 2.15: Repeating chain of pinwheels.



(a) Monotone polygon. (b) Orthogonal polygon.

Figure 2.16: Monotone and orthogonal polygons that cannot be immobilized by a convex vertex grasp.

immobilize it. Since each pinwheel has 3 concave vertices, this means that the overall shape requires $n_{\text{convex}} + \lfloor \frac{1}{3} n_{\text{concave}} \rfloor$ fingers. □

Theorem 2.12. *There exist non-stretchable orthogonal cloth polygons that require a grasp of* $n_{\text{convex}} + \lfloor \frac{1}{4} n_{\text{concave}} \rfloor$ *fingers to be immobilized.*

Proof. This theorem is identical to Theorem 2.11, except that we chain together the orthogonal 4-pinwheel in Figure 2.16(b). □

We are able to make statements about several classes of polygons. It is possible to place a support tree with non-positively-spanned vertices only at convex vertices in all star-shaped and convex polygons; such polygons are thus immobilized by a convex vertex grasp. Pinwheels do not fall into either of these classes. Interestingly, we can construct monotone (Figure 2.16(a)) and orthogonal (Figure 2.16(b)) pinwheels.

We have now shown that for a simple non-stretchable cloth polygon, the minimum number of fingers needed to immobilize it is $n_{\min} \in [n_{\text{convex}}, n_{\text{convex}} + \lfloor \frac{1}{3} n_{\text{concave}} \rfloor]$.

2.4 Grasping with Fewer Fingers

Algorithm 1 (Grasp-Build) gives a simple algorithm for generating grasps. It begins by testing a convex vertex grasp using our linear program formulation from Section 2.3.1. If this fails, the triangulation method is used to get a grasp that pins one third of the concave vertices.

Disregarding the LP step, this algorithm has a running time of $O(n)$. Chazelle showed that triangulation of a simple polygon requires $O(n)$ time [15], and the 3-coloring of a triangulation can be implemented with a simple linear time algorithm.

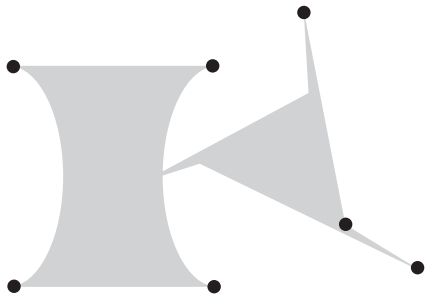
Grasp-Build is guaranteed to generate a valid grasp; however, the grasp may include unnecessary fingers if there are lengthy chains of concave vertices, as in Figure 2.17.

2.4.1 Grasp Reduction

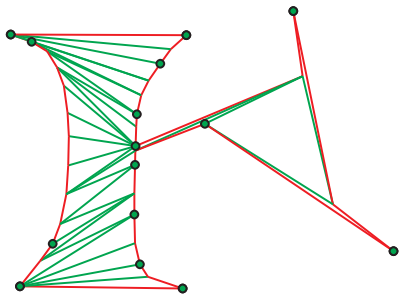
We have developed an algorithm to reduce the size of the grasp, which removes certain fingers by checking to see if they are already immobilized by other portions of the grasp. Consider the generic polygon shown in Figure 2.18, and assume that v_2 is pinned as a result of Grasp-Build. There exist lines starting from the neighboring vertices of v_2 (v_1 and v_3) that pass through v_2 and hit other edges of the polygon (e_1 and e_3, respectively). Arbitrarily pick one of these neighbors. The selected neighbor is either convex or concave. We will consider these cases independently.

Algorithm 1 GRASP-BUILD

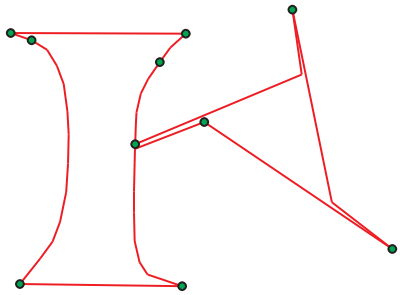
Require: P is a simple polygon
Ensure: Grasp is a valid grasp for P
  Grasp = V_{convex}
  if LPVerify(Grasp) **then**
    return Grasp
  end if
  T = Triangulate(P)
  T^* = T with V_{convex} and edges adjacent to V_{convex} removed
  Coloring = 3Color(T^*)
  MinColor = color applied to fewest vertices in Coloring
  Grasp = Grasp + (vertices colored with MinColor)
  return Grasp



(a) A valid (minimal) grasp (1 pinned concave vertex).



(b) Grasp built by GRASP-BUILD (9 pinned concave vertices).



(c) The result of GRASP-REDUCE (4 pinned concave vertices).

Figure 2.17: A polygon with $n_{concave} = 28$, $n_{convex} = 6$.

Case 1: Convex neighbor (v_1): Since initially the polygon was immobilized, e_1 has immobilized endpoints, and thus is a support line. Therefore $v_1v_2e_1$ (the blue dashed line) is also a support line that immobilizes v_2. In order to keep this second order support line valid, v_1 and e_1 must remain immobilized. Since v_1 is convex, it is guaranteed to remain immobilized. In addition, the endpoints of e_1 must remain immobilized, so we will simply require that they both remain pinned if they were pinned by the given grasp.

Case 2: Concave neighbor (v_3): If we extend v_2v_3 to some polygon edges (in the figure, edges e_3 and e_2, at the ends of the red dotted line), then we see that v_2 and v_3 lie on a support line if e_3 and e_2 are immobilized. Again, this requires that any pinned endpoints of e_2 and e_3 remain pinned.

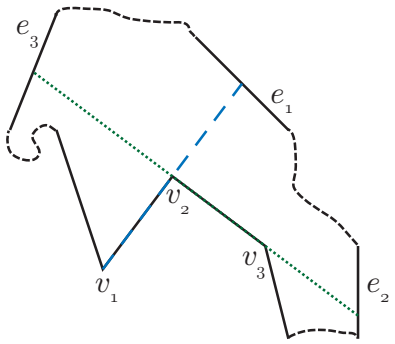


Figure 2.18: Method for reducing the number of pinned points.

In either case, we can unpin v_2, reducing the size of the grasp. Algorithm 2 (GRASP-REDUCE) is based on this method. This still does not guarantee a minimal grasp, but it will at least produce a reasonable approximation.

Algorithm 2 GRASP-REDUCE

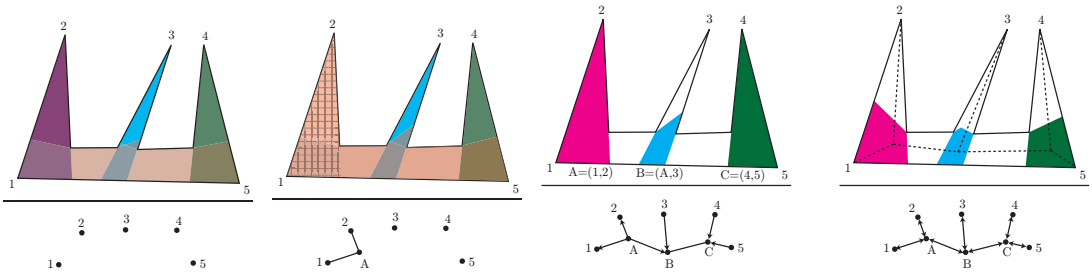
Require: Grasp is a valid grasp for a simple polygon P
Ensure: Grasp remains a valid grasp for P
Unpinnable = $\{v : v \in \text{Grasp} \wedge v \text{ is a concave vertex}\}$
for all $v \in$ Unpinnable **do**
-Construct lines l_1 and l_2 through v from v's neighbors.
-If either neighbor is concave, continue the line past the neighbor to a polygon edge.
-Pick the line that involves the fewest vertices in Unpinnable (a vertex is involved if it is an endpoint of an edge hit by the line).
-If an edge that is hit contains any vertices that were not in the original grasp, invalidate the line that hits it, as there is no guarantee that unpinning vertices will not allow these vertices to move.
-If any of the endpoints of edges hit by the chosen line are in Unpinnable, remove these vertices from Unpinnable, and keep them as part of the grasp.
-Remove v from Unpinnable and Grasp.
end for
return Grasp

GRASP-REDUCE has a running time of $O(n^2)$. The outer loop runs $O(n)$ times, and the various set operations take either $O(1)$ or $O(n)$ time. The key factor in the inner loop is determining which edges are hit by l_1 and l_2, which takes $O(n)$ time, as we must scan through all edges to find the closest one that intersects l_1 or l_2.

Figure 2.17 shows example results from our algorithms. Figure 2.17(a) gives a minimal grasp, which consists of 6 convex vertices, plus one concave vertex. Figure 2.17(b) shows the results of the grasp building algorithm, and Figure 2.17(c) shows the grasp after it has been reduced. A few extra vertices still remain; the algorithm could be improved by enabling it to recognize immobilized lines between immobilized edges, such as a vertical line through the hourglass portion of Figure 2.17(a).

2.4.2 Graphical Method for Analyzing Immobilization

We have developed a graphical method for determining if a cloth polygon is immobilized by a given grasp. This method relies on embedding a support tree within a polygon. A support tree



(a) Convex vertex visibility (b) Picking intersection (1,2) (hatched region) (c) Visibility tree (d) Polygon and tree after clipping

Figure 2.19: Steps in constructing a support tree, with the corresponding visibility tree at each step

is fundamentally based on visibility; in particular, adjacent vertices in the support tree must be mutually visible. Visibility is fairly easy to assess visually, and therefore manually placing a support tree in a polygon is a quick method for determining if a polygon is immobilized with a given grasp. We have taken this manual method and expanded it into an algorithm for constructing support trees. Our algorithm repeatedly intersects visibility regions to form a skeleton, and uses an optimizer to try to shift the vertices of the skeleton until the skeleton becomes a support tree.

In order for a tree to be valid, we require that the graph consisting of the tree plus polygon edges be planar. Also, if a pair of convex vertex visibility regions are intersected, the convex vertices must have been concave-adjacent, meaning that they were separated only by a chain of concave vertices (such as vertices 2 and 3 in Figure 2.19(a)).

The planarity requirement comes from the fact that we are dealing with cloth, and not string. In a line network, as in Section 2.2.1, it does not matter if two of the string edges cross, as there will be no interaction at the intersection. However, in a piece of cloth, if the support tree intersects itself, this intersection effectively becomes a vertex of the graph. This introduces a potentially problematic cycle into the support graph. We can check to see if the tree intersects itself by augmenting the support tree with edges between each pair of vertices corresponding to concave-adjacent convex vertices (e.g., in Figure 2.19(d), we would connect vertices (1,2); (2,3); (3,4); (4,5); and (5,1)). If this new graph is not planar, then the support tree self-intersects, and is considered invalid.

If a pair of intersecting convex vertex visibility regions are chosen as the site for a new vertex v_n, the vertices (v_i, v_j) must be concave-adjacent. If they are not concave-adjacent, that would imply that there are convex vertices in both polygons formed by partitioning the original polygon into two pieces using edges $v_i v_n v_j$. We are only allowing degree 3 vertices to simplify the construction; therefore, this implies that a tree edge crosses $v_i v_n v_j$, violating our planarity constraint.

We have developed an algorithm for constructing a support tree; however, it involves an optimizer, and is non-deterministic, which makes it difficult to prove anything regarding correctness or termination. Therefore, we will only provide a basic sketch of the algorithm rather than a full description.

To clarify our algorithm, the comb-shaped polygon in Figure 2.19 will be used as an example. The basic form of the construction is as follows: 1) find the regions visible from each currently existing vertex of the tree (Figure 2.19(a)), 2) find an intersecting pair of visibility regions, place a vertex in the intersection, and add edges to this vertex (Figure 2.19(b)), and 3) add the visibility region from this new vertex to the list of regions that are being considered. This is repeated until we have a tree that describes the visibility structure of the polygon (Figure 2.19(c)). At this point, we prune the intersecting regions to ensure mutual visibility between them, and then we place an

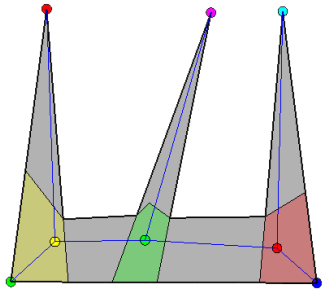


Figure 2.20: Output from the support tree construction algorithm.

initial estimate of the support tree in the polygon (Figure 2.19(d)). Our algorithm next runs an optimizer to try to make non-pinned vertices of the tree be spanned vertices, while keeping the tree inside the polygon. If the optimizer is unsuccessful, the algorithm repeats with a new tree by choosing visibility intersections in a different order.

The cost function tries to maximize the spanning of $\mathbb{R}^2$ at each interior vertex of the tree. If the edges at a vertex do not positively span $\mathbb{R}^2$, a penalty is assigned. There are additional penalties for vertices that leave their visibility region, or that leave the polygon entirely.

We select visibility intersections randomly, since both bad and good solutions may form clusters in the graph of possible intersection choices. However, in the worst case, this algorithm still has an exponential runtime. Additionally, there is no guarantee that the optimizer will correctly find a solution if one exists.

We have implemented this algorithm in Matlab, using CGAL [13] and VisiLibity [52] to handle polygon and visibility operations, and OGDF for graph planarity testing. We have tried three of Matlab's built-in optimizers: pattern search (`patternsearch`), the genetic algorithm (`ga`), and constrained nonlinear optimization, which utilizes an SQP method (`fmincon`). We found that the genetic algorithm had the best results, both in terms of speed and convergence, possibly because no gradient is available for the cost function. Figure 2.20 shows the result of running the algorithm on the comb shape.

2.5 Conclusion

We have determined that for simple cloth polygons,

$$n_{\min} \in \left[n_{\text{convex}}, n_{\text{convex}} + \left\lfloor \frac{n_{\text{concave}}}{3} \right\rfloor \right] \tag{2.5}$$

for simple orthogonal polygons,

$$n_{\min} \in \left[n_{\text{convex}}, n_{\text{convex}} + \left\lfloor \frac{n_{\text{concave}}}{4} \right\rfloor \right] \tag{2.6}$$

and for non-simple polygons,

$$n_{\min} \in \left[n_{\text{convex}}, n_{\text{convex}} + \left\lfloor \frac{n_{\text{concave}}}{3} \right\rfloor + 2n_{\text{holes}} \right] \tag{2.7}$$

We have shown that both bounds are tight for simple polygons, and that the lower bound is tight for polygons with holes. Additionally, we have developed the geometric method of using support trees to determine if a polygon is immobilized with a given grasp. This method is particularly

valuable for visually determining if a polygon is likely to be immobilized with a given grasp.

Our theorems directly led to an algorithm for constructing a valid grasp for any simple cloth polygon. This algorithm does not guarantee a minimal grasp, but it is a significant first step in designing grasps for cloth objects. The algorithm makes use of a simple linear programming method for verifying the validity of a given grasp. We implemented both a linear program based grasp verifier, and a support tree construction algorithm.

Natural extensions of this work include polygons with holes, and 3-dimensional cloth, such as cloth polyhedra. Our results are also applicable to cloth sensing. If the location of any grasp point is unknown, there is no way to show that the cloth is in a flat configuration. Thus, by sensing all of the grasp points, we can determine if a piece of cloth is flat.

Chapter 3

Cloth Folding

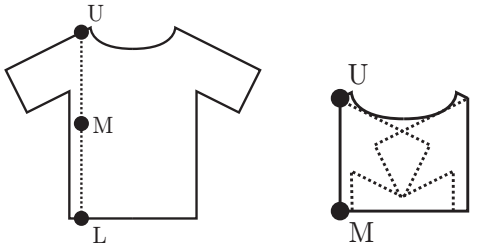
To manipulate cloth, we must first be able to grasp it. In the previous chapter, we showed that at a minimum, all the convex corners of a piece of cloth must be grasped to immobilize the cloth. In some cases, this is insufficient, and we may also need to grasp up to one third of the concave vertices. Even for a simple rectangular handkerchief, this indicates that four grasp points are required. Clearly, this rapidly becomes impractical for a robotic solution. Humans can only grasp two points at a time, and are able to accomplish folding tasks by using these two grasp points along with gravity and a flat surface; however, even folding a T-shirt can be tedious and cumbersome.

Osawa *et al.* implemented a folding technique in which they use a pair of robot arms to place a T-shirt on a surface consisting of one hinged plate [56]. One of the robot arms uses a high-friction flat plate to translate and rotate the shirt into position, at which point the hinged plate puts a fold in the shirt. This process is repeated until the shirt is fully folded. Correct placement during the folds is critical, and there is significant risk of destroying previous folds during the lengthy sequence of motions involved in the folding task.

A DailyMotion video on the web titled "ALLTRIBES" provides instructions for building a manually actuated foldable fixture, which consists of three hinged cardboard plates, depicted in Figure 3.1 [2]. This fixture is effectively a specialization of Osawa's work that only works for T-shirts, but which does not require repositioning of the shirt between folds. The shirt must initially be perfectly flat, and placed in a precise orientation on the plates. Any wrinkled regions in the shirt prior to folding will persist in the folded shirt. Once the shirt has been placed correctly, one of the side plates folds one sleeve inward. The plate then returns to its original flat position so that it is out of the way. Next, the plate on the other side folds the other sleeve inward. At this point, the central plate folds the shirt in half. With fast motors, this technique is very quick once the shirt has been flattened.

Figure 3.1: Foldable fixture (diagram based on Alltribes video [2])

A different folding technique uses two fingers to grasp the shirt at three points, and relies on



(a) Locations of grasp points (upper, middle, lower)

(b) Folded shirt

Figure 3.2: Configuration of T-shirt before and after folding using the two-finger technique

gravity and a table for assistance. A popular YouTube video titled "Japanese way of folding T-shirts" introduced this method [37]. The two-finger, three-point technique is simpler for a human to do; in particular, it does not require an extra mechanical system. The only needed components are two hands, gravity, and a flat surface. The T-shirt also does not need to be in a perfectly flat initial configuration; rather, it is only necessary that the three grasp points (shown in Figure 3.2(a)) are visible and accessible. To actually fold the shirt, a human would stand to the right of the shirt in the diagram, and reach across the shirt to grasp U in the right hand and M in the left hand. Point U is brought across to point L, at which point the right hand grasps L in addition to U. The human's hands are now crossed; uncrossing them and shaking the shirt results in the shirt hanging in the configuration visible in Figure 3.3(f). Now, the shirt is placed on the flat surface, and half the shirt is flipped back over to complete the fold. This sequence of steps is shown in Figure 3.3.

A key observation is that the location of the dotted line containing the grasp points matters for the final configuration, as it forms one of the edges of the final fold (Figure 3.2(b)). Sliding this line left or right (along with the grasp points) results in different amounts of the shoulder seam present in the final folded configuration. This also affects the location of the fold created using the table and gravity, which is the fold on the right side of the folded shirt. Typically, it is desirable to have a symmetrically folded shirt; thus, the right side fold has to be the same distance from the collar as the left side fold.

3.1 Automating the two-finger, three-point method

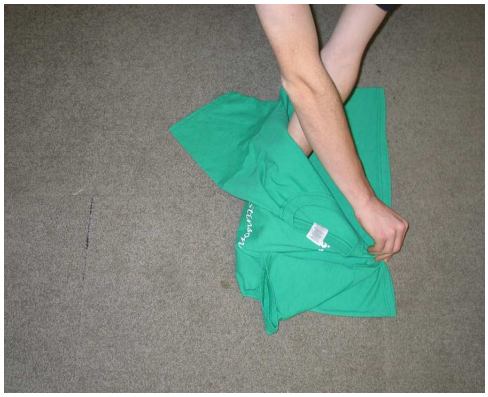
We observed that the middle grasp point must always be located halfway between the upper and lower grasp points, and the distance between these points never changes during the folding process, suggesting that a fixed-length manipulator suffices to grasp all three points and fold a shirt. For a real application, a custom gripper that reliably grasps multiple layers of the shirt is necessary. For our proof of concept, we attached magnets to the shirt at the three grasp points shown in Figure 3.2(a), and used a fixed-length iron bar to pick up the magnets. Since it is not necessary to release any grasps until the shirt is folded, the inability to drop the magnets does not matter.

