Grasping and folding knots

Weifu Wang Devin Balkcom

***Abstract*— String stretched tightly along a sequence of fixed grasp points takes the shape of a polygonal arc. In this work, we investigate how many points are necessary and sufficient to grasp and tie arbitrary knots while maintaining tension, so that the string remains polygonal. This approach allows reasoning that is entirely geometric, which does not rely on potentially inaccurate dynamic models of the string or detailed knowledge of physical characteristics of the string. Algorithms are proposed to determine the contact locations, and generate the motions needed to tie arbitrary knots. This work shows that a number of grasp points that is linear in the number of crossings in a knot diagram is sufficient to immobilize string in a polygonal shape with the topology of an arbitrary knot, or to fold or unfold the knot from a straight configuration.**

I. Introduction

In the game of cat's cradle, string is stretched tightly between fingers; as the fingers move, the configuration of the string changes. In this work, we study how many grasp points (analogous to fingers) are necessary and sufficient to tie a given knot, while maintaining complete control over the configuration of the string. We assume that all contacts are bilateral and have infinite friction.

The theoretical complexity of the knot tying process has begun to be studied from the perspective of topological structure (*e.g.*, sequences of *Reidemeister moves* [8]), but the details of the manipulation itself are left to the imagination.

How should knots be grasped? Work on robotic knot tying [9], [17], [18] has shown that knots can be practically tied with only a few fingers by making use of dynamic or quasistatic models of string and extensive sensing; however, speed, reliability, and the complexity of knots that can be tied are limited by modeling and sensing error. By forcing the string to take on a polygonal shape, we are able to study complexity, while also providing specific finger placements and motions to achieve grasping and tying.

Tying a knot requires that the string be *arranged* into the correct shape and also *tightened* [22], [23]. In this paper, we focus on arrangement; because we ensure that the string remains polygonal, we refer to the arrangement process as *folding*.

The work described in this paper is theoretical rather than experimental. We show the necessary and sufficient number of contacts to either hold or to fold a knot without string self-intersection. For an arbitrary polygonal knot with k crossings, it will turn out that at least $\left\lceil \frac{3+\sqrt{8k+1}}{2} \right\rceil + 1$ and no more than $3k-1$ grasp points are needed to grasp string in a polygonal knot configuration. Further, an arbitrary knot with Gauss code G where $|G| = 2k$ can always be folded from a polygonal arc of no more than $9k-6$ links, starting from a straight configuration. We give algorithms to determine the contact locations along the string, and to generate folding motions.

As a motivating example, Figure 1 shows several immobilized polygonal knots. The double coin knot, for example, is immobilized by 16 contacts placed by our approach.

In this paper, after a short introduction of necessary knot definitions, we will consider grasps for arbitrary knots. Then, we will show how to fold string into polygonal knotted configurations.

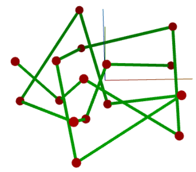
A. Related work

This paper makes use of the language of knot theory [1], [3], [7], [11], [12]. Within knot theory, the study of polygonal knots does not investigate the number of line segments needed to represent the knot, the focus of this work. The problems studied in this paper are also very related to a three-dimensional version of the *Carpenter's Rule Problem* [6], except that we allow joint locations (finger placements) to be specified. *Physical knot theory* [10] has studied the problem of tying knots modeled by thick string with friction rather than by curves.

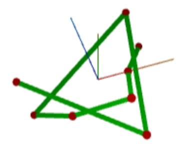
In robotics, knot tying frequently uses active controls and sensors. Inoue and Inaba used a 6+1 DOF robot arm equipped with stereo machine vision to tie knots [9] around a ring. A graph based language was developed by Hopcroft *et al.*to program knot tying motions, and tested through knot tying with a robot arm [8].

Real and simulated knots were tied using motion planning algorithms developed by Saha, Isto and Latombe with a string model [17], [19], [18]. Sequences of motions were planned and carried out by a pair of robot arms. The approach introduced a hierarchical decomposition of the knot using loops as basic structure.