Our folding system is pictured in (Figure 3.4). The major component is a 4-DOF SCARA arm. A high-torque servo motor has been attached to the end effector to give us the extra DOF necessary for folding. An iron bar is attached to the servo. Finally, we have a piece of sandpaper attached to the region of table that we use for the last fold of the T-shirt). The sandpaper prevents the shirt from slipping while it is being folded in half.

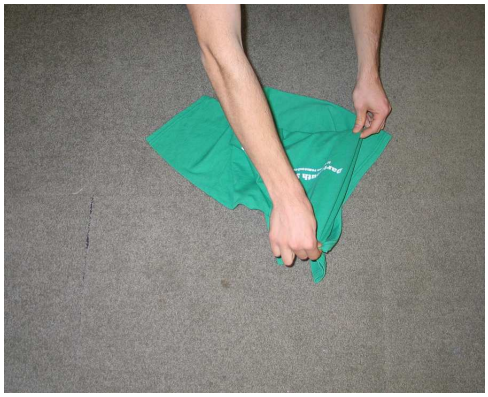
The folding process is shown in Figure 3.5. The steps are very similar to those described for manually folding a shirt with the Japanese method. First, the magnets at the top and middle grasp

(a) Flattened T-shirt

(b) Grasping top and middle points



(c) Grasping bottom point



(d) Uncrossing arms

(e) Shaking

(f) Result after shaking

(g) Laying the shirt on the table

(h) Folding completed

Figure 3.3: Manually folding a T-shirt using the two-finger, three-point method

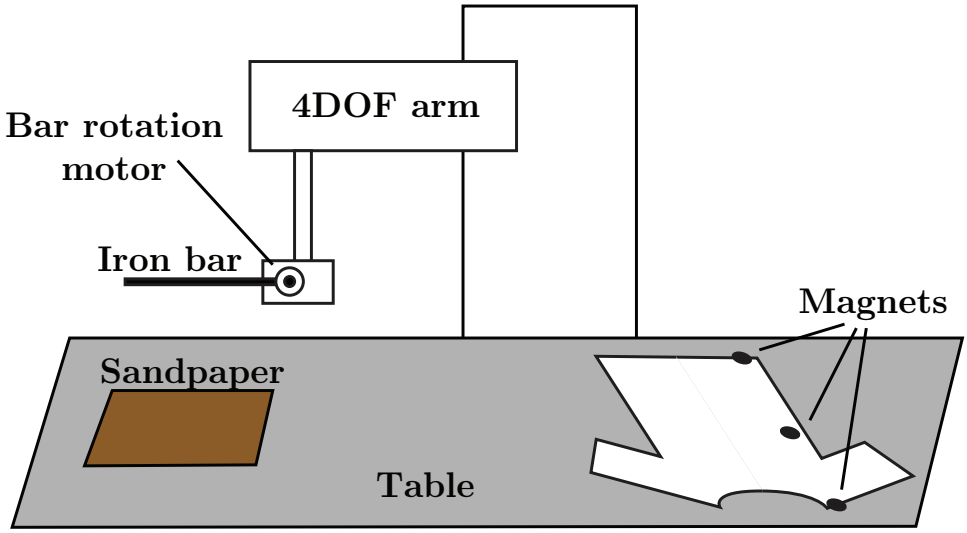


Figure 3.4: Folding system

points are picked up using the iron bar, followed by the bottom point (Figures 3.5(a) and 3.5(b)). These steps result in portions of the shirt sitting above the iron bar. Since we need the entire shirt to hang freely from the bar, the robot arm first rapidly shakes the shirt from side to side (Figures 3.5(c)-3.5(e)), and then shakes it vertically (Figure 3.5(f)). The shirt is laid out on the sandpaper on the table (Figure 3.5(g)). Finally, the shirt is folded in half (Figure 3.5(h)), with the sandpaper ensuring that the shirt does not slide off the table during this step.

These experiments suggest that T-shirt folding is not a particularly difficult problem if we are given an initially flat piece of cloth. Two fingers appear to be sufficient when combined with gravity and a table (after all, humans make do with the same limitations). However, a true clothing folding system should be able to handle any type of clothing, in any size. The fixed-length bar method has the limitation of only working for T-shirts of one size; a full system would need to have an adjustable length bar with at least two manipulators, similar to the one used by Shibata *et al.* [75]. In addition, a full system would need planning capabilities, which require an underlying understanding of the behavior of cloth in the presence of gravity, and knowledge of what sets of points are good grasp points for a two-fingered manipulator. In the next sections, we will discuss some basic analytical observations that serve as a basis for future formalized research in this direction.

3.2 Behavior of Cloth in a Vector Field

As already discussed, from examining human manipulation of cloth, we can see that two hands are sufficient. Manipulation typically involves suspending the cloth into some flat configuration, and then using a flat surface to assist with folding. To develop a planner to manipulate cloth autonomously, we must first understand how cloth behaves in the presence of gravity. In this analysis, it is trivial to generalize from cloth in the presence of gravity to cloth in the presence of any uniform or radial vector field.

First, consider a piece of string in a uniform vector field, such as gravity. Let one end of this string be grasped (Figure 3.6(a)). If the vector field is parallel to the string, then the force applied to every point mass in the string will also be parallel to the string. Since one end of the string is pinned, none of the point masses will move, and the string as a whole will not move. If the string is not parallel to the vector field, it will move until it is parallel to the field.

(a) Picking up top and middle points

(b) Picking up bottom point

(c) Before shaking

(d) Shaking to ensure that the shirt hangs freely

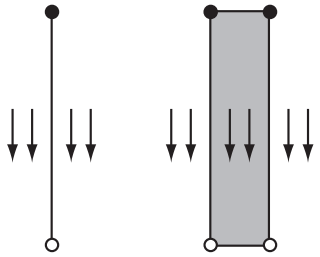
(e) After shaking

(f) Additional vertical shaking

(g) Laying the shirt on the table

(h) Folding completed

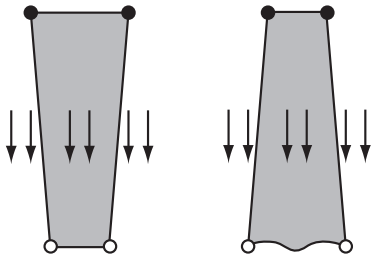
Figure 3.5: Folding a T-shirt with a robot arm



(a) String in vector field

(b) Cloth strip in vector field

Figure 3.6: String and a cloth strip in uniform vector fields. A closed circle denotes a pinned point, while an open circle is unpinned.



(a) No buckling

(b) Buckling

Figure 3.7: Cloth hanging in a vector field, with and without buckling present.

Now, we can consider a strip of cloth, which is effectively a large number of strings adjacent to each other. We will pin the two upper corners of the strip, which sets up a first order support line between these corners (Figure 3.6(b)). If each strand of string is parallel to the vector field, then each strand will not move, and the entire strip of cloth will not move. If the strip is not parallel to the vector field, then it will buckle as it moves into a configuration where each strand is approximately parallel to the field. Predicting the final configuration of buckled cloth is difficult. Since this state is undesirable when folding cloth, we will not analyze what actually happens to cloth when it buckles, and will instead simply avoid such states.

As already stated, we know that a piece of string pinned at one end and parallel to a vector field is in a stable configuration. Using this, we can develop a general model for cloth in a vector field. A piece of cloth in a vector field is in a stable configuration if, for every point p on the cloth, there exists an immobilized point p_i (either directly pinned, or immobilized by support lines), such that the vector $\overrightarrow{p_i p}$ lies entirely in the cloth and is parallel to the vector field. Figure 3.7 shows an example of cloth that is stable, and also of cloth that will buckle.

In addition to uniform vector fields, we can also consider radial fields emanating from a single point p_r on the cloth. If p_r is pinned, then the entire cloth is stable if for every point p, the vector $\overrightarrow{p_r p}$ lies entirely within the cloth and is parallel to the radial field (Figure 3.8). Radial fields may be useful for manipulation; for example, spinning a piece of cloth about a center point may be a good way to flatten the cloth.

Non-uniform vector fields pose a problem, as it becomes difficult to analyze how cloth will respond to such fields. However, we do not anticipate the use of non-uniform vector fields for manipulation, and therefore we will not analyze this situation.

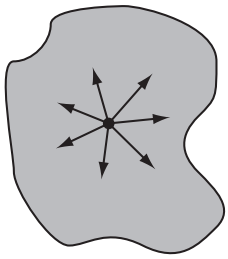
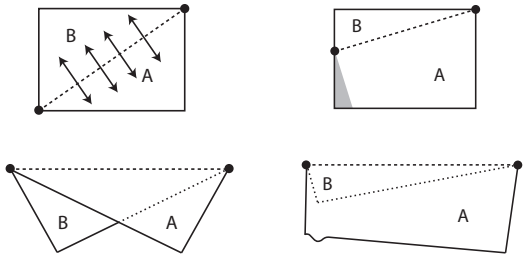


Figure 3.8: Cloth in a radial vector field.



(a) Grasp with no buckling

(b) Grasp that causes buckling

Figure 3.9: Two ways of gripping a cloth rectangle (grasp points indicated by closed circles)

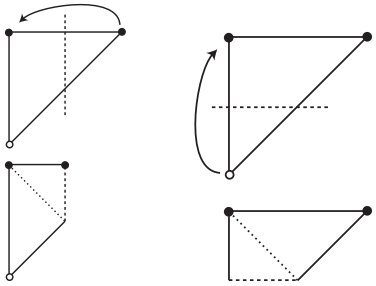
3.3 Folding With Two Fingers and Gravity

The grasps that we considered in the previous section all involved grasping along an edge of a piece of cloth. In this section, we will discuss grasps that result in support lines that cross the cloth, and grasps where multiple vertices of the cloth are pinned together at one point. These types of grasps set up folds in the resulting cloth. We can analyze these by considering separate polygonal regions of the cloth.

In Figure 3.9, a rectangular piece of cloth is grasped at two different sets of points. Figure 3.9(a) indicates the direction of the gravity field on either side of the fold, along with the resulting shape when the field is applied. Triangular regions A and B can be considered independently, as they are separated by the dashed first order support line. Each region is stable, as explained in Section 3.2. In Figure 3.9(b), the grasp shown results in buckling, since points p in the shaded region do not have corresponding supported points p_s such that $\overrightarrow{p_s p}$ is parallel to gravity.

Grasping multiple points with the same finger also introduces folds into the cloth. In Figure 3.10(a), we take the right point of the triangle, and fold it across to the upper left point. The resulting fold lies on the line that reflects the right point onto the upper left point. In general, a fold introduced by gripping two points of the cloth with the same finger will lie on the line of reflection between the two points, as shown in Figure 3.10. However, there is no guarantee that the resulting folded shape will not buckle in the presence of gravity. For example, Figure 3.10(a) will buckle if the right point is not pinned. In addition, the folded shape in Figure 3.10(b) may be unstable, as the smaller triangular section is not supported by a support line.

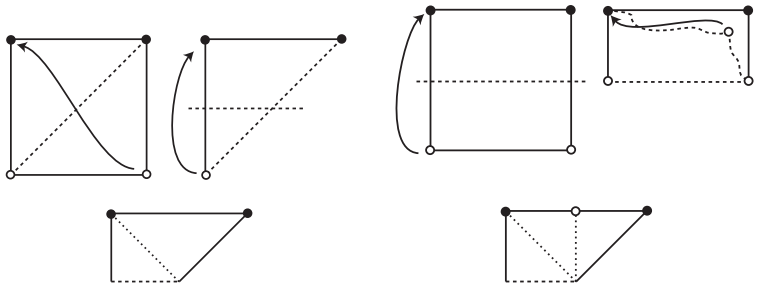
The order of the folds is also important. For example, there are two ways to fold a square of cloth to match the folded shape in Figure 3.10(b). The first ordering is to fold the lower right corner to the upper left corner, followed by folding the lower left corner to the upper left corner (Figure 3.11(a)). The second option folds the lower left corner to the upper left corner first, followed by the lower right corner to the upper left corner (Figure 3.11(b)). In this second case, the resulting



(a) Stable grasp at 3 points using 2 fingers.

(b) Possibly unstable grasp.

Figure 3.10: Two grasps of a folded triangle.



(a) Possibly stable folding of a square

(b) Unstable folding of a square

Figure 3.11: Two fold orderings for a cloth square.

shape has a loose unpinned vertex, indicated by the open circle. This vertex can fall downward, leading to buckling.

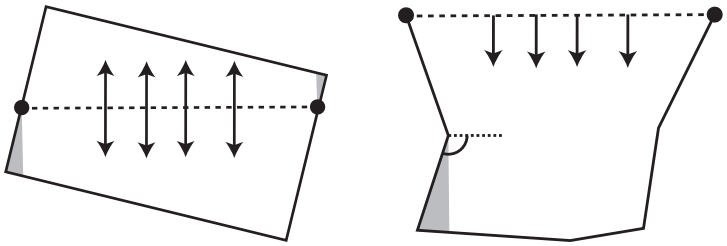
The folded shape in Figure 3.11(a) poses an interesting question, as we have experimentally verified that it does not buckle, yet it does not satisfy the conditions given in Section 3.2, as the points in the folded triangular region in the lower left do not have immobilized points directly above them. In experiments with very flexible cloth, the lower fold does not end up tightly folded along its entire length, although no buckling is introduced. The cloth model needs to be developed further to understand cases like this shape and the triangle in Figure 3.10(b).

3.4 Enumerating Valid Two Finger Grasps

The theoretical examples in this chapter have all used two fingers to grasp hanging cloth. Such grasps involve the cloth hanging from the support line between the two grasped points. We can make two important statements that must always be true for a grasp that involves no buckling.

Rule 1. If a grasp point is located on an edge e of the cloth, the other grasp point must be placed such that the support line between the two points is perpendicular to edge e. If this is not the case, there will be regions of cloth that will buckle (illustrated by the shaded regions in Figure 3.12(a)). By extension, if both grasp points are edges, the support line must be perpendicular to both edges.

Rule 2. Edges "below" (in both directions from) the support line must have angles $\leq 90^\circ$ with respect to the support line, as measured from the edge towards the interior of the cloth polygon.



(a) Grasp violating rule 1

(b) Grasp violating rule 2 (the indicated angle exceeds 90°)

Figure 3.12: Illustration of the two rules governing two finger grasps

Any edges with angles greater than 90° will never be supported by the support line (the shaded region and its edges in Figure 3.12(b) are not supported).

There are three possible cases for valid two finger grasps: edge-edge, vertex-edge, and vertex-vertex. It is also possible for one of the fingers to be placed somewhere inside the cloth polygon, but all such grasps result in buckling, so we will disregard these cases.

3.4.1 Edge-Edge

Based on Rule 1 above, edge-edge grasps can only take place between parallel edges, as a support line between non-parallel edges cannot be perpendicular to both edges. For parallel edges, the grasp can fall anywhere along the length of the edges provided that the support line remains perpendicular, and provided that the support line is fully contained within the cloth polygon (which is equivalent to stating that the grasp points must be mutually visible within the polygon).

To enumerate all possible edge-edge grasps, first find all pairs of parallel edges. Any pairs of points on parallel edges that are mutually visible constitute a valid grasp, provided that Rule 2 is met.

3.4.2 Vertex-Edge

For vertex-edge grasps, the support line must be perpendicular to the edge. This implies that for every vertex-edge pair, only one point on the edge may produce a valid grasp (a normal from this point must intersect the vertex, with the normal lying entirely in the cloth polygon). From Rule 2, the edges adjacent to the vertex must have angles $\leq 90°$ with respect to the support line. This implies that the vertex must be a convex vertex.

To enumerate all valid vertex-edge grasps, loop over all edges. For each edge, sweep a normal along the edge, recording all convex vertices encountered by the sweep (along with the point on the edge where the convex vertex was encountered). For all these points, Rule 2 must be met to ensure that the grasp is valid.

3.4.3 Vertex-Vertex

For vertex-vertex grasps, both vertices must be convex, and mutually visible. To enumerate all such grasps, find all pairs of mutually visible convex vertices, and check if Rule 2 holds for each pair.

Now, for a given cloth polygon, we can enumerate all valid two-finger grasps that do not result in buckling. Based on this list, it is possible to build a simple planner that picks a fold from the

list, and enumerates a new list of two-finger grasps for the folded shape. By repeating this process, it is possible to fold a piece of cloth until it fits within some maximum size given to the planner.

The planner relies on being able to compute valid two-finger grasps for partially folded shapes. The above enumeration methods are still valid; the only modification is that there may be multiple layers of cloth, which we can view as multiple overlapping polygons. A grasp must be valid for all polygons for it to be valid for the folded piece of cloth.

Chapter 4

Knot Tying

String can be thought of as one dimensional cloth. We have used string models as a fundamental element to develop an understanding of cloth. We used string lattices to build our theorems on cloth grasping, and also to aid in understanding the behavior of cloth in a vector field.

We have also worked directly on string manipulation, particularly on the challenging problem of knot tying. Autonomous knot tying machines have often used complex manipulators and extensive sensing. For example, Inoue and Inaba developed a system using a 6+1 DOF arm with stereo machine vision [36]. Wakamatsu *et al.* analyzed knotting and unknotting tasks, and developed a planner for tying arbitrary knots [82]. They tested their results with a 6 DOF arm and a single camera; however, they also proved that a SCARA arm (3 translational and one rotational DOF) is sufficient to tie any knot. They did this with the assumption that local manipulation did not cause effects to occur farther along the string (e.g., moving one segment of string did not introduce kinks elsewhere), and with the assumption that the knot is placed loosely enough to allow manipulation.

In contrast, we have shown that knot tying is possible with fixtures with very few degrees of freedom, and with no sensing (see Figure 4.1 for an example). In the context of this work, we consider fixtures to be static devices that manipulate an object into a desired configuration when the object is pushed against or through the fixture. A fixture allows a complex manipulation task to be achieved with a simple (typically one DOF) control.

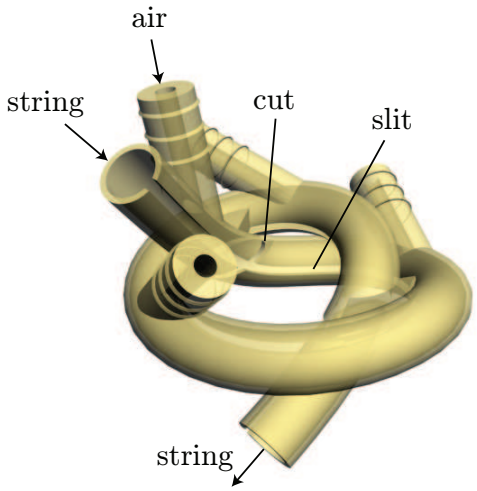


Figure 4.1: Example of knot tying fixture for the overhand knot

Our goal is to design a fixture such that string is inserted into one opening, pushed or pulled through, and knotted. It might seem surprising that this is possible, since we might expect the knot to become untied as the string is extracted. We accomplish this by exploiting the difference between pushing and pulling string through a curved tube. When pushed, string tends towards the outside of a curved tube, whereas when pulled, it moves towards the inside. If the inside is cut away, string will leave the tube when pulled, rather than being pulled back along the length of the tube. In all of our fixtures, the fundamental element is a tube in the shape of a knot, along with some means of extracting and tightening the knot.