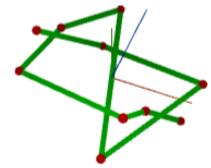
(a) A double coin knot.



(b) A double coin knot represented by polygonal arc.



(c) An overhand knot represented by polygonal arc.



(d) A figure eight knot represented by polygonal arc.

Fig. 1: Polygonal grasps of various knots.

Wakamatsu, Arai and Hirai's tree search planner finds sequences of motions (selected from four primitives) to tie or untie a knot [20].

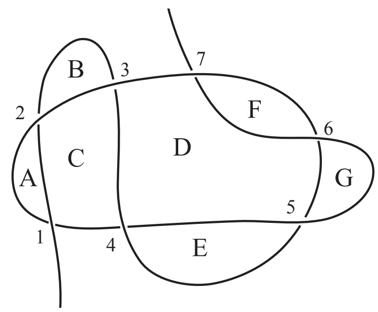
Recently, we started to studying how to tie knots with the assistance of fixtures [4], [5], [22], [23]. The approach resulted in a generalized knot tying approach, that can tie many different knots effectively using minimum control and no feedback information.

The approach we take in this work immobilizes each segment of string at different configurations to reduce uncertainty, just as rigid bodies are immobilized during grasping [13], [14], [15], [16].

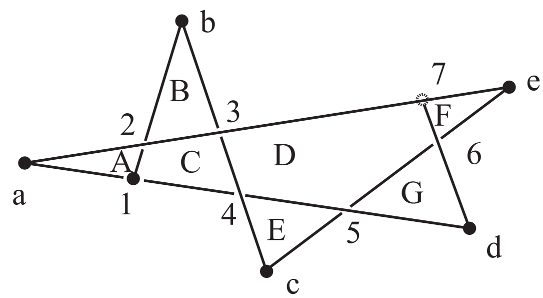
II. Polygonal knots

In mathematical knot theory, a knot is a smooth embedding of topological circle into $\mathbb{R}^3$ [1], [3], [7], [11], [12]. These knots have no open ends, such that they cannot be tied or untied. Here, we define *tie a knot* as the process of transforming the string from unknotted configuration to a knotted configuration. Because we are concerned with reconfiguring knots, and not just identifying classes of knots, we cut mathematical knots so that they have ends and can be untied.

To describe or illustrate a knot, we usually project a knot onto a plane. When the projection is regular (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 [11]), the projected diagram is called a *knot diagram*. On the projection, broken lines are used to indicate where one part of the knot under-crosses the other part of the knot that is directly "above" the broken lines, such locations are called crossings. Each crossing is labeled by a unique number, indicating the order of the appearance when tracing along the digram. Therefore, each knot diagram may have multiple different ways to label the same set of crossings. Figure 2a shows a shoelace knot diagram. The projected knot diagram separates the plane into several disconnected closed *cells*, labeled by capital letters in the figure.



(a) A knot diagram of a shoelace knot, with numbered crossings and cells labeled with letters.



(b) A polygonal (grasp-able) shoelace knot diagram.

Fig. 2: Shoelace knot diagrams.

We can also use the labels of the crossings to describe the knot. One such description is the *Gauss code*: a sequence of labels for crossings indicating a walk along the diagram from a given starting point. We use numbers to label crossings, and a superscript "+" or "-" to indicate an over-crossing or an under-crossing. For example, an overhand knot with Gauss code $G = \{1^+, 2^-, 3^+, 1^-, 2^+, 3^-\}$ has 3 crossings, so the length of G is 6 ($|G| = 6$).

Definition 1: [3] A polygonal knot is a knot whose image in $\mathbb{R}^3$ is the union of a finite set of line segments.

In this work, we focus on knots that are *tame*, who has finite many crossings. A regular projection of a polygonal knot is a set of planar line segments. We call the corresponding knot diagram *polygonal knot diagram* (Figure 2b). If a polygonal knot diagram contains n links, in addition to the Gauss code, the knot can be fully represented by the relative positions between the end points of these n links.

A polygonal knot diagram with k crossings is a 4-planar graph $G = (V, E)$ with straight line edges, and $|V| \geq k$. On the regular projection, we trim the first and the last link of the polygonal arc, so that the first and last crossings become the endpoints. $k-2$ of the vertices (the crossings) have degree four, and two vertices have degree three; the remainder have degree 2.

III. Properties of polygonal knot diagrams

Call the cell that extends to infinity the *exterior cell*, and all other cells *interior cells*. A sequence of one or more line segments connecting two adjacent labels on the Gauss code is a *c-path* (crossing path).

We call a c-path that contacts both the exterior face and at least one interior face an *exterior c-path* (c-paths $(1,2)$ through a, $(2,3)$ through b, $(3,7)$, $(7,6)$ through e, $(6,5)$ through d, $(5,4)$ through c and $(4,1)$ on Figure 2b). Let an *exterior crossing* be a crossing that is the common endpoint of two adjacent exterior c-paths; all the other crossings are *interior crossings*. If a connection between two crossings is not an exterior c-path, it is called an *interior c-path*.

Each crossing appears twice on a Gauss code for arbitrary knot. Let us define a *knot unit* as a connected subsection of Gauss code where all the labels in the subsection has appeared twice. For example, let us consider a three overhand knots in a row, with Gauss code $G = \{1^+, 2^-, 3^+, 1^-, 2^+, 3^-, 4^+, 5^-, 6^+, 4^-, 5^+, 6^-, 7^+, 8^-, 9^+, 7^-, 8^+, 9^-\}$. This structure contains three knot units, $G_1 = \{1^+, 2^-, 3^+, 1^-, 2^+, 3^-\}$, $G_2 = \{4^+, 5^-, 6^+, 4^-, 5^+, 6^-\}$ and $G_3 = \{7^+, 8^-, 9^+, 7^-, 8^+, 9^-\}$. We call a c-path a *bridge* if it connects two knot units.

Some proofs are omitted or replaced by sketches below due to length constraints, but can be found in the associated technical report [21] available from the authors' website.

Lemma 1: All the exterior c-paths of a knot unit form a Jordan curve.

Sketch of proof: Since a knot unit do not contain a bridge, then an edge is either adjacent to two interior cells, or one interior and the exterior cell. All the edges that are adjacent to one interior and the exterior cell encloses all other edges.

Lemma 2: A planar polygonal knot diagram with k crossings has k interior faces.

Proof: Let there be $m+k$ vertices in the planar polygonal drawing of the knot, such that there are k crossings, with degree 4 or 3, and m other vertices with degree 2. By counting edges, we find that there are $m + 2k - 1$ edges. Let the number of faces be $|f|$. By Euler's theorem, $|v| + |f| = |e| + 2$, and $|f| = m + 2k - 1 + 2 - (m+k) = k+1$ faces, k of which are interior. ∎

Lemma 3: For an arbitrary knot with Gauss code G where $|G| = 2k$, let there be n_{ic-p} interior c-paths, n_{ec-p} exterior c-paths, n_{ic} interior crossings and n_{ec} exterior crossing. Then, the following equations and inequalities hold:

$$n_{ic} = n_{ic-p} - (k-1) \tag{1}$$

$$n_{ec} = n_{ec-p} \tag{2}$$

$$n_{ic} + n_{ec} = k \tag{3}$$

$$n_{ic-p} + n_{ec-p} = 2k - 1 \tag{4}$$

$$n_{ec-p} \geq 3 \tag{5}$$

$$k - 1 \leq n_{ic-p} \leq 2k - 4 \tag{6}$$

Sketch of proof: The total number of crossings and c-paths are fixed. Each exterior crossing is connected to two exterior c-paths. Using this relation, all the equations can be derived.

IV. Grasping knots

Given a Gauss code G, we would like to generate an efficient polygonal knot diagram, corresponding to a grasp of the knot. The number of grasp points (fingers) needed is related to the number of crossings, which can be found from the Gauss code:

Lemma 4: For an arbitrary knot with k crossings, at least $\left\lceil \frac{3+\sqrt{8k+1}}{2} \right\rceil$ sequential line segments are needed to plot the corresponding polygonal knot diagram.