A fixture's design is dependent on the properties of the string or wire for which the fixture will be used. Our fixtures for wire generally rely on the difference in behavior in pushing wire as opposed to pulling wire. Since it is impossible to move string through a tube just by pushing at the entrance to the tube, string fixtures rely on air power to act along the entire length of a string. Differences in air flow along the length of the string make this function as both a pushing and pulling motion. This typically requires string-tying devices to be more fully enclosed to prevent the string from falling out.

We will describe a set of techniques for inserting string or wire into fixtures, as well as techniques for extracting and tightening the knot. Finally, we will present several examples of fixtures that we have built for tying knots in a variety of materials. We will examine the ways of constructing knot fixtures, along with the benefits of our choice to use a rapid prototyper for construction. We will discuss the problems encountered, and apply the various techniques to a complete implementation of an autonomous system for tying an overhand knot. Our system is capable of tying an overhand knot in a bundle of 4 steel wires 87% of the time with 1 DOF and no sensing.

4.1 Overview of the knot tying process

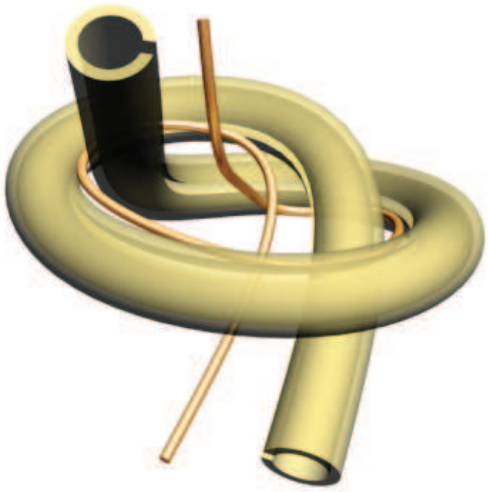
Knot fixtures are based on a tube in the shape of a knot (Figure 4.2(a)). Inserting string into this tube ties the string into a knot, although the knot has not yet been tightened. If it were possible to dissolve the fixture, the knot could be tightened, and the process would be complete.

Instead, as we pull on the ends of the inserted string, the string tightens towards the inside of the curves of the tube. Based on this observation, we can redesign the tube by placing a thin slit along the length of the tube in the region contacted by the partially tightened string. Now, pulling on the string will cause the string to leave the tube through the slit (Figure 4.2(b)). As in the figure, portions of the tube will be within loops of the string, preventing the knot from tightening fully. For some knot shapes, such as the overhand knot in the figure, we can free the knot from the fixture by sliding the knot along portions of the tube until the knot can be fully tightened. However, this process requires careful manipulation, and it seems likely that there exist knots for which this is not even possible. To resolve this, we extend the slit to cut through tubes (Figure 4.2(c)), enabling the knot to tighten fully in one smooth motion. Support pieces (not shown in the figure) hold the resulting tube components together.

4.2 Construction

We have built all of our fixtures using rapid prototypers, as machining techniques cannot build our knot fixtures out of a solid piece of metal due to the complex internal geometries. There are rapid prototypers capable of creating metal parts; however, these do not have the required resolution and surface smoothness for fixtures. Casting a fixture as a single piece is also not possible, since there is no way to make a rubber mold from a fixture. Prototypers capable of printing investment wax

(a) Knot-shaped tube



(b) Knot tube with slit. To extract the knot, the wire loops must slide off the ends of the tube.

(c) Knot tube with cuts to make extraction easier

Figure 4.2: Knot fixture basics

directly also exist, but the resolution is not high enough for knot fixtures. Once these technologies develop further, it should be possible to make more durable fixtures, with less friction than our plastic fixtures.

Our original process involved a fused deposition modeling (FDM) machine (a Stratasys FDM 2000). The support material used by our FDM machine had to be scraped off of the part by hand, which greatly limited our ability to build fixtures that had difficult to reach areas; in general, it was necessary to split fixtures into multiple pieces that were bolted together after cleaning, causing minor discontinuities in the knot tube. The FDM 2000 has a vertical layer size of 0.01”; as a result, fixtures produced on this machine had noticeable ridges between the layers, which snagged wire

ends in fixtures.

We have since switched to an Objet Eden 250, which has 16 micron (0.0006") thick vertical layers (with no ridges resulting between layers). The support material can be removed with high-pressure water, allowing us to build the fixtures as one piece. However, even with the Objet machine, support material is hard to remove from deep, narrow regions, such as the pressurized chamber described in Section 4.3.3. Cleaning such regions is impossible with the FDM machine, and very tedious and time-consuming with the Objet machine.

4.3 Insertion

The fixture-based knot tying process can be broken down into two major tasks: insertion and extraction. There are many ways to accomplish each task; in this section, we will discuss the various insertion methods and their advantages and disadvantages. Typically, string and wire techniques are distinct; for example, only wire is easy to push, whereas only string is easy to propel with air.

4.3.1 Pushing vs. pulling

There are two ways to get string or wire into a fixture: pushing or pulling. These differ in terms of where contact occurs between the string and the fixture. Pushing causes contact along the outside of the curves in the fixture, whereas pulling causes contact along the inside of the curves. If we can predict where this contact is for a given combination of fixture and material, we can design the fixture to correctly allow insertion.

Pushing a material along a tube requires the material to be capable of transmitting axial force over a long distance, through multiple curves. In general, stiffer materials such as wire are better at transmitting axial force. The inherent bending resistance reduces the risk of buckling. More flexible materials such as string are more difficult to push. Once string buckles, the normal force on the string from the walls of the tubes increases, causing even more buckling.

The easiest way to insert wire into a fixture is to push it by hand into the insertion end of the fixture. This has the advantage of being quick and easy to implement, and allows direct human sensing to inform the pushing motion.

The next logical step beyond manual pushing is motorized feeding, again at the insertion end of the fixture. In a conference paper, we described our implementation of using a robot arm to feed solder into a single-piece fixture [4]. We have since developed a much simpler feeding mechanism that uses a motor to drive a pair of rubber rollers. The rollers are clamped together with adjustable screws, and can thus be adapted to a wide variety of wire types.

Pulling for insertion is somewhat problematic, as our fixtures typically rely on a slit on the inside curvature of the tube for extraction, and this slit will interfere with pulling. If there is no such slit, pulling is quite reliable for a wide variety of material types, with the only limitation being friction between the fixture and the wire or string. However, extraction becomes very difficult if there is no slit; the fixture needs to be made of multiple pieces or have walls that slide out of the way to allow the string to be removed.

Many of our designs use compressed air to force string through fixtures. The goal is to set up airflow along the length of the knot tube such that it both pushes and pulls the entire length of the string. We have tried directing a nozzle into the entrance tube of a fixture; however, this did not give good results, as this did not provide enough air pressure to set up a good flow throughout the tube, particularly in the presence of a slit in the tube. Other designs with multiple air jets spaced

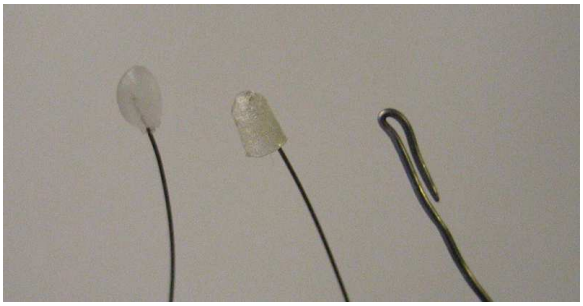


Figure 4.3: Three methods of reducing tip friction: wax, cap, and bent wire

(a) Slider with hook (b) Slider with ball-shaped ends

Figure 4.4: Possible track and slider designs

along the entire fixture gave much better results, particularly if the end of the string is modified in some way (such as with a small knot or a plug) to allow the air to apply greater force.

4.3.2 Pushing wire: Track/slider

The end of a piece of wire typically has a sharp point, which tends to get stuck in our fixtures. The plastic material produced by our rapid prototypers is soft enough that very hard and sharp wires dig into the tube walls. We considered three ideas for reducing the friction between the wire tip and the fixture: a bend in the end of the wire, a wax bead, and a cap (Figure 4.3).

Adding a bend to the end of a wire is difficult to do with materials that do not hold bends well, such as fishing line or steel wire, but is easy for electrical wires and solder. Our robot arm system used this method, placing a loop in the end of the solder before inserting it into the fixture. A bead of wax is functional for more materials, but the wax needs to be manually molded onto the wire tip to ensure that it stays attached.

Finally, we can place a small plastic (prototyped) cap on the end of the wire. This cap is a small cylinder, with a hole bored part of the way down its length for the wire. Using a cap requires the knot tube to be of uniform diameter, with a slit small enough to prevent the cap from jumping out of the tube. Of these methods, the cap shows the most promise for automation, although a simple cylindrical cap is insufficient, as insertion into a small cap is difficult, and extraction is hard when the wire end is covered with a cap. By attaching a cylindrical cap to a sliding piece, the insertion and extraction problems can be simplified. A slider can be used to guide the wire along a track in the shape of a knot.

"Rails" for the track can be designed in many different ways; we considered two designs. The slider can either have a hook that wraps around the track (Figure 4.4(a)), or it can have a cylindrical or ball-shaped piece that rides inside of a tube (Figure 4.4(b)). We tried both methods, and found

Figure 4.5: Single air nozzle in tube

that we had better results with tubular tracks, as small hooks tended to flex undesirably.

A tubular rail needs to have a slit through the tube to attach the cap portion of the slider to the ball or cylinder that sits inside the tube. We designed this as a 2 mm slit, with a 1.25 mm wide connector between the cap and the ball. This fairly significant clearance allows a flat connector piece to slide through a variety of curvatures of the tube without getting caught. We used a ball-shaped piece inside the tube as opposed to a partially cylindrical piece, as the spherical shape will slide easily in any configuration. The loose clearance in the slit combined with the ball-shaped guide allows the slider to rotate about 45° in either direction from parallel. This rotation makes it easier to push the slider with wire through variations in the track.

In addition to rails for the slider, the track also needs to have a region to guide the wire into the correct shape. A half-tube is sufficient to guide the wire, and allows easy extraction during the tightening process. Based on this design, a track with just one rail tube does not seem feasible, as the only way to make the force on the slider not have a component pushing the slider against the slit wall is to put the rail tube behind the half-tube. In this case, the slit would be in the half-tube, which would allow the wire to jump into the rail tube. Therefore, we used a symmetrical design, with two rail tubes located above and below the wire-guiding half-tube.

With our rapid prototyper, none of the components of the slider or track can be smaller than 1-1.5 mm without risking accidental breakage during use, which makes it difficult for us to test this technique at smaller scales. However, this could be reduced further by building the slider out of stronger materials, such as some form of metal.

4.3.3 Pushing string

It is impossible to push string through a fixture by applying force only at the insertion point into the fixture, as we do for wire. Instead, we use compressed air to provide force along the entire length of the string within the fixture. We have examined several possible ways to apply air within fixtures.

Single air jets at intervals

Multiple air nozzles built directly into the knot fixture make air delivery consistent in a way that is not possible with a hand-held nozzle aimed at the entrance to the fixture. A basic approach is to have barbed connectors on the outside of the fixture, which directly deliver air to the knot tube (Figure 4.5). Such nozzles can be spaced at appropriate intervals along the knot tube, and they will set up an overall airflow along the tube, propelling string inserted into the tube.

This design has one flaw, which is the directionality of the air nozzles. Ideally, the air flow should be perfectly parallel to the tube everywhere, but at the nozzles, the flow is coming in at an angle. As indicated in the figure, this pushes the string up against the tube wall opposite to the nozzle, increasing the normal force (and thus friction).

(a) End view (b) Side view

Figure 4.6: Air ring with pressurized chamber (darker region indicates air chamber)

Air rings with pressurized chamber

A second method of setting up a flow of air in the knot tube is to use a symmetrical ring of small air nozzles. While this still does not provide air flow parallel to the tube near the nozzles, the symmetry should at least prevent the string from being wedged up against the tube wall. The air volume can be approximately equalized across the nozzles by having a pressurized chamber outside of the knot tube, with the nozzles embedded in the wall between this pressurized chamber and the knot tube (Figure 4.6). The pressurized chamber can be created by having a larger diameter tube fully enclosing the knot tube, with barbed connectors for air input. Occasional walls can be inserted to break up the air chamber into multiple smaller chambers, as needed.

Improving air impulse

Particularly with thin or smooth string, it is difficult to ensure that air exerts enough force on the string. This can be improved by expanding the size of the end of the string to enable the air to push it. One option is to tie a knot in the leading end of the string. This seems contrary to the point of the knot fixtures, but it would make sense to tie one or two simple overhand knots prior to using a fixture meant for a much more complex knot.

The second option is to attach a small plug to the end of the string, either with some sort of clamp, or by tying the string onto the plug. This serves a similar purpose as the knot, but is more effective since the diameter of the plug can be tailored to match the diameter of the knot tube. However, this also has the limitation of adding a step to the tying process.

4.4 Extraction

We consider extraction separately from insertion. The choice of an extraction method tends to be motivated most by the material that a fixture is meant for. For example, the size of a slit in the tube is dictated by the diameter of the wire or string, as we want to ensure that extraction is actually possible, while preventing the string from accidentally falling through the slit during insertion.

4.4.1 Slit

We can extract string from a fixture by cutting a slit in the knot tube on the inside of the tube's curves. At this point, it is necessary to slide the loops of the knot along the various sections of the tube to fully free the knot from the tube. It is unclear if this is always possible, although topologically it seems as though it should be, since a knot in the physical sense typically has two ends (as opposed to the mathematical sense with no ends), and can be reduced to a point. However,

Figure 4.7: Knot tube with shortened entrance and exit tubes

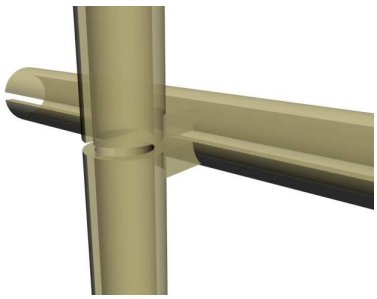


Figure 4.8: Supports for holding cut portions of tube in place.

it is very likely that there are knots for which it is prohibitively difficult to slide all the loops of the knot off the ends of the knot tube. For other knots, such as the overhand knot, it is possible to carefully position the entrance and exit portions of the tube, such that minimal motion is necessary to free the knot from the tube (Figure 4.7). In the figure, this was accomplished by cutting the tube at the point at which the tightening knot would intersect the tube, and by removing the ends that were freed.

4.4.2 Slit with cuts through tube

In cases where it is not possible to manipulate the tube to simplify freeing the knot from the tube, we can cut through any additional tubes in the way (previously shown in Figure 4.2(c)). This can be done by identifying a central region into which the knot will tighten, and by cutting any tubes that obstruct portions of the knot from reaching this center region. In our experience, it has only ever been necessary to cut across one tube to allow a portion of the knot to reach the center region; however, it is unclear if this is the case for every knot.

These cuts introduce discontinuities into the knot tube walls. With string, this is typically not an issue, as the cut size is relatively small (approximately the same as the diameter of the string), and the string does not exit the tube through the cut. However, this can be a problem for pushing wire if the wire is following the tube wall (i.e., any method except those involving a cap). In the single-piece fixtures, this is solved by shifting the tube slightly, with the tube wall after the cut being slightly farther out from the center of the tube than in the portion before the cut.

Cutting the tube in this way breaks the tube into multiple components. To ensure that these components are always aligned, we need to attach them to each other without introducing obstacles that prevent the knot from tightening. Figure 4.8 shows how these supports are placed. In the figure, the slit from the horizontal tube has been extended to cut the vertical tube into two pieces. Rectangular support pieces have been added to connect the cut portions of the vertical tube to points above and below the slit in the horizontal tube. These supports also serve to guide string from the horizontal tube into the cut in the vertical tube during the extraction process. The use of such supports is visible in the fixtures shown in Figures 4.25 and 4.32.

4.4.3 Rubber-covered slit

Another option would be to have a permanent slit with a rubber flap covering it. While difficult to construct with our rapid prototyper, machines do exist that can print rubber and plastic materials in the same part, making this type of design possible. This flap would provide some resistance to

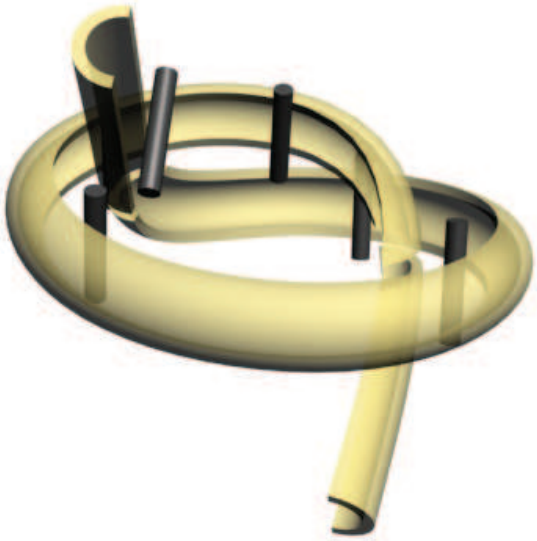


Figure 4.9: Knot tube with rollers

string, preventing it from easily falling through the slit, but would fold out of the way when the string is actually being tightened into a knot.

4.4.4 Retractable rollers

A more complex approach is to use a small number of rollers placed strategically along a half-tube (Figure 4.9). These can serve as low-friction barriers to guide string or wire to the correct path at critical points (e.g., near crossings), while leaving things open in less critical regions. This reduces the overall friction on the string or wire, and should allow for significantly longer knot tubes. Some mechanism for moving the rollers out of the way would then allow for extraction of the string or wire.

4.4.5 Two-piece approach (removable fixture wall)

In some cases, we may not want to have a standard slit, as the string or wire may accidentally escape the tube through the slit prematurely. To prevent this, the fixture can be designed with a removable fixture wall that can be closed up for insertion, and opened for extraction. It is possible to design a fixture such that moving portions of the fixture up and portions down (in one translation direction) exposes the entire string. Such a fixture would be comprised of just two pieces, for the up and down portions.

To show the design of such a fixture, we will begin by considering several definitions:

Definition 4.1. A theoretic **knot** $K \subset \mathbb{R}^3$ is a subset of points homeomorphic to a circle [21].

Definition 4.2. The **unknot** is defined as a planar, round circle.

Definition 4.3. A **link** is the finite union of disjoint knots. (In particular, a knot is a link with one component). An **unlink** is the union of any number of unknots all lying in a plane [46].

Definition 4.4. A **knot projection** is a projection of a knot or link from $\mathbb{R}^3$ into $\mathbb{R}^2$. 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 [46].

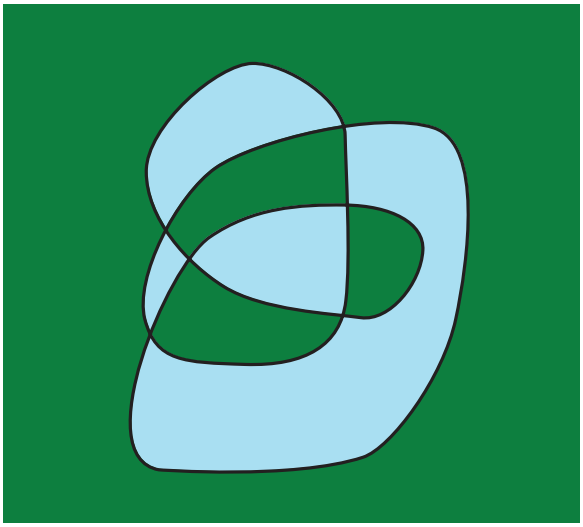


Figure 4.10: Knot coloring example

We will only consider regular projections. This means that each crossing consists of exactly two lines crossing, and that strands to not overlap. Such knot diagrams can also be thought of as graphs, which have the property that they are 4-regular graphs (graphs with degree 4 for every vertex).