Proof: [2] indicates that n independent line segments (not sequential line segments) can create at most $n(n-1)/2$ intersections. However, for a sequence of n line segments (polygonal arc), there is one common endpoints between each pair of adjacent line segments, which results in $n-1$ common endpoints. Therefore, the maximum number of intersections for a sequence of n line segments is $n(n-1)/2 - n - 1$ or $(n-1)(n-2)/2$.

Given n sequential line segments, at most $(n-1)(n-2)/2$ intersections (crossings) can be created. In order to make it a knot diagram, we need to have $(n-1)(n-2)/2 \geq k$. Therefore $n \geq \left\lceil \frac{3+\sqrt{8k+1}}{2} \right\rceil$. ∎

Lemma 4 implies a necessary number of grasp points (fingers) needed to polygonalize a knot. What is a sufficient number? We introduce an algorithm to place all the crossings and extra vertices, starting from exterior crossings and move inwards. We will show that by connecting all the placed crossings (and extra vertices) using line segments, no intersection (except at common end points) is introduced.

We represent a crossing using a lower case letter. If there is a superscript $^+$ or $^-$, it specifies an over- or under-crossing. A negative sign before a crossing notation negates the superscript. For example, an over-crossing can be represented as i^+, or $-i^-$.

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

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

Sketch of proof: A crossing e is result of the intersection of two curves. Among the four half-planes separated by the two curve, only one of them is the exterior cell, which must contact both curves.

Without loss of generality, let the first crossing be 1^a and last crossing be l^b.

Lemma 6: The two c-paths with -1^a as endpoints and two c-paths with $-l^b$ as endpoints are exterior c-paths.

Sketch of proof: This is the special case of Lemma 5 with the first and last crossing. The trimming of the link before first crossing (or after the last crossing) is to ensure there is no link contacting only the exterior cell. Then, it leaves the curve passing through -1^a (or $-l^b$) as the boundary.

Now that we have ways to identify exterior c-paths and know two exterior crossings, we can identify all the exterior crossings using the following procedure. Procedure 1 shows the process of finding the exterior crossings.

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

We find that 1^- and 5^+ are related to the exterior c-paths, add $(4^-, 5^+)$, $(5^+, 6^-)$, $(6^+, 1^-)$ and $(1^-, 7^+)$ to the exterior c-path list, and put 4^-, 6^+, 6^- and 7^- in the queue. Since 6 has appeared twice, delete it from the queue. Now start with crossing 4. Note that 4^+ has previous crossing 7^- and next crossing 8^- in the Gauss code. We find that 7 is already in the queue, add c-path $(7^-, 4^+)$ to the exterior c-path list, delete 7 from the queue, and notice the queue is empty; terminate.

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

Procedure 1: Find exterior crossings and c-paths

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

Procedure 2: Place all exterior crossings

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

Lemma 7: If all exterior crossings are placed using

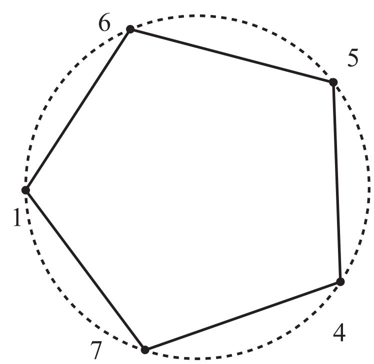


Fig. 3: Place all exterior crossings for double coin knot.

Procedure 2, then line segments connecting exterior crossings that are adjacent on the Gauss code do not intersect except at common endpoints.

Proof: All exterior c-paths form a regular convex polygon P. Because no chord of P will intersect the boundary except at endpoints, no exterior c-path will intersect with non-exterior c-paths between exterior crossings.

Now we will show that chords formed by connecting exterior crossings that are adjacent in the Gauss code do not intersect except at common endpoints. Consider four exterior crossings a, b, c, d, such that a and c are adjacent in the Gauss code, and b and d are adjacent in the Gauss code. Line segment ac cuts P into two pieces. If b and d belong to different pieces of P w.r.t. ac, then the only way to avoid intersection between ac and bd using any arbitrary curve is if b and d are connected outside P. But c-path $(b,\ d)$ is not an exterior c-path; therefore to be consistent with the Gauss code, b and d must be in the same piece. ■

So far, we have placed all the exterior crossings and connected them if they are adjacent on the Gauss code. So far, no undesired intersection have been introduced. An example of placing all exterior crossings and connecting them is shown in Figure 3. Now we will work inwards, starting by placing all the crossings that are directly connected to the exterior crossings.

Figure 4 demonstrates step three. Since v connects to a, b and c (or even d), the wedge with oa and ob as boundary contains all the crossings v connects to. Place v on the smaller circle with radius $1/2$.

Lemma 8: If $v \in V$ is placed using Procedure 3, and all $v \in V$ are connected to to their corresponding exterior crossings using line segments, there exists an $\varepsilon \geq 0$ perturbation of each v along ov away from o such that these c-paths do not intersect with each other or with previously connected c-paths.

Procedure 3: Place interior crossings

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

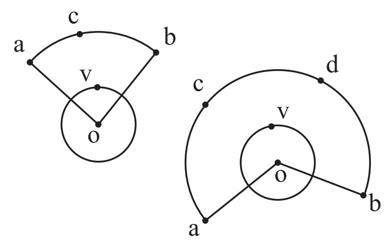


Fig. 4: Finding placement for v based on the exterior crossings it connects to.

Sketch of proof: As in the proof of Lemma 7, separate the polygon into two pieces. Show that either the endpoints of the interior c-paths belongs to one piece, or the c-paths are inconsistent with the Gauss code. (The ε perturbation allows a case where placing v on the smaller circle separates v from its adjacent exterior crossings to be avoided.)

An example of connecting all crossings in V to exterior crossings for the double coin knot is shown in Figure 5.

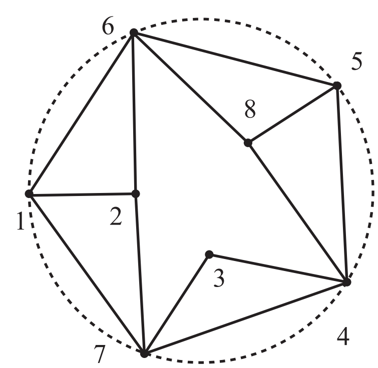


Fig. 5: Connect crossings in V with the exterior crossings based on the Gauss code.

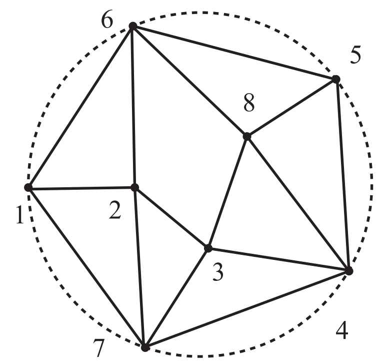


Fig. 6: Connect crossings in V to other crossings in V for double coin knot.

We now connect the crossings within V. We will show that these c-paths does not create intersections.

Let us consider $v_i \in V$ for $i = \{1,2,3,4\}$ and the case where v_3 and v_4 belong to different pieces inside of P, w.r.t. a polygonal arc (a, v_1, v_2, b) connecting two exterior crossings a and b and passing through v_1 and v_2. Since v_3 and v_4 are both connected to exterior crossings as well (otherwise they will not be in V), then consider a polygonal arc (c, v_3, v_4, d) connecting exterior crossings c and d which passes through v_3 and v_4. These two exterior crossings c and d must also belong to two different pieces w.r.t. (a, v_1, v_2, b) because all v are placed adjacent to the exterior crossings they are connected to. Then, the only way to avoid intersection between (a, v_1, v_2, b) and (c, v_3, v_4, d) is that one of the two polygonal arcs is outside P, and neither is. Therefore, v_3 and v_4 must belong to the same piece w.r.t. (a, v_1, v_2, b).

The final laid-out polygonal knot diagram for the double coin knot is shown in Figure 6. No extra intersections are introduced, and 15 line segments are used.

If there are still crossings not placed, recursively find crossings V_i that direct connect to crossings in V_{i-1} where $V_0 = V$, and place them on the circle with radius $1/2^{n+1}$, perturbing slightly if needed. Analogously to the previous argument, these connections will not intersect with each other.