Lemma 4.1. *Consider a knot projection. If we use the knot strands as the boundaries of regions, we can consider the projection to be a planar map. This planar map is 2-colorable.*

Proof. It is a known result that the dual of any planar Eulerian graph is a planar bipartite graph [87]. All 4-regular graphs are Eulerian, and therefore their duals are bipartite and 2-colorable. Since knot diagrams are 4-regular graphs, their duals (which are equivalent to the planar map of the knot) are 2-colorable. □

Corollary 4.2. *The planar map formed by any link is 2-colorable.*

Proof. A link is also 4-regular, and therefore the same proof applies. □

Figure 4.10 shows an example of a 2-coloring for the map of a more complex knot. The type of crossing (under or over) is not indicated, as it does not matter for coloring purposes.

The 2-colorability of a knot diagram can also be seen by using a constructive proof relying on Reidemeister moves and knot theory. We begin as before by considering a knot diagram. Every knot K has some unknotting number, denoted by $u(K)$ [21]. This represents the number of times a strand of the knot must be passed through another strand to turn K into the unknot. Equivalently, this can be thought of as switching which strand is on top of the other at a crossing in the knot. It should be clear that $u(K)$ is finite, as there are finitely many crossings in any knot. Thus, we can turn any knot (or link) into the unknot (or unlink) with a finite sequence of steps. According to Hass [31], we know that any diagram of a link (or knot) can be converted to another diagram of the same link (or knot) with a finite sequence of Reidemeister moves. Thus, we can use a finite sequence of Reidemeister moves to convert the unknot created in the first step to the trivial unknot.

If we reverse the above process, we can construct any knot (or link) from an unknot (or unlink) by using Reidemeister moves until we have the right knot shape, and then by switching crossings until we have the correct knot. It is trivial to show that 2-colorability is preserved during this

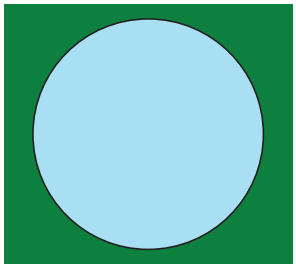
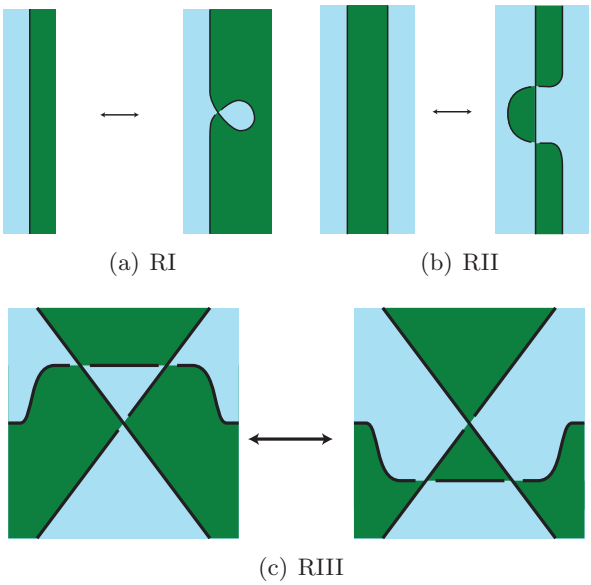


Figure 4.11: Unknot coloring



(a) RI (b) RII

(c) RIII

Figure 4.12: Coloring under Reidemeister moves

construction, starting from the 2-colorable unknot shown in Figure 4.11. We will show that applying Reidemeister moves does not change the colorability of the knot. These moves are abbreviated as RI, RII, and RIII. As shown in Figure 4.12, the three moves do not affect the 2-colorability of the knot. RI simply twists the string to add a loop. RII also adds a loop, but it does so by sliding one strand past a second strand. RIII slides a strand past a crossing, and this move only switches the color of the small triangle at the center of the diagram. The coloring of the rest of the knot diagram (everything outside of the regions shown in Figure 4.12) is not affected by these moves. To see this more clearly, note that the colors at the edges of the diagrams in the figure remain the same when the Reidemeister moves are performed. Finally, it is clear that switching crossings has no effect on colorability. Therefore, any knot remains 2-colorable as it is constructed from the trivial unknot.

Lemma 4.1 can be used to show that any knot box can be separated into two pieces with one translation in order to expose the knot. This is accomplished by projecting the knot box along any axis to obtain a knot diagram. Ideally, the projection with the minimum number of crossings should be used, to minimize the number of regions that need to be colored. Once this projection is obtained, it is used to slice entirely through the knot box along the axis of projection. This cuts the box into c pieces, where c is the number of components in the planar map formed by the knot diagram. Actually coloring the map (and box) is achieved by picking a region to color dark

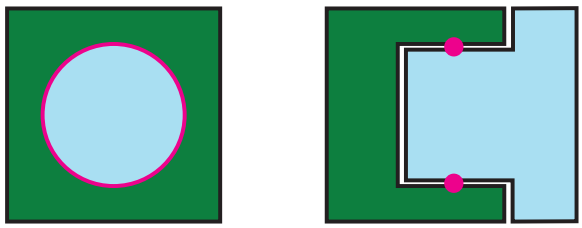


Figure 4.13: Diagram of simple knot box from the top (left) and side (right)

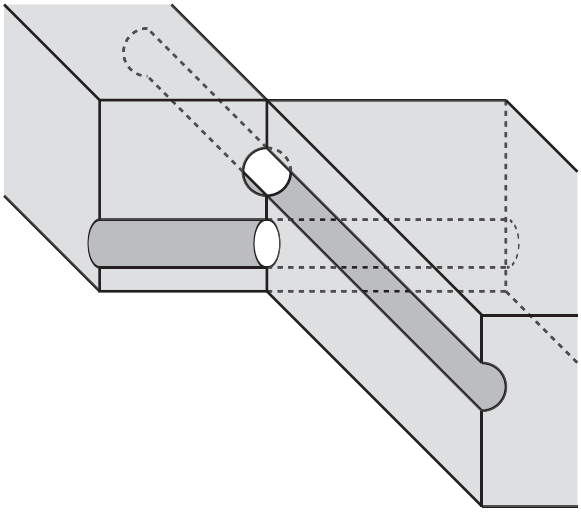


Figure 4.14: Diagram of a junction in the knot box

green, coloring all adjacent regions light blue, and so on. Next, solid rectangular caps are placed on opposite ends of the cuts. Label one cap dark green, and the other light blue. Each cap is then joined to the pieces of the knot box that correspond to its color. By following this procedure for creating a box, we can ensure that the tube is cut along its entire length, and that it will be fully exposed to air when one side of the knot box is removed.

The resulting box is shown in Figure 4.13 for the trivial unknot. The left side of the figure shows the projection of the knot, which is used to cut the box. The right side shows a side view of the box, with the rectangular caps attached. The small dark circles represent the knot tube, which extends into and out of the page to form the full circle. The light blue piece is a solid cylinder, which slides into a cylindrical hollow in the dark green piece.

The two-piece knot boxes we have considered so far have no connection to the outside world. As such, the only way to actually make a knot is by injecting it into the knot tube. We can easily extend this to knots with multiple strands, as such knots are links, which are 2-colorable by Corollary 4.2. Extending to boxes with entrance and exit tubes is also simple. If we draw the boundary of the knot box as another loop of string, we have a different link, which is 2-colorable. We can discard graph edges and planar regions outside the boundary without affecting the colorability (note that the region outside the boundary is not colored in this case), so the resulting modified map is still 2-colorable, and a knot box with a connection to the outside world can be broken into two pieces separable by translation.

There are at least two potential practical issues with constructing knot boxes in this way. The

first happens at junctions. Such a junction is pictured in Figure 4.14. One of the two pieces of the knot box is shown in light gray. The other piece would sit in the other two corners of the junction. As shown in the figure, each piece of the knot box has 2 components, connected on the diagonal across the junction (although in reality, only one of the two sets of components will be connected on the diagonal). The real issue has to do with extraction of the string. There will always be solid material both above and below any point in the tube (where above and below are defined along the slices through the box). Normally, this is not a problem, as the string can shift to one side of the tube as needed. At a junction, though, there will be a solid piece over the tube along one of the two diagonals, which will prevent removal of one of the pieces. We can attempt to solve this problem by drilling small holes at each junction. This will create some tolerance at the junctions, but it also adds extra possible paths for string, which is opposed to the goal of having a fully closed knot tube.

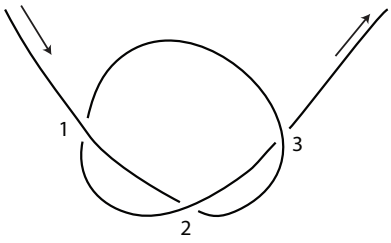
The second practical issue has to do with actually removing the knot from the box. Even removing one side of the box by sliding it apart from the other half can be tricky, since string is not infinitely thin. The actual knot may be even more difficult to remove. Consider again the knot box of Figure 4.13. If the string ends up left in the dark green piece, it is trivial to remove by tightening the circle. If it is left in the light blue piece, it will be wrapped around a cylinder, and will be more difficult to remove. This will become especially problematic for more complex knots. There is no clear means for solving this problem for two-piece knot boxes; however, the four-piece fixtures discussed in the next section provide a more feasible method of constructing fully enclosed fixtures that solves the extraction problem.

4.4.6 Four-piece fixtures

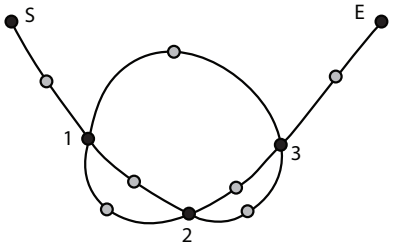
By splitting a knot box into four pieces instead of two, we can solve some of the practical issues involved in separating the box and exposing the knot. During conversations with Yuliy Baryshnikov, we developed a proof that for any knot, a four-piece fixture can be designed such that pure translations of the fixture pieces fully expose the knot. To construct a four-piece fixture, we begin by creating a knot diagram based on the knot description given as a Gauss code, which consists of an ordered list of junctions, each of which is denoted as an over-crossing or an under-crossing. A Gauss code can be thought of as an Eulerian path through the directed graph representing the knot. As an example, the overhand knot shown in Figure 4.15(a) has the Gauss code `1o, 2u, 3o, 1u, 2o, 3u`. We can construct a planar directed graph from the diagram by placing a graph node at each crossing, and a graph node on each strand between crossings (Figure 4.15(b)). Crossing nodes (shown in black) are located at crossings, and segment nodes (in gray) are located along knot segments between crossings. The resulting graph is 4-planar (a planar graph with maximum degree 4).

Next, we separate the graph into "up" and "down" layers by walking through the graph using the Gauss code. If an edge is adjacent to an over-crossing, that edge is in the upper layer; if it is adjacent to an under-crossing, it is in the lower layer. Each graph edge is labeled with '+' or '-' to indicate its layer (Figure 4.15(c)). For convenience, the start and end segments (adjacent to S and E in the figure) receive the same label as the next segments in from the start and end.

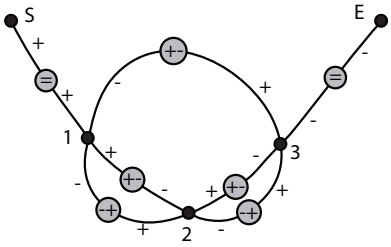
The segment nodes are labeled based on whether the adjacent edges are ascending from the lower layer to the upper layer ('- +'), descending ('+ -'), or remaining in the same layer ('='). We can uniquely identify segment nodes by using the adjacent crossing node labels; for example, the node that is moved in Figure 4.15(e) is 1(+ -)2.



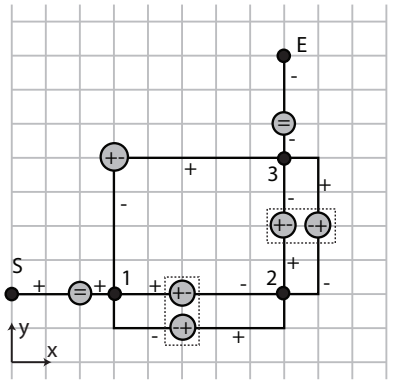
(a) Knot diagram



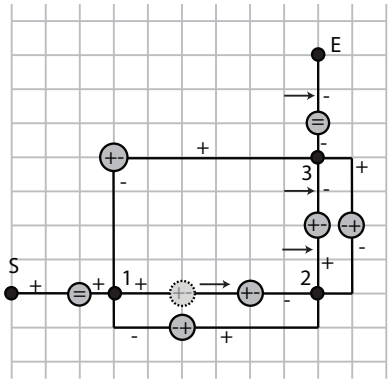
(b) Knot diagram interpreted as graph (black nodes are crossing nodes, gray nodes are segment nodes)



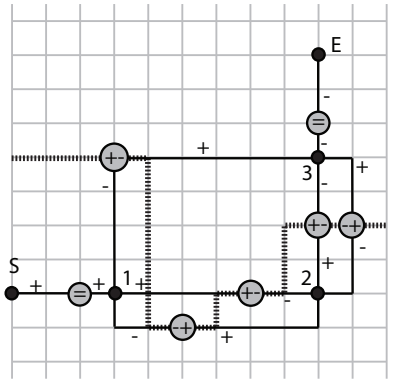
(c) Knot graph with edges assigned to layers, and labels on segment nodes



(d) Orthogonal version of knot graph (occlusions indicated with dashed boxes)



(e) Graph after expansion to remove occlusion



(f) Location of vertical cut through middle layer

Figure 4.15: Development of orthogonal structure for overhand knot

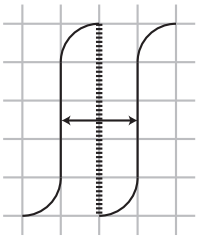


Figure 4.16: Two grid units of space are required between shaft edges to allow for rounded corners.

This graph is then converted into an orthogonal graph (Figure 4.15(d)). Tamassia showed that this can be done while preserving faces (the regions between graph edges) for 4-planar graphs [80]. Tamassia presented an $O(n^2 \log n)$ algorithm for constructing an orthogonalization with the minimum number of bends; this bound was subsequently refined by Garg and Tamassia to $O(n^{\frac{7}{4}}\sqrt{\log n})$ [27].

To construct the actual box, we add a third dimension to our orthogonal grid, with two possible coordinates: top or bottom. Edges are placed at the appropriate coordinate based on their label. At ascending or descending segment nodes, we add a shaft edge from the top layer to the bottom layer. Now, all planar edges are located in either the top or bottom layers, and the middle layer contains only out-of-plane shaft edges. This is similar to a two-layer circuit board, with the shaft edges equivalent to vias.

Two pieces of the four-piece knot box act as caps for the top and bottom layers of the 3D orthogonal graph. These pieces are removed by translating them up or down, respectively. The remaining two pieces slide together horizontally to form the middle layer. In order for these pieces to be separable, the middle layer must be arranged such that there is some projection direction in which no shaft edges occlude each other (in the example of Figure 4.15(d), the dashed boxes indicate the occluded nodes in both the x and y directions).

To allow for rounded corners, we can impose an additional restriction that there must be at least two grid units between shaft edges (and thus segment nodes) to allow for rounded corners. Figure 4.16 illustrates the need for the space. The two curved tube pieces are at different depths in the page. In order for the entire curved portion to be on one of the middle box pieces at the correct depth, the cut in the middle piece must be at least one unit away from the shaft, since the curves are of unit radius. This constraint only needs to be applied when the edges adjacent to a shaft edge are orthogonal to the projection direction (i.e., when the curves are visible along the projection direction, as in the figure). If the adjacent edges are parallel to the projection direction (as with the edges adjacent to segment node 2(+ -)3 in Figure 4.15), then the extra unit needed for the curve is guaranteed to be available, since there will always be at least one unit between the segment node and any adjacent bends, or crossing or segment nodes.

A very basic algorithm for expanding the graph to remove occlusion is as follows:

1. Pick a projection direction (x or y) with the fewest occlusions. With n shaft edges, this will always result in no more than $n/2$ occlusions. Without loss of generality, let the projection direction be along the y-axis.

2. For each pair of occluding edges (e_i, e_j) with $|x_j - x_i| < 2$:

   (a) Take all vertices with $x_v > x_j$, and set $x'_v = x_v + (2 - |x_j - x_i|)$. This expands the graph at x_j, shifting everything to the right of x_j one or two grid units (as needed) to the right. This shift preserves the graph regions (and the knot).

(b) Shift e_j to the right ($x'_j = x_j + (2 - |x_j - x_i|)$).

(c) If e_j was connected to an edge e_y parallel to the y-axis, keep e_y in place while shifting e_j, and add a new edge parallel to the x-axis between e_y and e_j (this new edge should be in the same layer as e_y).

In the example shown in Figure 4.15(e), only one node (1(+ -)2) needs to be moved to remove the occlusion in the y direction. The apparent occlusion of segment nodes (between 2(+ -)3 and 3(=)E) is not actually an issue, as the 3(=)E node remains in the same layer, and thus no shaft edge is present.

At worst, the graph will expand by $n/2$ units in the x-axis during this algorithm. At step 2a, we can insert a check to see if it is possible to simply shift e_j without having to shift the entire graph right of x_j, which may reduce the amount of expansion if the graph was not initially compact.

Once there is no occlusion along the projection direction, it is trivial to slice the middle layer to allow the two pieces to slide free, fully exposing the knot and allowing it to tighten. Given a list of shaft edges E_v, sorted by x coordinate, the slice between shaft edges e_k and e_{k+1} is given by the set of coordinates $\{(x_k, y_k), (\frac{x_k+x_{k+1}}{2}, y_k), (\frac{x_k+x_{k+1}}{2}, y_{k+1}), (x_{k+1}, y_{k+1})\}$. Figure 4.15(f) shows the location of the cut for the overhand knot.

Since this procedure of orthogonalizing the graph, splitting it into top and bottom pieces, expanding the orthogonalization, and slicing the middle layer of the fixture yields a four-piece fixture, we have proven the following theorem:

Theorem 4.3. *A four-piece fixture can be constructed for any knot consisting of one or more strands of string, such that all four pieces can be removed by pure translation without colliding with the string.*

Building the knot on an orthogonal grid with extra space between shaft edges guarantees that there will be enough room to round off the corners when actually building the knot box. Corners both in the plane and transitioning from the plane to the shaft direction can be replaced with arcs with grid unit radius. Thus, the maximum curvature is bounded exactly based on the choice of grid size. For a grid with spacing r, the maximum curvature is $\kappa = 1/r$.

4.4.7 Scalability

Fixture size is affected by the flexibility of the material for which it will be used (steel wire, solder, fishing line, string, or thread). Very flexible materials such as solder and string can be used in small-scale fixtures, while materials which are difficult to bend, like steel wire, require larger fixtures.

Additionally, our construction techniques impose limitations on scale. Prototyping very large fixtures is expensive (and impossible on our machines once the size exceeds a 10 inch cube). In the other direction, the Objet machine's resolution is high enough to allow fixtures to be built at a very small scale (for example, we have built a functional single-piece overhand fixture that is 0.5" in diameter). However, the plastic material becomes fragile when the feature size gets below 1 mm. Thin walls will break easily if they are bent. Cleaning also becomes more difficult when all of the internal openings are much smaller. As mentioned, metal should enable smaller scales when the technology is better developed.

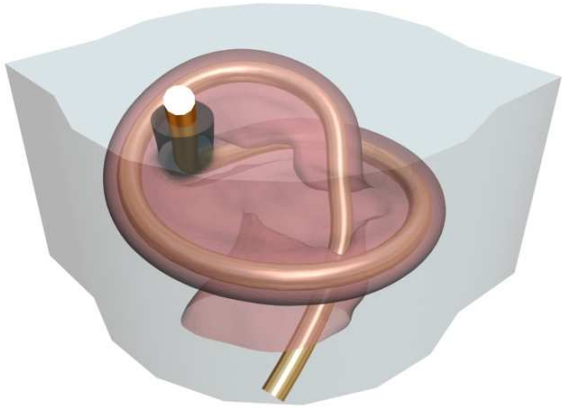


Figure 4.17: Single-piece fixture for overhand knot

4.5 Case studies

We combined the above techniques in various ways to produce a variety of fixture types that work on both wire and string. Many of these fixtures use slits for extraction; any of these designs can be further modified with a rubberized slit or rollers to make them more versatile. We have taken the most practical wire-tying system, the track-based system, and implemented a full autonomous knot tying device for tying overhand knots in bundles of steel wire. This will be discussed in detail in Section 4.5.8.

During the design process, we have sometimes found it useful to consider fixtures that do not allow the knot to remain tied during extraction (i.e., the string can only be extracted by reversing the insertion process). This allows us to evaluate different insertion methods without any interference from extraction regions cut into the tube.

4.5.1 Single-piece fixtures [1]

Insertion methods: Manual or motorized pushing for wire, air at tube entrance for string
Extraction: Slit with cuts through tube

Our initial single-piece fixture designs relied on manual creation of the slit and the cuts, with the result that both are uneven and relatively large with respect to the tube. The slit opens up into a hollow inner region in the center of the box, the knot extraction region. The tube, slit, and extraction region are subtracted from a cylinder or cube to form a "knot box". Figure 4.17 shows a single-piece fixture for an overhand knot. When wire is pulled after insertion, the wire passes through the slit into the extraction region. At this point, the wire can be tightened into a knot and removed through one of the holes in the knot box.

In order to remove the knot from the fixture without breaking the fixture, the volume swept by the string as it tightens into the knot must be free of obstacles. We can ensure that this is the case by making the entire interior region of the box topologically spherical. This is not a sufficient condition for extraction, as it is possible to have obstacles that maintain a spherical topology inside the box, but yet still cause the knot to become stuck. For example, if there is a spool-like shape in the fixture, the knot can wrap around it with no possibility of extraction (Figure 4.18). All of our knot boxes satisfy this constraint, which is particularly critical when multiple tubes are involved,

[1] This section is based on work published at ICRA 2008 [4].

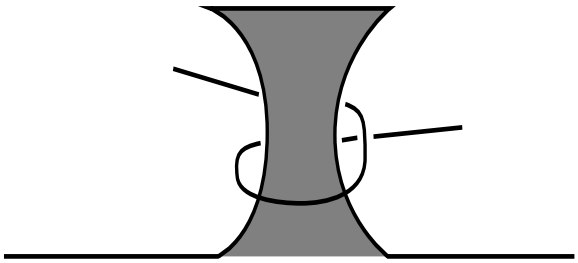


Figure 4.18: Obstacle preventing extraction

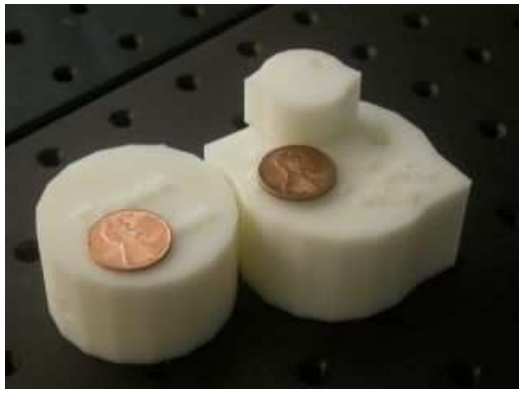
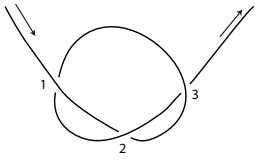
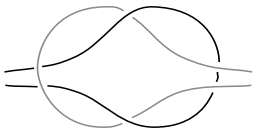


Figure 4.19: Square knot (left) and overhand knot (right, 2 sizes) fixtures



(a) Overhand knot



(b) Square knot

(c) Knots tied by manually using the knot boxes (overhand, double overhand, square)

Figure 4.20: Diagrams of the knot types, with knot examples

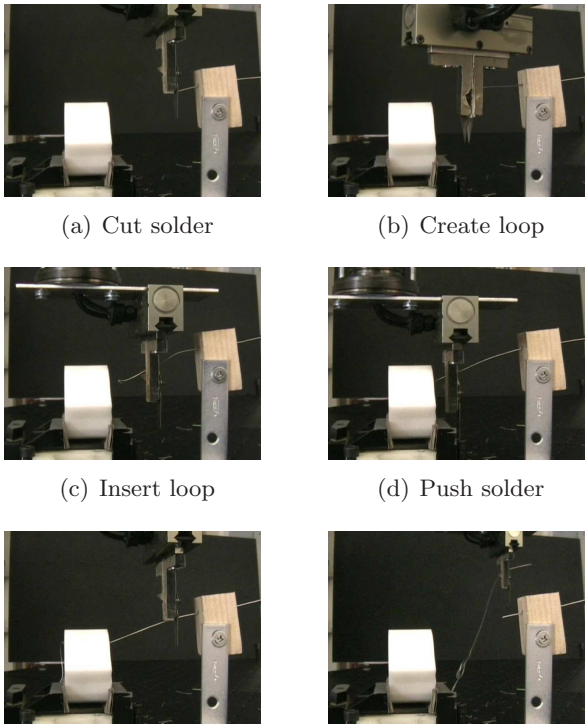
such as in the square knot.

We have developed knot boxes for two different knots (Figures 4.19 and 4.20). The most basic is the overhand knot. This fixture can be used in two ways. Pushing a single wire into it will simply tie a knot in the wire (Figure 4.20(c), left). Simultaneously pushing two wires through will tie both wires together into the same knot, effectively joining the wires (Figure 4.20(c), center). The next is the square knot, which allows us to more securely tie two wires together (Figure 4.20(c), right). The knot boxes all work on a variety of materials, including thin wires and fishing line.

In addition, we have used the overhand knot box for autonomous knot tying. A Cobra i600 SCARA arm has been fitted with a custom manipulator that can both grip and cut wire. We have been using solder for our experiments, as it is flexible, but also capable of maintaining a shape, which makes its behavior more predictable. The system is capable of pushing the solder through the knot box, cutting the knotted portion away from the spool, and extracting the knot. This entire process can be repeated without human intervention to produce multiple separate knots in sequence. This process is shown in Figure 4.21.

Using solder as the material, we are able to tie approximately one overhand knot per minute using the automated system. It should be possible to increase this speed, as we are running the robot arm at less than 50% of its full speed due to technical limitations of our setup. Under manual operation, it is possible to tie knots much more quickly. An overhand knot can be tied using the knot box in as little as 15-20 seconds. More materials can also be knotted by manually pushing them through the knot box (Figure 4.22), including

1. 0.025” diameter solder



(a) Cut solder (b) Create loop

(c) Insert loop (d) Push solder

(e) Cut solder (f) Extract knot

Figure 4.21: Autonomous Knot Tying

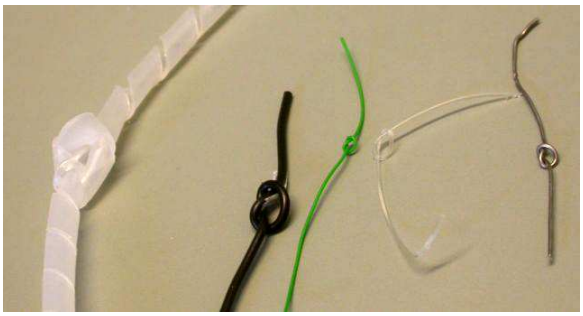


Figure 4.22: Overhand knots in different materials tied manually using the knot box (R to L: .032" solder, fishing line, 30 and 22 AWG wire, wire loom)

2. 0.032" diameter solder
3. 25 lb fishing line
4. 22 AWG single strand wire
5. 30 AWG single strand wire wrap wire
6. 5 mm diameter wire loom

We have observed that we can place the knots at a specific point on the wire with some precision. This occurs because the tightened knot must emerge from the knot box through a short section of

narrow tube. The knot will not pass through this tube while the knot diameter is larger than the tube diameter. As a result, the knot position is determined by the ratio of the lengths of wire that are pulled out of each end. We have observed a high degree of repeatability when holding one end of the wire at the same point, and pulling the other end out of the box. Multiple knots tied in this manner are within 5 mm of the same point on the wire.

We also experimented with sending string through these fixtures by applying compressed air at the entrance to the knot boxes; this was somewhat problematic because the air nozzle was manually held in place. In the majority of these experiments, the string would jump through the slit into the extraction region (and from there to the exit tube), thus skipping the knot tying portion of the tube.

4.5.2 Two-piece fixtures

Insertion methods: Manual pushing, motorized pushing, or air at the entrance
Extraction: Separation of two pieces of fixture

We have already shown that a two-piece fixture exists for any knot, such that the entire knot is exposed when the two pieces are pulled apart. Implementing a two-piece knot box is not practical, however, due to the problems that result at crossings, particularly relating to extraction.

The simplest approach to splitting a box into two pieces is to cut the knot tube down the middle. However, with this approach, string or wire in the tube will not consistently be in one of the two pieces, and it is impossible to slide the two pieces apart since string and wire are not infinitely thin. We can fix this problem by shifting the split so that it places most of the tube in one piece, with just a tube wall on the other piece. At crossings, the split must switch sides to ensure that the larger tube section remains on the same piece. Additionally, crossings must have some open space surrounding them in order to allow the finitely thick string to slide through the crossing without getting stuck when the pieces are separated.

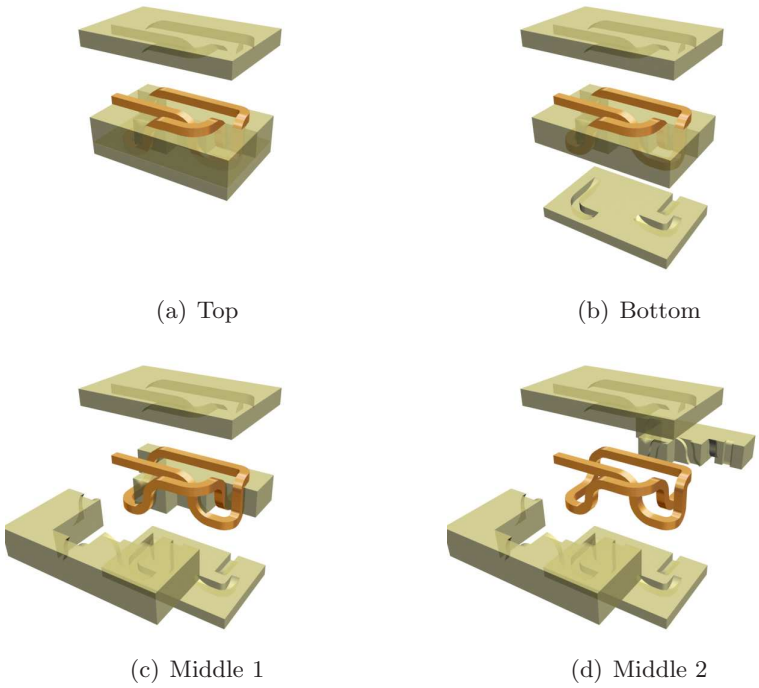
The open spaces at crossings, combined with tube walls being formed from different pieces, results in discontinuities in the tube wall. When wire is pushed through the box, it tends to get stuck on these discontinuities. We did not experiment significantly with string in two-piece fixtures, but based on our experience with single-piece and modular air-based fixtures, we believe that we would not gain any advantages from two-piece air-powered fixtures, as tube length is more of a limiting factor than the size of any slit in the tube.

4.5.3 Four-piece fixtures

Insertion methods: Fully actuated tube (cilia or rollers)
Extraction: Separation of four pieces of fixture

Constructing a four-piece fixture is a simple process. Given the graph produced by the algorithm in Section 4.4.6, it is trivial to produce an orthogonal set of splines in 3ds max. The corners are all filleted to a radius equivalent to the grid spacing. An extruded version of the knot spline is subtracted from the top, bottom, and middle fixture pieces, and the middle piece is sliced according to the path given by the algorithm. Once the fixture is built, it is clear that it is very easy to take apart without any interference from any string or wire in the fixture (Figure 4.23). We built four-piece fixtures for the overhand and square knots, along with plastic versions of the knots that fit within the fixture for demonstration purposes.

Figure 4.24 shows a rendering of the four-piece square knot fixture, along with splines for three other knots. The square knot and sheet bend both use two pieces of string, but the fixtures still only require four pieces. Additionally, the sheet bend is meant for joining ropes of different sizes,



(a) Top (b) Bottom

(c) Middle 1 (d) Middle 2

Figure 4.23: Sequence of steps for taking apart a four-piece fixture

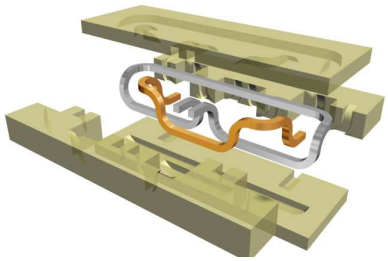
which is possible with four-piece fixtures. Finally, the bowline is a knot that involves a large loop, as indicated by the section of string indicated in a darker color. This loop would end up outside the fixture, requiring the user to reinsert the string into the fixture after passing the loop around some object.

It seems difficult to convert the mostly orthogonal design to a more rounded design without drastically increasing the length of the tube. Although some of the sharp curves can be smoothed easily, this is not the case for all the curves, particularly if we want to maintain the ease with which the fixture is taken apart. In theory, if all of the curves entering and exiting shaft edges existed along one principal axis (e.g., the x-axis, such as the two shaft edges between crossings 1 and 2 in Figure 4.15(f)), then we could set the projection direction along the other principal axis (the y-axis in this example). Provided that there is enough room between shaft edges in the x direction, this would allow much smoother curves into and out of the shaft edges, while still allowing the two middle pieces to separate easily. The graph could be further expanded in the x and y directions to allow the other bends to also be smoother, at the expense of a longer tube. The top and bottom pieces would not be affected by these changes.

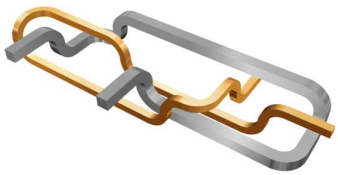
Due to the many bends resulting from the mostly orthogonal design of the four-piece fixtures, it is impractical to push wire or blow string into the fixtures. A practical implementation would require actuation along the entire length of the knot tube. Possible sources of this actuation are cilia-like manipulators, or many drive rollers. Since the tube is fully enclosed, there will not be any issues with string or wire going down unexpected side paths.

4.5.4 Single-chamber air fixtures

Insertion methods: Single air jets at intervals
Extraction: Slit with entrance/exit positioning, slit with cuts through tube

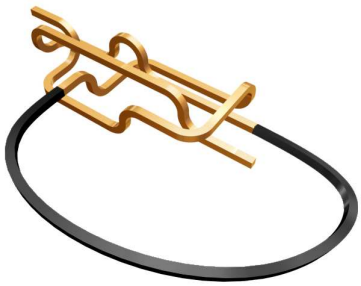


(a) Square knot (two pieces of string)



(b) Sheet bend (two different sized pieces of string

(c) Figure eight



(d) Bowline (dark loop is outside fixture)

Figure 4.24: Orthogonal versions of several knots

The single-chamber air fixture design is a modification of the single-piece fixture design, in which the knot tube is no longer subtracted from a larger solid. Instead, the knot tube exists as a thin-walled (typically 1-2 mm) tube with a diameter of 5-6 mm. The slit is cut at a consistent-sized 1-1.5 mm. The slit size is dependent on the type of string used, and is designed to be slightly narrower than the string, ensuring that the string does not easily jump through the slit when being pushed with air. Actuation is provided by air jets built into the tube, which are able to provide air at the same location through multiple experiments. For the overhand knot box, we used three air nozzles (Figure 4.25).

4.5.5 Double-chamber air fixtures

Insertion methods: Air rings with pressurized chamber
Extraction: Slit with no cuts; slit with cuts through tube is possible, but design is more complex

By enclosing the entire knot tube with a pressurized chamber, it is possible to place air rings at multiple locations along the knot tube. One challenge with this method is that the overall diameter of the tube is increased, requiring a larger fixture.

The biggest practical problem with this method is the process of cleaning out support material from the rapid prototyping process. Although we can soak parts in a sodium hydroxide solution to remove support material, the minimal access to the pressurized chamber means that it can take a long time to clean that region. In addition, prolonged soaking in NaOH makes the fixture more flexible, leading to warping and breaks.

The results from the overhand fixture (Figure 4.26) were not particularly promising. Due to the increase in diameter involved in this type of fixture, we switched to a thin suturing thread, and rescaled the fixture appropriately. The diameter of the inner knot tube was 2.5 mm in this design,

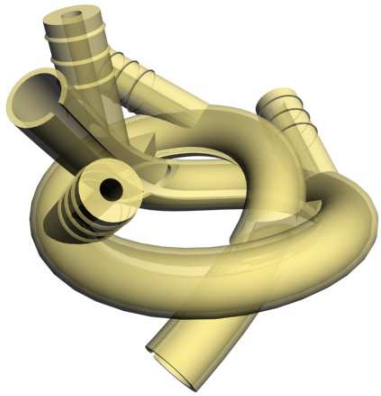


Figure 4.25: Overhand knot tube, with single-chamber air design

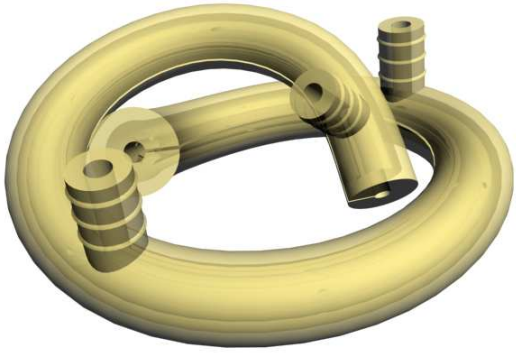


Figure 4.26: Overhand knot tube, with double-chamber air design

with a 0.4 mm slit. It was difficult to feed more than 10 cm of thread into the fixture (about half the length of the tube). The first problem resulted from the slit; since the thread was at least partially pulled by the air, it tended to get pulled into the slit. This caused the thread to get stuck, since the slit size is slightly smaller than the thread's diameter. The second problem is that there is too much air pressure in the knot tube, which causes some air to move backwards out of the insertion end of the tube, making insertion difficult.

4.5.6 Tubing with air accelerators

Insertion methods: Manual pushing combined with air rings with pressurized chamber
Extraction: None, or thin slit in tubing

To alleviate some of the problems with the high-friction surface of the rapid prototyped material, we experimented with using flexible plastic tubing instead, with prototyped air-powered string accelerators. The string accelerators are short double-chamber fixtures. The accelerator is about 1 inch long, with one or two rings of air jets from the pressurized chamber to the central string tube. The accelerators are slit along one side for extraction, and have barbed connectors to attach to the plastic tubing (Figure 4.27).