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

Proof: Place all the crossings using Procedures 2 to 3, connecting all of them with line segments. Each of the k crossings is visited twice, so $2k-1$ line segments are used.

However, two crossings a and b may be adjacent twice in the Gauss code, so that the two line segments connecting them overlap. For example, an overhand knot with Gauss code $G = \{1^+, 2^-, 3^+, 1^-, 2^+, 3^-\}$, crossings 1 and 2 appear adjacent to each other twice in G.

For the second time the two crossings a and b are adjacent in the Gauss code, use two line segments instead of one to connect them. Denote the midpoint of ab as $m_{a,b}$. For the second time a connects to b, if it is an exterior c-path, place p at $(1+\varepsilon)o\vec{m}_{a,b}$, otherwise, place p at $(1-\varepsilon)o\vec{m}_{a,b}$, where $\varepsilon > 0$.

With k crossings overlapping may occur at most $k-1$ times. Therefore, at most $2k-1+k-1 = 3k-2$ line segments are necessary to draw any planar polygonal knot diagram. ■

Combining Lemma 4 and Lemma 9, we have,

Theorem 10: For an arbitrary knot with Gauss code G where $|G| = 2k$, to draw a polygonal knot diagram of this knot, at least $\left\lceil \frac{3+\sqrt{8k+1}}{2} \right\rceil$ and no more than $3k-2$ sequential line segments are required.

Then, for the polygonal knot itself, we have:

Theorem 11: For arbitrary polygonal knot with k crossings, at least $\left\lceil \frac{3+\sqrt{8k+1}}{2} \right\rceil$ and no more than $3k-2$ sequential line segments are required to represent the polygonal knot.

Proof: In order to project to n line segments on the plane, at least n line segments are needed in 3D. In the polygonal knot diagram constructed in the proof of Lemma 9, all the crossing are the vertices of the projected planar graph. This, combined with fact that the knot diagram is a regular projection, implies that the same number of projected line segments is sufficient to represent the 3D polygonal knot.

Place a crossing with coordinates (x_i, y_i) in the knot diagram at $(x_i, y_i, 1)$ if it is an over crossing, and at $(x_i, y_i, -1)$ if it is an under crossing. If a vertex on the polygonal knot diagram has degree 2, place it at $(x_i, y_i, 0)$. Since the projected line segments do not intersect, the original 3D line segments also do not intersect. Therefore, no more than $3k-2$ line segments are needed to describe an arbitrary polygonal knot with k crossings. ■

Since each polygonal arc with n links has $n+1$ end points, we have the following corollary.

Corollary 1: At least $\left\lceil \frac{3+\sqrt{8k+1}}{2} \right\rceil + 1$ and no more than $3k-1$ grasp points are needed to grasp an arbitrary knot with k crossings in a polygonal knot configuration.

V. Folding knots

Now we investigate the problem of folding a polygonal arc from a straight line configuration to a knotted configuration. Any lower bound for the number of line segments needed to represent the knot is also a necessary number of line segments needed to fold the knot. What is a sufficient number of line segments to fold the knot?

In this section, we say a polygonal knot K_p is approximated by a polygonal arc P if the projection of P on the x-y plane is a polygonal knot diagram of K_p. A link l on K_p is approximated by a link l' on P if the projection of l' overlaps with the projection of l on the x-y plane.

Theorem 12: An arbitrary knot with Gauss code G where $|G| = 2k$ can always be folded from a polygonal arc of no more than $9k-6$ links, starting from a straight configuration.

Proof: Let us use a polygonal arc P to approximate the configuration of a polygonal knot K_p constructed in Theorem 11. Let the two endpoints on the K_p be v_0 and v_n.