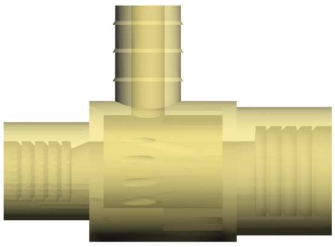


Figure 4.27: Air accelerator side view

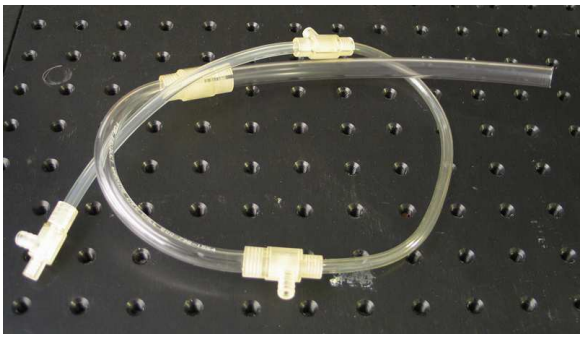


Figure 4.28: Overhand knot formed from air accelerators and increasing sizes of tubing

This system also suffered from an excess of air pressure, leading to air coming out the insertion end of the first accelerator, as well as excess amounts of air emerging from the slits. The air coming through the slits caused the string to try to emerge through the slits. We experimented with taping over the slits, which had the effect of increasing the back-pressure at the insertion end. By reducing the air flow to the system, we were able to achieve some success, inserting 85 cm of string into a relatively curved system in the shape of an overhand knot (Figure 4.28).

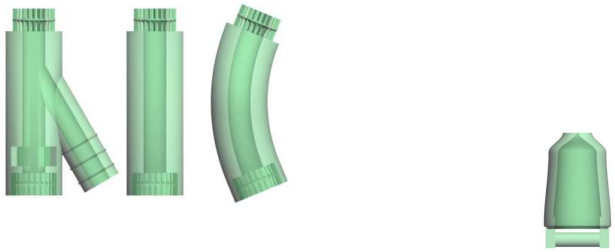
A second attempt to deal with the pressure problem was to use steadily increasing tube diameters, with appropriately sized air accelerators to act as adapters between tube segments. This approach worked somewhat better, but the string still had a tendency to emerge through the slits in the accelerators.

One unsolved problem is extraction from the plastic tubing. A possible solution is to use a knife to cut the tubing, without removing any actual material. Since the tubing is flexible, string can be pulled through such a cut. However, slit tubing is less flexible, since too much bending destroys the tube shape, and opens the slit. It is also more difficult to attach the tubing to the accelerators. Still, this method shows the most promise as being the most modular method that allows relatively simple extraction.

4.5.7 Modular, enclosed, air-powered tubing

Insertion methods: Single air jets, plug attached to string
Extraction: None

The final air-powered variation that we built is a fully modular system, consisting of straight and 45° curved segments that snap together, enabling the creation of essentially any knot (Figure 4.29(a)). There are also straight segments with single air jets placed at an angle (similar to



(a) Air jet, straight, and 45° curved segments. (b) Plug

Figure 4.29: Modular air-based system components

Figure 4.30: Overhand knot formed using the modular components, with two air jet segments (the two lighter colored segments)

those used in the single-chamber air fixtures). These air jet segments also have some openings to relieve excess pressure in the tubing. String is attached to a small plastic plug that is designed to take up almost the entire diameter of the tube, enabling the air to pull on the string with more force (Figure 4.29(b)). The plug also ensures that the string does not leave the tube through the pressure-reducing openings in the air jet segments. There is no way to extract string from this system, although once a design is tested with the modular system, it can be built as a continuous tube with a slit to allow extraction. Figure 4.30 shows an example of an overhand knot built with this design. The tube shown in the figure is 50 cm long, and thread with a plug on the end can be sent through almost instantaneously. In a longer curved tube (85 cm) with four air jet segments, it is slightly more difficult to get the thread through the whole tube due to friction with the tube walls.

This design still does not allow arbitrarily long knot tubes, as eventually the friction between the string and the tube walls cannot be overcome by the airflow. Smoother tube walls would help, as would greater air pressure, but the system would still be limited at some longer length.

Figure 4.31: Knot in bundle of wires

Figure 4.32: Track-based fixture for overhand knot

4.5.8 Track-based fixtures

Insertion methods: Track/slider, motorized or manual pushing
Extraction: Very large slit (half-tube), with either entrance/exit positioning or cuts through tube

In response to a request by Veyance Technologies, we have worked on an improved automated device for tying an overhand knot in wire. Specifically, Veyance wanted a device that could simultaneously tie an overhand knot in a bundle of 2 to 12 steel wires (Figure 4.31). This is something that was possible to a limited degree with solder in the original single-piece fixture, but the steel wire did not work well in the original knot box. The thin steel wire (0.01-0.015" diam.) has sharp points and is fairly resistant to bending. The wire comes in spools, and when left loose, it tends to coil in loose loops about 10 cm across. The sharpness of the wire caused problems in the original fixture because the wire scratched the relatively soft plastic walls of the fixture, and became stuck in the walls. With solder, it was easy to form a loop in the leading tip of the wire; with the stiff steel wire, this was not possible. The bending resistance also led to an increase in the normal force as the wire was forced through the tight (3 cm diam.) loop in the single-piece fixture.

We built a track and slider system as described in Section 4.3.2. Actuation was still accomplished by pushing on the wire at the entrance to the fixture; the slider acts as a passive device to keep the wire ends away from the track wall. The slider contains magnets to loosely attach the cap to the steel wire. We used cuts through the tube to make extraction possible. The cuts are narrow enough to not cause problems for the slider, while still being wide enough to allow the largest wire bundle to pass through them. The resulting track is shown in Figure 4.32.

To more fully automate the process, we built two additional systems and made a minor change

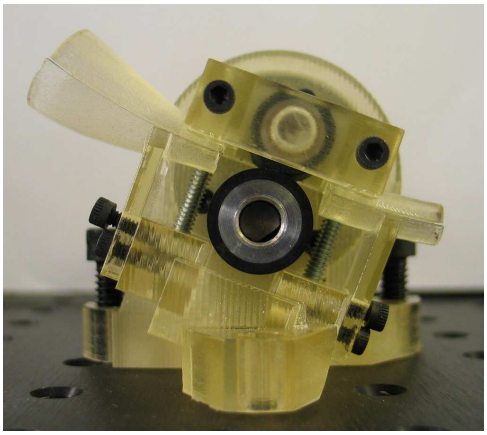


Figure 4.33: Feeding mechanism

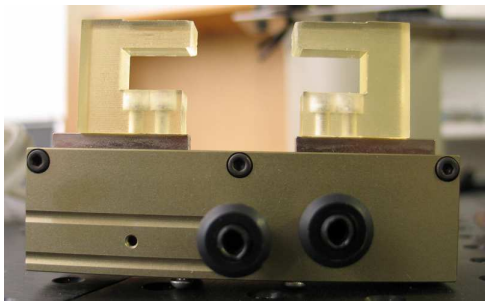


Figure 4.34: Pneumatic gripper with custom jaws

Figure 4.35: Full knot-tying system

to the track. The beginning of the track is carefully narrowed to ensure that the slider is in the correct configuration (normally, the slider can rotate about 45° from parallel to the track; this flexibility makes it easier to push the slider with wire). The feeder consists of a pair of rollers and two guide tubes. One of the rollers is powered by a standard DC gearmotor, and the other idle roller is adjustable to allow for different wire thicknesses (Figure 4.33). The first guide tube guides the bundle of wires to the rollers, while the second guide tube aims the wires into the slider.

Once the wire is pushed through to the end of the fixture, we need a way to hold the end of the wire while the rollers pull back on the wire to tighten the knot. The track at the very end of the fixture consists of just the rail tubes, with no central half-open tube for guiding wire, which provides space for a gripper to hold the end of the wire. The gripper is a standard pneumatic gripper, with a custom set of jaws that are designed to fit into the space in the track (Figure 4.34).

When combined, these three components (motorized rollers, track/slider fixture, and gripper) are a sufficient mechanical system for automatically tying knots in wire (Figure 4.35). One remaining problem is that the roller system we are using does not have enough friction to fully tighten knots in steel wire, again due to the bending resistance (although it is sufficient for producing a

Table 4.1: Success rates for various wire sizes

Wire size	Success
0.01”	100%
0.012”	87%
0.015”	60%

Table 4.2: Success rates for different bundles

Number of wires	Success
1	93%
4	87%
8[a]	80%
12[b]	0%

[a] 0.012” and 0.015” only
[b] 0.015” only

fully tightened knot in solder). However, if the wire is pulled back by some other mechanism (such as an external motorized spool), it is possible to tighten the knot. Replacing the rubber rollers with some other type of roller with more friction might also be sufficient.

4.6 Verification and Experiments

We performed 45 trials of the track-based overhand fixture using various wire thicknesses (0.01”, 0.012”, and 0.015”) and wire bundle sizes (1, 4, 8, and 12 wires), with 5 trials for each combination. Overall, we had a 78% success rate in tying overhand knots in wires. The thinner wires were easier to tie, probably because of the smaller bending resistance. Larger wire bundles became more difficult. Tables 4.1 and 4.2 show the results for the different variations.

In all cases of failure, the cause was an incorrect insertion into the slider. Once the wire was correctly inserted into the slider, the slider and track mechanism worked 100% of the time, as did the clamp and the extraction/tightening process. In particular, we were able to tie knots in bundles of 12 wires by manually aiding insertion into the slider. Insertion failures were primarily a result of the wire bundle not penetrating deep enough into the hollow cylinder within the slider, causing the wire to fall out of the slider very early in the process. Most of the time, only one or two of the wires would fall out of the slider, and the remaining wires could be tied into a knot (trials that resulted in this state were counted as failures). This could be improved by increasing the friction between the slider and the beginning of the rail tubes, or by a mechanism to hold the slider until the wires are fully inserted. More fine-tuning of the relative position of the slider with respect to the guide tubes on the feeder would also help increase reliability.

Table 4.3 presents a list of the different techniques, as well as their relative success for string and wire.

4.7 Curvature and friction in knot tubes

Friction is a major issue with our knot fixtures. Contrary to the typical case of surface area not changing the amount of friction, friction increases with the amount of surface contact between wire and the fixture. This occurs because the tube walls must serve to redirect force being applied to

Table 4.3: Comparison of knot tying techniques

Technique	Results with	
	Wire	**String**
Single-piece knot boxes	Generally good results; hard to design correctly	Very unreliable; no more than 1 success in 30 tries
Two-piece fixtures	Insertion tricky due to edges; extraction impossible	Insertion difficult; extraction impossible
Four-piece fixtures	Insertion difficult without actuation; very easy extraction	Insertion difficult without actuation; very easy extraction
Single-chamber air	Unable to move wire with air	OK; airflow can wedge string against tube wall
Double-chamber air	Unable to move wire with air	OK; difficult to construct, difficult to clear support material
Tubing with accelerators	Unable to move wire with air	OK; can move string over longer distance, but extraction is a problem
Modular air tubing	Unable to move wire with air	OK; plug enables longer string travel; friction eventually a problem; no extraction
Track-based fixtures	Excellent; easy to design; no problems with wire tip	OK, if string is constrained to not leave tube

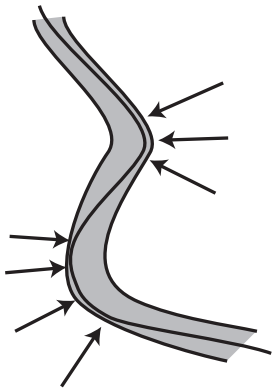


Figure 4.36: Normal forces on wire in an S-bend

the wire (Figure 4.36). If a given curve bends wire into a half-circle, there must be some normal force being exerted on the wire, since the wire has some bending resistance. The amount of normal force necessary does not change if the wire then enters a second half-circle that bends it back into the original direction.

Based on the following analysis, and as expected by intuition, the curvature of the tube significantly affects the ability to both push and pull string and wire through a tube.

Let $f(t) = (x(t), y(t))$ be a 2D parametric function defining a tube boundary. In general, we will use Bezier splines for this function, but any parametric function can be used. Now, let us assume that we are pushing a string through the tube, such that the string is contacting this boundary. To model this, we can view the string as a collection of rigid sticks connected by hinges, where the hinges contact the tube boundary and are located at points $x_i = f(t_i)$. Figure 4.37 shows a short portion of such an approximation, indicated by the thick lines. Some (axial) input force is

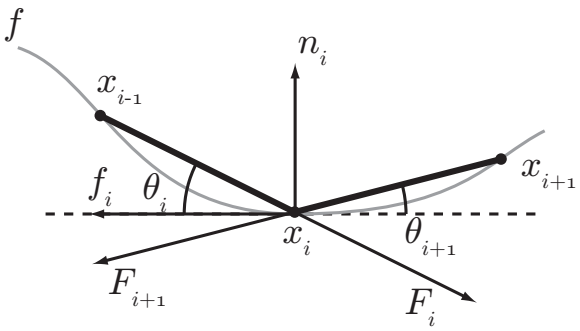


Figure 4.37: Spline approximation

applied to the first node, given by F_1. Gravity and bending resistance are disregarded in this model. Additionally, to simplify the friction equations, we assume that the string is currently sliding along the tube at constant velocity ($a = 0$), allowing us to use kinetic friction with $f_k = \mu_k n$.

Figure 4.37 contains the free body diagram for hinge x_i in this model. Applied forces are:

1. The normal force from the tube boundary (n_i).
2. The sliding friction force ($f_i = \mu n_i$).
3. The axial force being applied from the previous node (F_i)
4. The reaction force resulting from the axial force applied by x_i to x_{i+1} (F_{i+1}).

The force balance is

$$\sum F_x = F_i \cos\theta_i - F_{i+1}\cos\theta_{i+1} - \mu n_i \tag{4.1}$$

$$\sum F_y = -F_i \sin\theta_i - F_{i+1}\sin\theta_{i+1} + n_i \tag{4.2}$$

As we are assuming constant velocity, these equations sum to zero.

Given n nodes, this builds a system of $2n$ equations (F_x and F_y for each node) with $2n-1$ unknowns $F_2, \ldots, F_n$ and $n_1, \ldots, n_n$ (F_1 is known).

If we let $X_i = \frac{n_i}{F_i}$ be the ratio of normal force to axial force, we can obtain an expression for directly computing the normal force in units of axial force at any given node. This is done by dividing Equations 4.1 and 4.2 by F_i, yielding

$$\begin{aligned} \cos\theta_i - \frac{F_{i+1}}{F_i}\cos\theta_{i+1} - \mu X_i &= 0 \\ \sin\theta_i + \frac{F_{i+1}}{F_i}\sin\theta_{i+1} - X_i &= 0 \end{aligned}$$

We see that $\frac{F_{i+1}}{F_i} = \frac{\cos\theta_i - \mu X_i}{\cos\theta_{i+1}}$, from which we can obtain

$$X_i = \frac{\sin\theta_i + \tan\theta_{i+1}\cos\theta_i}{1 + \mu\tan\theta_{i+1}} \tag{4.3}$$

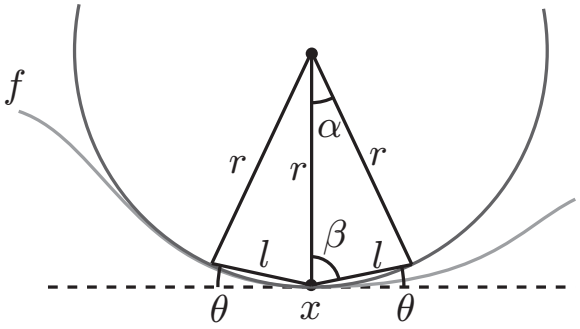


Figure 4.38: Continuous formulation

Equation 4.3 is dependent only on the angles to the neighboring nodes of node i, suggesting that it depends only on the curvature of the tube. As the resolution increases and we have more nodes, the angles tend to 0, and as a result $X_i \to 0$ as well. In order to get an expression for the force ratio at a point on the tube regardless of resolution, we will construct a continous formulation that expresses X_i in terms of the curvature, κ.

Figure 4.38 shows the basis for this formulation. We will consider a point $x = f(t)$ on the tube, which has curvature $\kappa = \frac{1}{r}$, where r is the radius of curvature. We can locally approximate f by a circle of radius r, as shown in the figure. We consider points before and after x, which are offset from x by arcs defined by α. These points are a distance l from x, and the angle from horizontal to these distances is given by θ. This angle will be used as both θ_i and θ_{i+1} in Equation 4.3. First, we must derive θ in terms of l and r.

We can apply the law of cosines to the triangle in the figure to obtain α: $l^2 = 2r^2 - 2r^2 \cos\alpha$. Since the triangle is an isosceles triangle, $\beta = \frac{\pi-\alpha}{2}$, and $\theta = \frac{pi}{2} - \beta = \frac{\alpha}{2}$.

$$\begin{aligned} \theta &= \frac{1}{2}\cos^{-1}\frac{2r^2 - l^2}{2r^2} \\ &= \frac{1}{2}\cos^{-1}\left(1 - \frac{l^2}{2r^2}\right) \\ &= \frac{1}{2}\cos^{-1} Q \end{aligned}$$

with $Q = 1 - \frac{l^2}{2r^2}$. With $\theta = \theta_i = \theta_{i+1}$, Equation 4.3 simplifies to

$$X_i = \frac{2\sin\frac{\alpha}{2}}{1 + \mu\tan\frac{\alpha}{2}} \tag{4.4}$$

Using the half-angle formulas $\tan\frac{\theta}{2} = \frac{1-\cos\theta}{\sin\theta}$ and $\sin\frac{\theta}{2} = \pm\sqrt{\frac{1-\cos\theta}{2}}$ and the inverse formula $\sin(\cos^{-1} x) = \sqrt{1-x^2}$, we obtain

$$
\begin{aligned}
X_i &= \frac{2\sqrt{\frac{1-\cos\alpha}{2}}}{1+\mu\frac{1-\cos\alpha}{\sin\alpha}} \\
&= \frac{2\sqrt{\frac{1-Q}{2}}}{1+\mu\frac{1-Q}{\sqrt{1-Q^2}}} \\
&= \frac{2\sqrt{\frac{l^2}{4r^2}}}{1+\mu\frac{\frac{l^2}{4r^2}}{\sqrt{\frac{l^2}{r^2}-\frac{l^4}{4r^4}}}} \\
&= \frac{\frac{l}{r}}{1+\mu\frac{\frac{l}{r}}{\sqrt{1-\frac{l^2}{4r^2}}}} \\
&= \frac{\frac{l}{r}}{1+\mu\frac{l}{\sqrt{4r^2-l^2}}}
\end{aligned}
$$

If we take the limit $\lim_{l\to 0}\frac{X_i}{l}$, we get

$$
\begin{aligned}
\lim_{l\to 0}\frac{X_i}{l} &= \lim_{l\to 0}\frac{\frac{1}{r}}{1+\mu\frac{l}{\sqrt{4r^2-l^2}}} && (4.5)\\
&= \frac{1}{r}=\kappa && (4.6)
\end{aligned}
$$

An obvious modification for reducing friction is to use a lubricant in the knot tube. We have used a graphite-based lubricant, which sprays on in liquid form and then dries to form a graphite film. Since this is dry, it does not leave residue on wire or string passing through the system. However, apart from residue, there is no reason why a wet lubricant could not be used for tying wire.

A more complex modification is the use of rollers, as discussed in Section 4.4.4. With rollers, there is no longer an issue of cumulative friction on the string, as the rollers will always be static with respect to the string. The friction has been transferred to the axles holding the rollers, and it can be easily reduced with oiled bearings. Thus, this method shows the most promise for allowing arbitrarily long fixtures; however, it is also the most complex.

Chapter 5

Lessons Learned

There are three main classes of flexible materials: curves, surfaces, and volumes. This thesis explored some of the most important examples of two of these classes: string and wire, and cloth. String and wire are major components in the space of flexible curves (rubber bands are another possible variation). Cloth is an important example of a flexible surface (stretchable sheets, such as rubber, and developable surfaces, like paper, form two other major classes of flexible surfaces).

In this section, I will discuss some observations made during this thesis, and some conclusions that might inform future work.

5.1 Knot Tying

During our knot tying work, I learned that fixture design consisted of trade-offs between easy insertion and easy extraction, and that string and wire require very different designs, with compressed air being the most useful method for string insertion. I learned that little to no sensing is necessary for knot tying, and that we need to take the thickness of string into account when designing fixtures.