We will approximate as follows. For two vertices v_i and v_j on K_p, if $i < j$, then let v_i appear earlier than v_j when traversing along the K_p from endpoint v_0. Once the links up to l_j ($p_{j-1}p_j$) on P have approximated the links up to v_iv_{i+1} on K_p, then all vertices before p_j on P are fixed. When approximating a link v_iv_{i+1} using links from vertex p_a to p_b on P ($a < b$), it is possible to ensure that p_a and v_i, p_b and v_{i+1} have same coordinates; we will do so. Initially, for the ith link (l_i) on P between vertex p_{i-1} and p_i for which $1 \le i \le n$, let the length of the link $d(p_{i-1}, p_i)$ ($d(l_i)$) equal the distance between v_{i-1} and v_i, the $i-1$th and ith vertex on K_p, $d(v_{i-1}, v_i)$.

Now fold, starting with the first link of the polygonal knot, v_0v_1. Without loss of generality, let v_1 be an over-crossing. We use the link p_0p_1 on P to approximate the first link of the knot, while keeping all the links from p_1 to p_m ($m \ge n$) above the plane $z = z(v_1)$. Then approximate v_1v_2 using link p_1p_2, by rotating all the links following p_1p_2 around p_1 until p_2 has the same coordinates as v_2 and all other vertices are below plane $z = z(v_2)$. Then, repeat this process if such a rotation around p_i does not collide with the fixed links (we will consider the collision case next). After approximating an over-crossing (under-crossing) v_i using links before vertex p_j on P, all vertices after p_j will be above (under) the plane $z = z(v_i)$.

When approximating a vertex v_i (where all vertices before p_j are used to approximate vertices up to v_{i-1}, without loss of generality, let v_i be an under-crossing and v_{i-1} be an over-crossing), since all the previous links are fixed in space, it is possible that their projection forms a loop on x-y plane such that the previous attempted rotation around p_j will collide with the fixed links. Let v'_{i-1} and v'_i denote the projected points on x-y plane, and

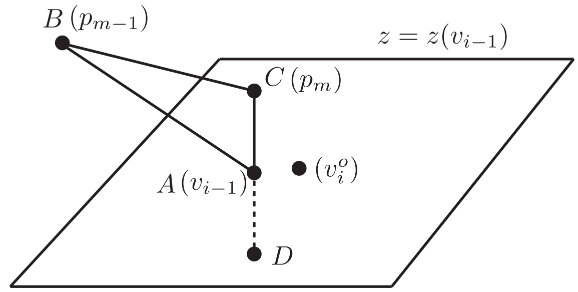


Fig. 7: Folding links given arbitrary small clearance.

the crossing over v_i (denote as v_i^o) has been placed, we have $d(p_j, p_{j+1}) \ge d(v'_{i-1}, v'_i)$.

In this situation, insert two additional links l^1_{j+1} and l^2_{j+1} before link l_{j+2} and after link l_{j+1}, pointing from p_j to p_{j+1} (recall that links l_1 to l_i are fixed, and we need to move l_{i+1} and all links after it to let p_{j+1} occupy the same location as v_i), and change the length of l_{i+1}. Let $d(l^1_{j+1}) = d(l_{j+1}) \ge \max(l^a_b)$ where $b \ge j+2$ and $a = \{0,1,2\}$ ($l^0_i = l_i$), and let $l^2_{j+1} = \sqrt{d(v_i, v_i^o)^2 + \varepsilon^2}$. For arbitrary small $d(v'_{i-1}, v'_i)$, every link l^a_b and its previous link can rotate around one of the two endpoints with the smaller index such that all vertices after p_a (including p_a) can move continuously along a line $L^{v_{i-1}}$ that is perpendicular to x-y plane and pass through v_{i-1}, either up or down depending on whether v_i is an over-crossing or an under-crossing respectively. During the folding, since all vertices are moving along $L^{v_{i-1}}$, no link will collide with the fixed links.

An example of the folding is shown in Figure 7, as long as we maintain the length $l(AB) \ge l(BC)$. If the link before p_{m-1} does not satisfy the situation, we let AB be the set of links that is first longer than BC. Since we have set $d(l^1_{j+1}) = d(l_{j+1}) \ge \max(l^a_b)$, the situation will always be satisfied near p_j.