5.1.1 Insertion/extraction conflict

The knot tying problem can be broken down into three major phases: insertion, extraction, and tightening. This thesis considers primarily the first two problems; we generally assume that if a knot can be extracted from a fixture, we can tighten it by pulling the ends of the wire or string. However, this is not the case for all knots. Some knots, such as the slipknot or sheepshank, require additional grasp points at loops in order for the knot to tighten correctly. While we do not address this issue, it was explored by Wakamatsu *et al.* [82]. They propose a method of finding regions of the knot that require extra fingers for tightening, and give a way to identify tightenable sub-knots that must be grasped.

The insertion and extraction phases of knot tying are typically in conflict. Designs that simplify insertion cause problems for extraction (e.g., fully enclosed tubes), and designs for easy extraction can make insertion very difficult (four-piece knot boxes). A truly robust system needs to combine insertion and extraction techniques in a way that resolves this conflict. Techniques involving fully enclosed tubes with removable walls (four-piece, rubber-covered slit, or retractable rollers) make insertion simpler, in the sense that the string or wire will not follow an undesirable side path, while still preserving easy extraction. However, these techniques may require better methods of providing force to the string or wire for insertion. We can continue to increase the number of pieces, making extraction progressively easier (with the limit of a dissolving fixture) at the expense of additional actuation for all the pieces.

Four-piece fixtures might require full actuation in order to be practical, but we believe that with this modification, they provide the best possibility for a universal technique for all knots.

5.1.2 Different fixture designs for string and wire

String and wire behave very differently, and require different classes of fixtures for knot tying. Wire can be pushed through long knot tubes, but poses additional difficulties due to memory effects and potentially sharp tips. Wire is also more sensitive to changes in the shape of the curve that forms the tube. Fixtures with slits take advantage of the lower flexibility of wire by depending on the tendency of wire to follow the outside of a curved tube when pushed, and exiting through the slit when pulled. With wire, it is possible to make the slit very large, to the point where the wire guide is a half-open tube, which greatly simplifies extraction.

Not surprisingly, we found that it is very difficult to push string through a tube. String buckles and becomes stuck very easily. We actuate string by using air, which simultaneously pushes and pulls on the entire length of the string. As a result, we need to minimize possible incorrect paths for air (and string), meaning that the slit needs to be very narrow, which can make extraction more difficult.

In conclusion, I learned that we need to have different fixture designs for string and wire.

5.1.3 Compressed air

Using compressed air to actuate string has several advantages. Due to the high speed of the airflow through the knot fixtures, string can be inserted very quickly, to the point where insertion looks almost instantaneous. We can also use air to reduce the amount of contact between string and the walls of the fixture. The double-chambered fixtures use rings of air jets to simultaneously apply air from multiple directions, with the goal of keeping the string in the middle of the tube and away from the walls. This could be extended further with even more air jets in the knot tube, similar to the way an air hockey table uses a very large number of small air jets to reduce the friction between the puck and the table surface.

Compressed air is the most promising approach that we have found for quickly pushing string through fixtures.

5.1.4 Minimal sensing systems can be effective

We have developed many different fixtures for knot tying, and none of them require any sensing. We have developed open-loop systems for tying using single-piece and track-based fixtures. We believe that sensing is not necessary for the knot tying process. Sensing might assist in tasks such as knowing when to clamp the end of the wire, or in aiding the insertion process, but it is not required.

5.1.5 Lack of accurate simulation drives simple designs

Accurately simulating string or wire is very challenging, particularly for knot tying since it involves self-collision, and even more so for knot fixtures. String in a fixture has a very high number of contacts with the fixture, and buckles somewhat unpredictably. For air-actuated fixtures, simulation would require incorporating a fluid dynamics engine, with an accurate model of the interaction of different types of string with a complex airflow that contacts the entire string at the same time.

Our fixtures were designed manually, and improved through multiple design-test iterations. The lack of input from simulations led to relatively simple designs for smaller knots; future work in accurate simulation would make more complex designs much easier.

5.1.6 String thickness can be a major challenge in extraction

The two-piece fixture we built did not work well due to the fact that string is not infinitely thin. The string interferes with separation of the two pieces, preventing extraction. If we take the thickness of string or wire into account while designing a method by treating the string as a tube rather than as a strand, this explicitly leads to more guaranteeable extraction. The four-piece fixture does this well by relying on separation directions that are designed to avoid any contact between the fixture and the string. Designs with this property will always have simple extraction that is not affected by string thickness.

The thickness of string also plays a role in the success or failure of other extraction techniques. Fixtures with slits need to have the slit width tuned to the string width to prevent the string from prematurely entering the slit, particularly for air-powered fixtures. We found that a slit that is slightly narrower than the diameter of the string works well for air-powered fixtures.

5.1.7 Precise tightening

As discussed earlier, precise positioning of a knot with single-piece fixtures is possible because the knot must leave a single-piece fixture through a narrow tube. The knot cannot emerge until the knot's diameter is smaller than the tube's diameter. By holding the leading end of the string at the exit from the fixture, and by pulling the trailing end back through the fixture and the narrow tube, we can reliably place the knot within 5 mm of the same position over multiple attempts.

From this, it is readily apparent that a funnel or short tube segment can be added to other fixture designs to assist with precise tightening. We can only be certain that the knot is in a known location on the string if we grasp the string or wire at the exit from the fixture, and have a known distance d_0 from this grasp point to the funnel. When we pull the trailing end back through the funnel, the knot will always be a distance d_0 from the grasped point of the string.

Distances shorter than d_0 between the knot and the end of the string are possible if we allow for some sensing. Once the knot is tightened, the string is pulled back through the funnel or tube, and cut off once the desired length has been pulled. Distances longer than d_0 are possible if we push extra wire through the fixture prior to grasping; however, measuring this may be difficult for string since there is less control over the insertion process (as compared to wire, where a motorized feeding mechanism can insert a known amount of wire).

5.1.8 Tying around objects

The knots explored in this thesis do not involve tying string to rigid objects, but we can modify our fixtures to incorporate rigid objects. For any knot diagram, a rigid body will be placed within loops of the diagram. If we use the four-piece approach, these rigid body pieces can be incorporated into the graph of the knot as shaft edges.

This adds several challenges to the design. The middle layers must now also account for the rigid body shaft edges in addition to the string shaft edges when searching for occlusions, and the top and bottom layers must be separable around the rigid body shaft edges (which turns the four-piece fixtures into six-piece fixtures). We will need to search for occlusions in the top and bottom layers as well to allow separation of those layers, although the separation direction need not be the same as the middle layer.

The shape of the rigid body affects whether or not this is possible. For some shapes, there may not be a way to prevent occlusion, which will prevent separation. Also, since we need to lift the top and bottom layers of the fixture away from the string before we can slide them sideways, the rigid body must be shaped in a way that allows this lifting motion.

5.2 Cloth Folding

The most important lesson I learned from our cloth folding experiments is that cloth folding is relatively easy, and that flattening cloth is much harder. Folding only requires a few fingers and simple techniques, whereas flattening most likely requires a much more complex system.

5.2.1 Folding is easy, flattening is hard

As demonstrated with our T-shirt folding system, and Osawa's hinged plate system [56], folding cloth is relatively easy. With our T-shirt folding system, no regrasping is necessary, so it is only necessary to locate the three key points at the beginning of the folding process. The hinged plate system requires some careful repositioning during the process, but the sliding manipulation required is relatively simple.

However, flattening clothing is a very difficult problem. The range of possible inputs is much larger, since clothing can be in many possible configurations when it comes out of a dryer, with inside-out clothing being one of the most difficult configurations. Given an article of clothing from the dryer, we first need to recognize what type of clothing we have. Given the type, we can make a guess as to the configuration based on vision or laser rangefinder data. Then, we need to somehow manipulate the clothing into a flat configuration. It seems likely that many degrees of freedom are necessary to accomplish this manipulation, and that there are no easy shortcuts or sensorless methods.

The flattening method that is most promising is picking up and shaking the cloth. If we pick up cloth by an arbitrary set of points and shake it into a hanging configuration, we can examine the resulting shape to determine some information about the cloth (such as classification and a guess at configuration). Kaneko and Kakikura demonstrated that classification by examining the silhouette of hanging cloth is possible [40]. With this information, we can put the cloth down, and try to grasp a new set of points that is more likely to hang the cloth in a flat configuration that can be used for folding. In this context, a flat configuration may mean one that has a fold present, with the two sides of the fold hanging without any wrinkles.

5.2.2 Not many fingers are needed for successful folding

Given gravity and a table, very few fingers are needed to fold cloth. We have shown how to fold a T-shirt using two fingers that grasp the shirt at three points, with the additional stipulation that the two fingers remain a fixed distance apart. Pants and shorts can be folded similarly by grasping along the waistline, with one finger at the center, and one bringing the two sides together. Rectangular items such as towels and handkerchiefs are also easily foldable with two fingers, although they may require that the fingers are a variable distance apart.

Since humans can fold clothing using two fingers, gravity, and a flat surface, it is likely that a robot can accomplish all folding tasks with just two manipulators. Large items, such as sheets, are an exception; humans find it difficult to fold such items by themselves, but two people (and four fingers) suffice, suggesting that a pair of laundry folding robots would suffice.

5.2.3 Components necessary for a laundry folding robot

From our results, we can make some informed conjectures on how a laundry folding robot would be built. As just discussed, the flattening task is very difficult, and requires a more complex system than the folding task. Sensing is necessary, as are multiple degrees of freedom.

An ideal robot would have a very thorough sensor system, probably combining machine vision and a laser rangefinder for full color and 3D position information. Color vision is useful for segmenting cloth into multiple pieces, and for finding edges, while the rangefinder provides 3D data that simplifies grasping. Sensitive grippers would also be useful, as they would be able to detect if a piece of cloth has been successfully grasped, in addition to providing warnings if the cloth starts slipping out of the grasp.

Although we have shown that one arm may be enough for some folding applications, two arms would make the process simpler, and additional arms may unnecessarily complicate the system. These two arms need to have 6 degrees of freedom, with a fairly large workspace (at least 5 feet side to side) to be able to grasp and stretch larger items such as towels. Both arms need to be capable of reaching most of the workspace, since many folds require bringing one side of the cloth across to the other side.

The grippers need to be capable of grasping the cloth both securely (for folding) and loosely (for hem following in flattening). They also may need to grasp multiple layers of cloth at once, as when grasping the middle of a T-shirt in the two-finger, three-point method. However, they may sometimes need to grasp just one layer, such as when turning clothing inside-out, or rotating a T-shirt or pants leg.

This system does not need to be mobile, as a folding station immediately next to a dryer should have enough reach to pick up clothes from the inside of the dryer. However, given that the system described has very broad capabilities beyond laundry folding, it might make sense to build this into a general-purpose mobile household robot, increasing the value of the system without increasing the cost.

5.3 Cloth Grasping and Immobilization

Convex vertices are problematic for immobilization, since each convex vertex requires a finger in order to be immobilized. Concave vertices add at most one third of a finger, which suggests that if we are designing a particular shape of cloth for an application requiring immobilization, we should attempt to minimize the number of convex vertices.

Chapter 6

Future Work

We have only begun to understand the behavior of flexible materials. While we know how to tie knots, we do not have a good way to simulate the process, or to analyze fixture types to understand why some techniques work better than others. We also do not have a good way to come up with fixtures for new knots, particularly for knots that are attached to objects (such as hitches). Although we know exactly how to fully immobilize cloth, and have some understanding of how to fold it, we do not know how to reliably flatten a piece of clothing. In both aspects, we need to better connect theory and practice, and examine ways to add new capabilities such as sensing, actuation, and modifications to the environment.

6.1 Exact number of fingers necessary for immobilization

In the cloth immobilization chapter, we discuss one class of polygons (pinwheels) that require fingers at points other than the convex vertices. We conjecture that polygons with pinwheel-like substructures are the only class of polygons that require extra fingers. (We are defining pinwheel-like substructures to be areas of the polygon with cyclic visibility structures, as in Figure 2.13.) Pinwheel-like substructures do not admit a support tree, as the visibility structure does not allow for positively-spanned vertices without the addition of a cycle, which requires an extra finger. One possible approach to proving this fact is to use some type of polygon decomposition, such as a decomposition that incorporates support graphs. The components of such a decomposition might be ears (regions containing a single convex vertex and a single non-positively-spanned graph vertex); tubes (regions containing only concave vertices and part of an edge of a support graph, with no graph vertices); and intersections (regions with either just concave vertices or no polygon vertices, and positively-spanned graph vertices).

We also conjecture that to immobilize a simple polygon, there must be fingers at the convex vertices, plus one finger per pinwheel-like substructure. If pinwheels are the only class of polygons requiring extra fingers, this second conjecture is clearly true, as a pinwheel requires one extra finger for immobilization (Theorem 2.10).

6.2 Gravity-assisted manipulation

Although we understand how to immobilize cloth with many fingers, we do not have a complete understanding of immobilization in the presence of gravity, where we can rely on the force of gravity to immobilize (or at least constrain) portions of the cloth. The basic analysis in Chapter 3 gives us some ideas on how gravity affects cloth, but it provides no information on what happens in

cases involving buckling, which may turn out to be useful for folding. For example, the three-point, two-finger method demonstrated in this thesis initially puts the cloth in a highly buckled state before using gravity to straighten it.

Once we know how to constrain cloth in various states using gravity, we need to know how to transition between these states to accomplish folding or flattening tasks. An apparently obvious motion may introduce extra undesirable folds, or it may cause existing folds to fall apart. If such transitions involve regrasping, we need a way to transition in both directions between the hanging and flat on a table states (assuming we only have two hands for manipulation).

Finally, we may not be able to predict the full state of cloth during a set of gravity-assisted tasks. How much sensing is necessary, and when do we need it? If the cloth is fully flattened on a table, can we fold it with no sensing, or is sensing necessary (or helpful) at an intermediate phase in the process? For example, during the shaking phases of the three-point, two-finger method for shirt folding, it is helpful to know when the shaking has correctly transitioned the shirt from buckled to hanging, since unnecessary shaking risks dropping the shirt, and increases the time needed for folding.

6.3 Cloth sensing

Some amount of sensing is essential for practical cloth manipulation (particularly flattening) due to the unpredictability of cloth. At a minimum, a few IR sensors can be used to detect the presence of cloth at key locations (such as within the jaws of a gripper). A camera would provide more complete information, although processing the resulting images is very complex. RFID tags or a motion capture system would be capable of providing information on a subset of points on cloth, and it has the advantage of working in the presence of occlusion. A laser rangefinder would provide complete position information for non-occluded regions, although such a system would be very expensive. It is unclear which of these various methods is actually the most useful, although it seems that the RFID method may be a good solution due to its low complexity and cost.

RFID tags have the advantage of being very low cost; one set that we looked at costs $1 per tag. The tags are small 18 mm diameter buttons,and have a fairly limited range, which is beneficial for location sensing. However, as with any RFID system, precision is still limited, with error on the order of 1 to 2 inches.

We did some basic experiments with a Phidgets RFID reader, and determined that with the reader within 0.25 inches of the tags, a single tag can be read with the center of the reader located up to 2 inches away from the center of the tag. With the reader located 0.5 to 1 inch above the tags, a single tag can be read up to 1.5 inches away.

More complex RFID systems have the ability to read multiple tags simultaneously; the Phidgets reader will not report a tag present if multiple tags interfere with each other. However, we can use this to our advantage. Tags can be spaced with their centers 1 inch apart, in which case the reader will correctly identify the tags when it is centered directly over them at a height of 0.5 inches, with no tag registering when the reader is between the tags.

The above results suggest that closely spaced tags actually improve precision, although it is somewhat impractical to try to line the entire boundary of a piece of cloth with RFID tags. It seems that even a limited set of tags at key points on the cloth would provide useful information. An RFID reader can be mounted on a robot arm, which can sweep the reader back and forth over a piece of cloth on a table. From this scan, it should be possible to approximate the locations of the RFID tags on the table's 2D surface, and to estimate the locations of the edges between them.

6.4 Cloth flattening

As we have observed, flattening a piece of cloth is a much harder problem than folding it. It seems very unlikely that there is a sensorless solution to this problem. It may be possible to pick up cloth with no sensing if we make a few assumptions, such as assuming the cloth is in a container (e.g., a laundry hamper) at a known location. However, picking up cloth from an arbitrary point is not useful for flattening or folding; at a minimum, the locations of a few points are needed.

Assuming we have some information about the locations of various points on the cloth, we can attempt to flatten it using various methods. The most obvious attempt is to use a pick and place approach to move key points (such as those needed for an immobilizing grasp) to their locations in a flattened configuration. However, the problem of occlusion still remains. Even if we assume that we can sense occluded points with a method such as RFID tags, we still cannot directly grasp an occluded point. Moving just the top layer of cloth will not in fact move the correct point. We can detect this by ensuring that the gripping mechanism only grasps one layer of cloth, and by sensing if the RFID tag moves along with the gripper. If not, we know that an incorrect layer was moved, and we can take steps to try to remove the occlusion, possibly by hanging the cloth.

A simpler approach is to somehow identify the type of cloth that needs to be flattened, and to flatten it by hanging it from a few fingers. This again has the potential problem of occlusion preventing grasping, in addition to the more complex issue of cloth classification. However, if we have an idea of the shape of the cloth, it should be possible to compute grasps involving a few fingers (ideally two, but some items like button-down dress shirts require more fingers) that will flatten the shirt when it is suspended from the fingers.

One final approach that we briefly experimented with is flattening by spinning. For some types of mostly convex cloth (such as T-shirts, handkerchiefs, or towels), a rotation center exists such that a vector field radiating out from this rotation center flattens the cloth. If we spin the cloth about this point, it will flatten. By combining spinning with an umbrella-like mechanism, it is possible to flatten cloth, and to keep it in the flat configuration after the spinning is stopped.

6.5 Reliable grasping

How can we ensure that we can reliably grasp key points on a piece of cloth (realizing that high precision is not always necessary for manipulation tasks)? In the cloth sensing section above, we propose RFID tags as a possible sensing solution, but it is unclear if they will provide sufficient precision for reliable grasping. In our cloth folding setup in Chapter 3, we solve the problem by attaching magnets to the shirt at key points. Although very reliable, this is undesirable since unfolding the shirt becomes more difficult when portions of it are stuck together with magnets.

6.6 Modifying cloth

Are there ways to modify clothing to make flattening and folding tasks easier? One modification that we have already discussed is embedding RFID tags or magnets in the shirt to assist with grasping points, but we can also consider more radical modifications. Inflating an air mattress in a folded configuration causes the mattress to unfold itself, so if we do something similar and line the entire edge of a piece of clothing with a small flexible air bladder, will the clothing move into a flattened configuration if we inflate the air bladder? What if portions of the surface also have air bladders?

Are there other markers that we can consider? As mentioned in the related work section, Scholz *et al.* use M-arrays to uniquely identify points on cloth [74]. It may not make sense to cover the entire cloth with this pattern, but can we at least cover small regions? Can we incorporate low-level radiation sources that are easily detectable while not harming humans?

Are there built-in sensors that can provide useful information? Would a miniature sensor network be able to provide relative distance information, potentially allowing us to gather some information about configuration? Can we combine light sensors with some other method of identifying key points to only try to grasp points that are not occluded?

6.7 Simulation of string in fixtures

For knot tying, the most critical missing piece is a proper understanding of the behavior of string or wire inside a tube. As discussed in the related work section, there has been significant work done on general string simulation; however, there has been little on simulation within a complex environment such as our knot fixtures. Building such a simulation would require a detailed understanding of the frictional interactions between string and the fixture walls, as well as a better understanding of buckling. With this information, it would become possible to automate the fixture design process for different classes of materials. Additionally, good models of the airflow within air-based fixtures would be useful for designing such fixtures; currently, we build them by hand, which likely introduces problematic turbulent regions.

6.8 New separable fixtures

There are a few potentially interesting separable fixture ideas based on four-piece fixtures. First, given a four-piece fixture, we can force all tubes in the top layer to be horizontal, and all tubes in the bottom layer to be vertical (or vice versa). Any change from horizontal to vertical would require a shaft edge between the two layers. This is possible for any knot, as it is trivial to convert a standard orthogonal graph to this variation. However, this adds a significant number of bends to the fixture, with no clear benefits.

Second, we can convert a four-piece fixture into a three-piece fixture by joining the top and bottom layers into a U-shaped piece, with the join oriented such that both pieces of the middle layer can slide out without interfering with the knot. Extraction and tightening then takes place in the space left by the middle layer. While less elegant than the four-piece version, this has the benefit of slightly reducing the amount of actuation needed to take the fixture apart without introducing any new complications.

6.9 Adding more actuation to knot fixtures

There is also room to make the existing fixtures more practical by adding a greater amount of actuation. Cilia-like manipulators or drive rollers along the entire length of a knot tube would essentially remove the friction issue. Since all points on the string will be moving at the same speed, the string will never start to drag against any sharp curves. Although implementing such a system would be a major engineering project, it should enable arbitrarily complex knot shapes of any length. The algorithm for creating four-piece knot boxes may not currently produce practical fixtures, but with a fully actuated tube, four-piece fixtures could easily serve as a standard, easy to design fixture type.

A simpler method of improving actuation in fixtures is to pull string through a fixture by attaching it to a guide string that is already placed within the fixture's knot tube. This guide string would be long enough that it can be pulled back an equal distance after an insertion, allowing the same guide string to be used repeatedly. This method would only work well in fixtures that do not have many sharp bends, as the friction involved might make it impossible to pull the guide string.

An air-based fixture that is more sensor-based with finer control over air supply should provide better results than the air-based fixtures described in the case studies. Such a system could be modular, as in Section 4.5.7, with a symmetrical ring (and pressure vents) in each segment. A low volume of air can be sent into each segment constantly to try to keep the string away from the walls, with a significantly higher volume of air at a few segments immediately behind a plug attached to the end of the string to assist in moving the plug along. This requires building a complex air control system, along with sensing to determine where the plug is currently located.

Although hard to model and design, one could imagine a system that consists of just air nozzles, with no fixture. A careful arrangement of air nozzles with complex controls should make it possible to tie a knot in mid-air by using highly directed air flows as "fingers". Such a system would require a very extensive understanding of string dynamics, and would likely be very material-specific.

6.10 The future

Is it practical to build a system designed just for laundry folding? Or should such a system be part of a mobile household robot? For large-scale operations, such as towel folding at a hotel, or shirt folding at a shirt factory, it might make sense to have a system dedicated to folding one particular item very quickly and very reliably. However, for home use, we believe that it the average homeowner will not want to buy multiple expensive robot gadgets for different tasks, and instead will want one robot that can vacuum, fold laundry, cook, and so on. This robot may not be as efficient at these tasks as a set of individually tailored robots (in particular, it will have to make do with a generic set of manipulators and sensors), but the versatility will be a major selling point. Given such a system, how can we program it to handle a pile of laundry fresh from the dryer? We need to handle inside-out clothing, multiple different types of clothing, and a huge variety of colors (not to mention static cling).

The ideas discussed in this thesis are applicable at different scales, particularly towards the smaller side. Our immobilization proofs are valid even at the micro scale, and can be applied to lattices of nanotubes, for example. Although there are additional issues such as stiction, knot tying fixtures should work at the micro scale as well. These concepts could lead to new ways of building self-assembling systems, where fixtures attach and shape lattices into desired configurations. Small-scale knot tying seems particularly useful for this purpose. One potentially interesting application might be permanent velcro, where two panels covered in tiny rods are brought together through a middle layer consisting of square knot fixtures, tying the rods on the two panels together. Dissolving the fixture layer would yield a very solid and permanent bond between the two velcro-like layers.

Acknowledgments

This thesis would not have been possible without the help and support of many, many people.

I would like to thank my parents for their love and support; their encouragement throughout my life made it possible for me to end up at a place as wonderful as Dartmouth, and their constant words of advice through the process were invaluable.

My long-distance relationship with my fiancée, Katherine, always served as an excellent motivating factor, and her love and encouragement kept me going through the difficult times. Her faith in me even in the face of failed experiments and stubborn proofs was incredibly sustaining.

I also owe many thanks to my advisor, Devin Balkcom. He proposed knot tying with fixtures as an interesting thought problem at one of our lab meetings, and then helped me turn the first prototype of a curiosity into a full-fledged research project. His insistence that the first cloth immobilization proofs had a bug, despite the belief of myself and others, helped me find polygons that were not immobilized with just convex vertices, and led to a set of proofs that provided a much better understanding of cloth grasping. Thanks for all of the "deep thoughts" over the years!

Thanks go to Yuliy Baryshnikov, Scot Drysdale, Fabio Pellacini, and Daniela Rus for valuable discussions regarding various portions of this thesis, with special thanks to Scot and Daniela for encouraging a high school sophomore to pursue his dreams of getting involved in robotics back in 2000. Our conversations were a major part of my decision to go to Dartmouth. Also, thanks to Tanzeem Choudhury for stepping in at the last minute to rescue my committee.

My labmates (Paritosh, Andrei, Govind, and Anne) were also always a great source of ideas. Thanks for your many thoughts, and for being willing to listen to all those practice talks!

The CS department staff have been tremendously helpful through the years. Joe, Christine, and Kelly were always able to help with any random administrative questions, and the sysadmins (Wayne, Tim, and Sandy) kept my computers running, and put up with my many requests to borrow miscellaneous hardware.

Finally, a huge thank you goes to the Aquinas House community, with a special note of thanks to Caitlin Johnson for proofreading. AQ served as my home away from home, and I will always treasure the many memories and friendships I have from my seemingly never-ending tenure at Dartmouth and AQ. Sadly, it has now come to an end, but you will all remain in my heart forever.

This work was supported in part by NSF grants IIS-0643476 and CNS-0708209.

Bibliography

[1] Oswin Aichholzer, Franz Aurenhammer, David Alberts, and Bernd Gärtner. A novel type of skeleton for polygons. *Journal of Universal Computer Science*, 1(12):752–761, 1995.

[2] All-tribes.info. ALLTRIBES. `http://www.dailymotion.com/video/xmxl_alltribes_news`, January 2006. Available on DailyMotion on Jan 14, 2010.

[3] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54, New York, NY, 1998. ACM.

[4] Matthew Bell and Devin Balkcom. Knot tying with single piece fixtures. In *Proc. IEEE International Conference on Robotics and Automation*, pages 379–384, Pasadena, CA, May 2008.

[5] Matthew Bell and Devin Balkcom. Grasping cloth polygons. Submitted to International Journal of Robotics Research, 2009.

[6] Robert-Paul Berretty, Ken Goldberg, Mark H. Overmars, and A. Frank van der Stappen. Trap design for vibratory bowl feeders. *International Journal of Robotics Research*, 20(11):891–908, 2001.

[7] Antonio Bicchi and Vijay Kumar. Robotic grasping and contact: A review. In *IEEE International Conference on Robotics and Automation*, pages 348–353, San Francisco, CA, April 2000.

[8] Iain M. Boyle, Kevin Rong, and David C. Brown. Cafixd: A case-based reasoning fixture design method. framework and indexing mechanisms. *Journal of Computing and Information Science in Engineering*, 6(1):40–48, March 2006.

[9] David E. Breen, Donald H. House, and Michael J. Wozny. A particle-based model for simulating the draping behavior of woven cloth. *Textile Research Journal*, 64(11):663–685, November 1994.

[10] R. C. Brost and K. Y. Goldberg. A complete algorithm for designing planar fixtures using modular components. *IEEE Trans. Robot. Autom.*, 12(1):31–46, February 1996.

[11] Joel W. Burdick, J. Radford, and Gregory S. Chirikjian. A "sidewinding" locomotion gait for hyper-redundant robots. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 101–106, Atlanta, GA, May 1993.

[12] James Burns and Andrew Fung. Shoelace knot assisting device. US Patent No. 7044508, May 2006.

[13] CGAL, Computational Geometry Algorithms Library, v3.3.1, 2008. http://www.cgal.org.

[14] Mark Champion. Knot tying device. US Patent No. 6817634, November 2004.

[15] Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete and Computational Geometry*, 6(1):485–524, December 1991.

[16] 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.

[17] 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.

[18] Jae-Sook Cheong, A. Frank van der Stappen, Kenneth Y. Goldberg, Mark H. Overmars, and Elon Rimon. Immobilizing hinged polygons. *International Journal of Computational Geometry and Applications*, 17(1):45–70, February 2007.

[19] Kwang-Jin Choi and Hyeong-Seok Ko. Stable but responsive cloth. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 604–611, San Antonio, TX, 2002. ACM Press.

[20] Robert Connelly. Tensegrity stuctures: Why are they stable? In M. F. Thorpe and P. M. Duxbury, editors, *Rigidity Theory and Applications*, pages 47–54. Kluwer/Plenum Publishers, New York, NY, 1999.

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

[22] Chandler Davis. Theory of positive linear dependence. *American Journal of Mathematics*, 76(4):733–746, October 1954.

[23] Jeff Erickson, Shripad Thite, Fred Rothganger, and Jean Ponce. Capturing a convex object with three discs. In *IEEE International Conference on Robotics and Automation*, pages 2242–2247, Taipei, Taiwan, September 2003.

[24] Jeff Erickson, Shripad Thite, Fred Rothganger, and Jean Ponce. Capturing a convex object with three discs. *IEEE Transactions on Robotics*, 23(6):1133–1140, December 2007.

[25] Carl Richard Feynman. Modeling the appearance of cloth. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 1987.

[26] Steve Fisk. A short proof of Chvátal's Watchman Theorem. *Journal of Combinatorial Theory, Series B*, 24(3):374, 1978.

[27] 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.

[28] Subir Kumar Ghosh. *Visibility Algorithms in the Plane*. Cambridge University Press, New York, NY, 2007.

[29] K. "Gopal" Gopalakrishnan and Ken Goldberg. D-Space and deform closure grasps of deformable parts. *International Journal of Robotics Research*, 24(11):899–910, November 2005.

[30] Kyoko Hamajima and Masayoshi Kakikura. Planning strategy for task of unfolding clothes. *Robotics and Autonomous Systems*, 32(2–3):145–152, August 2000.

[31] Joel Hass and Jeffrey C. Lagarias. The number of Reidemeister moves needed for unknotting. *J. Amer. Math. Soc.*, 14(2):399–428, 2001.

[32] Shinichi Hirai, Tatsuhiko Tsuboi, and Takahiro Wada. Robust grasping manipulation of deformable objects. In *IEEE International Symposium on Assembly and Task Planning*, pages 411–416, May 2001.

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

[34] Ayanna M. Howard and George A. Bekey. Recursive learning for deformable object manipulation. In *International Conference on Advanced Robotics*, pages 939–944, Monterey, CA, July 1997.

[35] Robert J. Webster III, Jin Seob Kim, Noah J. Cowan, Gregory S. Chirikjian, and Allison M. Okamura. Nonholonomic model of needle steering. *International Journal of Robotics Research*, 25(5–6):509–525, 2006.

[36] Hirochika Inoue and Masayuki Inaba. Hand-eye coordination in rope handling. In *Robotics Research: The First International Symposium*, pages 163–174, 1985.

[37] Japanese way of folding t-shirts! `http://www.youtube.com/watch?v=b5AWQ5aBjgE`, June 2006. Available on YouTube on Jan 14, 2010.

[38] Yan-Bin Jia. Computation on parametric curves with an application in grasping. *International Journal of Robotics Research*, 23(7–8):827–857, 2004.

[39] J. Kahn, M. Klawe, and D. Kleitman. Traditional galleries require fewer watchmen. *SIAM Journal on Algebraic and Discrete Methods*, 4(2):194–206, June 1983.

[40] Manabu Kaneko and Masayoshi Kakikura. Planning strategy for putting away laundry - isolating and unfolding task. In *IEEE International Symposium on Assembly and Task Planning*, pages 429–434, Fukuoka, Japan, May 2001.

[41] 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.

[42] 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.

[43] Yasuyo Kita and Nobuyuki Kita. A model-driven method of estimating the state of clothes for manipulating it. In *WACV '02: Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision*, pages 63–69, Washington, DC, USA, December 2002. IEEE Computer Society.

[44] Yasuyo Kita, Fuminori Saito, and Nobuyuki Kita. A deformable model driven visual method for handling clothes. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 3889–3895, New Orleans, LA, May 2004.

[45] Wlodzimierz Kuperberg. Problems on polytopes and convex sets. In *Proc. DIMACS Workshop on Polytopes*, pages 584–589, Rutgers University, January 1990.

[46] Charles Livingston. *Knot Theory*. The Mathematical Association of America, 1993.

[47] Liang Lu and Srinivas Akella. Folding cartons with fixtures: A motion planning approach. In *IEEE International Conference on Robotics and Automation*, May 1999.

[48] Liang Lu and Srinivas Akella. Folding cartons with fixtures: A motion planning approach. *IEEE Trans. Robot. Autom.*, 16(4):346–356, August 2000.

[49] Bhubaneswar Mishra, Jacob T. Schwartz, and Micha Sharir. On the existence and synthesis of multifinger positive grips. *Algorithmica*, 2(4):541–558, November 1987.

[50] Van-Duc Nguyen. Constructing stable force-closure grasps. In *Proceedings of 1986 ACM Fall joint computer conference*, pages 129–137, 1986.

[51] Shigeru Nishikawa, Haruo Niwaya, Atsuo Shibuya, and Noboru Aisaka. Measuring of surface shapes of cloths by ultrasonic sensor. *Journal of the Textile Machinery Society of Japan*, 34(4):111–115, 1988.

[52] Karl J. Obermeyer. The VisiLibity library. http://www.VisiLibity.org, 2008. R-1.

[53] Eiichi Ono, H. Ichijo, and N. Aisaka. Robot hand for handling cloth. In *International Conference on Advanced Robotics*, volume 1, pages 769–774, Pisa, Italy, June 1991.

[54] Eiichi Ono, Noboyuki Kita, and Shigeyuki Sakane. Strategy for unfolding a fabric piece by cooperative sensing of touch and vision. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 441–445, Pittsburgh, PA, August 1995.

[55] Joseph O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, New York, NY, 1987.

[56] Fumiaki Osawa, Hiroaki Seki, and Yoshitsugu Kamiya. Clothes folding task by tool-using robot. *Journal of Robotics and Mechatronics*, 18(5):618–625, 2006.

[57] Dinesh K. Pai. STRANDS: Interactive simulation of thin solids using Cosserat models. *Computer Graphics Forum*, 21(3):347–352, 2002.

[58] R. Patton, F. Swern, S. Tricamo, and A. van der Veen. Automated cloth handling using adaptive force feedback. *Journal of Dynamic Systems, Measurement, and Control*, 114(4):731–733, 1992.

[59] Jeff Phillips, Andrew M. Ladd, and Lydia E. Kavraki. Simulated knot tying. In *Proc. IEEE International Conference on Robotics and Automation*, pages 841–846, May 2002.

[60] Franco P. Preparata. The medial axis of a simple polygon. In *Mathematical Foundations of Computer Science*, volume 53, pages 443–450, Tatranska Lomnica, Czechoslovakia, September 1977.

[61] David Pritchard and Wolfgang Heidrich. Cloth motion capture. In *Eurographics*, pages 263–271, September 2003.

[62] B. Ravi and M. N. Srinivasan. Decision criteria for computer-aided parting surface design. *Computer-Aided Design*, 22(1):11–18, 1990.

[63] Kurt Reidemeister. Knotten und gruppen. *Abhandlungen aus dem mathematischen Seminar der Hamburgischen Universität*, 5:7–23, 1927.

[64] Franz Reuleaux. *The Kinematics of Machinery*. MacMillan, 1876. Reprinted by Dover, 1963.

[65] 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.

[66] 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.

[67] 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.

[68] 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.

[69] Mitul Saha and Pekka Isto. Manipulation planning for deformable linear objects. *IEEE Trans. Robot.*, 23(6):1141–1150, December 2007.

[70] Mitul Saha, Pekka Isto, and J.-C. Latombe. Motion planning for robotic knot tying. In *Proc. International Symposium on Experimental Robotics*, July 2006.

[71] Khairul Salleh, Hiroaki Seki, Yoshitsugu Kamiya, and Masatoshi Hikizu. Spreading of clothes by robot arms using tracing method. In *Proc. 5th International Conference on Machine Automation*, pages 63–68, Osaka, Japan, November 2004.

[72] Khairul Salleh, Hiroaki Seki, Yoshitsugu Kamiya, and Masatoshi Hikizu. Inchworm robot grippers for clothes manipulation. *Artificial Life and Robotics*, 12(1–2):142–147, March 2008.

[73] Volker Scholz and Marcus Magnor. Cloth motion capture from optical flow. In *Proc. of 9th International Fall Workshop on Vision, Modeling and Visualization*, Stanford, CA, November 2004.

[74] Volker Scholz, Timo Stich, Michael Keckeisen, Markus Wacker, and Marcus Magnor. Garment motion capture using color-coded patterns. *Computer Graphics Forum*, 24(3):439–447, October 2005.

[75] Mizuho Shibata, Tsuyoshi Ota, Yoshimasa Endo, and Shinichi Hirai. Handling of hemmed fabrics by a single-armed robot. In *IEEE International Conference on Automation Science and Engineering*, pages 882–887, Washington, DC, August 2008.

[76] Xiangyang Shu, Han Ding, and Jun Wang. Grasp analysis and synthesis based on a new quantitative measure. *IEEE Transactions on Robotics and Automation*, 19(6):942–953, December 2003.

[77] Wamis Singhatat. Intracorporeal knot tier. US Patent No. 6716224, April 2004.

[78] Jongchul Song, Carl T. Haas, and Carlos H. Caldas. A proximity-based method for locating RFID tagged objects. *Advanced Engineering Informatics*, 21(4):367 – 376, 2007.

[79] J. Takamatsu, T. Morita, K. Ogawara, H. Kimura, and K. Ikeuchi. Representation for knot-tying tasks. In *Proc. IEEE International Conference on Robotics and Automation*, volume 22, pages 65–78, February 2006.

[80] Roberto Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444, 1987.

[81] R. H. Taylor and D. Stoianovici. Medical robotics in computer-integrated surgery. *IEEE Trans. Robot. Autom.*, 19(5):765–781, October 2003.

[82] Hidefumi Wakamatsu, Eiji Arai, and Shinichi Hirai. Knotting/unknotting manipulation of deformable linear objects. *International Journal of Robotics Research*, 25:371–395, 2006.

[83] Hidefumi Wakamatsu and Shinichi Hirai. Static modeling of linear object deformation based on differential geometry. *International Journal of Robotics Research*, 23(3):293–311, March 2004.

[84] Hidefumi Wakamatsu, Shinichi Hirai, and Kazuaki Iwata. Static analysis of deformable object grasping based on bounded force closure. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 3324–3329, Minneapolis, MN, April 1996.

[85] 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.

[86] Jerry Weil. The synthesis of cloth objects. *SIGGRAPH Computer Graphics*, 20(4):49–54, August 1986.

[87] Eric W. Weisstein. *CRC Concise Encyclopedia of Mathematics*. Chapman & Hall/CRC, 2nd edition, 2003.

[88] Cornell Wright, Aaron Johnson, Aaron Peck, Zachary McCord, Allison Naaktgeboren, Philip Gianfortoni, Manuel Gonzalez-Rivero, Ross, and Howie Choset. Design of a modular snake robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2603–2608, San Diego, CA, October 2007.