After the folding sequence moves link l^1_{j+1} ($p_{j+1}p^1_{j+1}$) to be perpendicular to the x-y plane where p_{j+1} is directly above p^1_{j+1}, move p^1_{j+1} on the plane $z = z(v_{i-1})$ along $\overrightarrow{v'_{i-1}v'_i}$ and stop when $d(p^1_{j+1}, v_i^o) = \varepsilon > 0$. Then move link l^2_{j+1} to let p^2_{j+1} occupy the location of v_i. Repeat this process to approximate K_p using P.

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

Although the preceding theorem gives a sufficient number of links, analysis of particular knots may allow tighter bounds. For example, to fold the configuration calculated using the algorithms from Section **??**, 7 links are sufficient to fold an overhand knot (Theorem 11 requires 7 line segments), and 18 links are sufficient for the double coin knot (Theorem 11 15 line segments).

Theorem 13: At least 5 links are necessary to fold an overhand polygonal knot.

Sketch of proof: Four links are necessary to create 3 crossings. However, consider the plane P formed by the first three endpoints of the links v_1, v_2 and v_3. To form an overhand knot, the projection of v_4 must be outside the triangle formed by v_1, v_2 and v_3. Without loss of generality, let v_4 be above P. Then, to form an overhand knot, the last link must penetrates P, with the last vertex v_n above P and projects outside triangle formed by v_1, v_2 and v_3 (can be verified using the Gauss code). Then, v_4 (above P) must connects to v_{n-1} which is below P. Therefore, $n-1=5$, and 5 links are necessary to form an overhand knot.

Theorem 14: There exist 5-link polygonal arcs that cannot fold into any overhand polygonal knot.

Sketch of proof: Consider a 5-link polygonal arc with lengths $(100,1,1,1,100)$. Folding the overhand knot involves an operation to insert one of the links on the end through a triangle formed by other links, which in this case has a limited size. Two of the edges have length 1, and the other edge is shorter than 2. What is more, one of the endpoints for the outer link has to be no further than 1 unit away from the triangle. Therefore, such motion cannot be achieved.

VI. Conclusions and Future work

In this work, we showed theoretical bounds on the number of fingers needed to grasp and fold string into knots, while ensuring that the string is held tautly in a polygonal arc. The proofs are constructive and give explicit finger placements and folding motions.

In the future, we expect to further study more efficient motions of the fingers, possibly in parallel, to fold knots. We hope to learn systematically how to quickly tie knots, and build knot-tying devices based on these principles.

This work is supported by NSF grant IIS-1217447.

References

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

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

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

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

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

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

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

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

[9] H. Inoue and M. Inaba. Hand-eye coordination in rope handling. *Robotics Research: The first International Symposium*, pages 163–174, 1985.

[10] L.H. Kauffman. *Knots and Physics*. K & E series on knots and everything. World Scientific, 1991.

[11] Charles Livingston. *Knot theory*. The carus mathematical monographs, Volume Twenty Four. The mathematical Association of America, Washington D.C., 1993.

[12] Vassily Manturov. *Knot Theory*. CRC Press, 2004.

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

[14] Elon Rimon and Andrew Blake. Caging planar bodies by one-parameter two-fingered gripping systems. *I. J. Robotic Res.*, 18(3):299–318, 1999.

[15] 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.

[16] Elon Rimon and A. Frank van der Stappen. Immobilizing 2-d serial chains in form-closure grasps. *IEEE Transactions on Robotics*, 28(1):32–43, 2012.

[17] 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.

[18] Mitul Saha and Pekka Isto. Manipulation planning for deformable linear objects. *IEEE Transaction on Robotics*, 23(6):1141–1150, December 2007.

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

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

[21] Weifu Wang and Devin Balkcom. From grasping to folding polygonal knots. *http://cs.dartmouth.edu/~harrisonwfw/tech-report-folding-poly.pdf*, 2015.

[22] Weifu Wang, Matthew P. Bell, and Devin Balkcom. Towards arranging and tightening knots and unknots with fixtures. In *Workshop on the Algorithmic Foundations of Robotics (WAFR), Istanbul, Turkey*, 2014.

[23] Weifu Wang, Matthew P. Bell, and Devin Balkcom. Towards arranging and tightening knots and unknots with fixtures. *IEEE Transactions on Automation Science and Engineering, Special Issue*, to appear, 2015.