Randomized Path Planning for Linkages with Closed Kinematic Chains

Jeffery H. Yakey, Steven M. LaValle, Lydia E. Kavraki

Abstract—
We extend randomized path planning algorithms to the case of articulated robots that have closed kinematic chains. This is an important class of problems, which includes applications such as manipulation planning using multiple open-chain manipulators that cooperatively grasp an object, and planning for reconfigurable robots in which links might be arranged in a loop to ease manipulation or locomotion. Applications also exist in areas beyond robotics, including computer graphics, computational chemistry and virtual prototyping. Such applications typically involve high degrees of freedom, and a parameterization of the configurations that satisfy closure constraints is usually not available. We show how to implement key primitive operations of randomized path planners for general closed kinematics chains. These primitives include the generation of random free configurations and the generation of local paths. To demonstrate the feasibility of our primitives for general chains, we show their application to recently developed randomized planners and present computed results for high-dimensional problems.

Keywords: randomized path planning, kinematic chains, closed linkages.

I. Introduction

This paper addresses the problem of path planning for general linkages that have closed kinematic chains with redundant degrees of freedom, in an environment that contains obstacles, as shown in Figure 1. In general, the constraints imposed by a closed linkage form an algebraic variety and in principle complete planners such as [6], [3] could be used; however, the high computational complexity of all of these algorithms for problems with high degree of freedom makes them too prohibitive for practical use. This motivates our approach in this paper, which extends randomized planning techniques that were developed for open-chain systems [13], [19], to general closed-chain systems.

Planning for linkages with closed kinematic chains has applications both in and beyond robotics. Parallel manipulators involve closed kinematic constraints [22]. In manipulation planning, when multiple robots grasp a single object, they form a closed loop containing the object as a link of the chain [1], [15]. Many of the existing methods for manipulation planning require inverse kinematics solutions for the robots [15] which can be a limitation. Regrasping is also an important issue as one or more of the manipulators often attain a singular configuration [23]. The ability to plan for linkages with closed kinematics chains eliminates the need of inverse kinematics solutions and could reduce the number of regrasps needed during manipulation tasks, as the linkage will be considered as a whole rather than as multiple, independent manipulators. A planner for closed linkages can also be applied to *reconfigurable* robots. Typically, this type of robot is composed of multiple, independent robots that can connect and disconnect from one another [16], [24], [27]. Closed linkages often occur during locomotion or reconfiguration of such complex robots [16], [27].

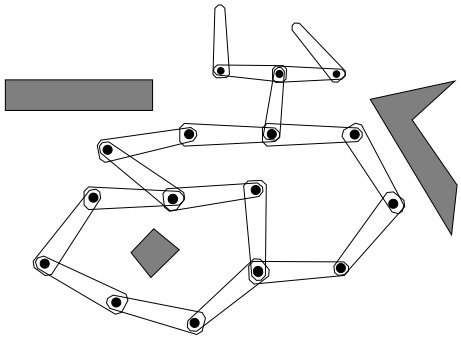


Fig. 1. We investigate path planning for linkages that have closed kinematic chains and must avoid static obstacles.

Many of the concepts used in path planning for robotics can also be applied to computer graphical animation [5]. Human-like characters can naturally be modeled as linkages and planning techniques are well suited to animate those characters [28]. However, a difficulty arises when these characters manipulate an object with both arms (e.g., pick up a box, two characters grasp each other, etc.), because this forms a closed linkage. There already exist algorithms capable of planning for this problem [14], but as in coordinated manipulation planning, a decoupling of the planning for the animated character and the object is done. Another application that could benefit from a planner for closed linkages lies in virtual prototyping [7]. For designs that include closed linkages, a planner could automate testing, and potentially avoid constructing physical prototypes. Applications of path planning for linkages with closed chains also exist in computational chemistry. For example, a fundamental problem in drug design is to find low-energy configurations of molecules that satisfy rigidity constraints similar to those obtained for 3D linkages [18].

In this paper, we extend randomized path planners to deal with closed kinematic chains by showing how two important primitives of these planners can be implemented for closed kinematic chains. These primitives are the genera-

J. H. Yakey is with the Dept. of Computer Science, Iowa State University, Ames, IA 50011 USA. E-mail: yakeyj@cs.iastate.edu

S. M. LaValle (corresponding author) is with the Dept. of Computer Science, University of Illinois, Urbana, IL 61801 USA. E-mail: lavalle@cs.uiuc.edu, Phone: +1-217-265-6313

L. E. Kavraki is with the Dept. of Computer Science, Rice University, Houston, TX 77005 USA. E-mail: kavraki@cs.rice.edu

tion of random configurations and the generation of local paths. We adopt a very general definition for the kinematic chain. We also assume that inverse kinematics solutions are not available. Our goal is to demonstrate the feasibility of extending randomized planners in the general case. It is clear that by using robot-specific characteristics or inverse kinematics a more efficient solution can be achieved. We implement our developed primitives in the context of the Probabilistic Roadmap Planner (PRM) [13] and Rapidly-exploring Random Trees (RRTs) [19].

To the best of our knowledge, there are two published papers that extend randomized planners to handle closed kinematic chains apart from the earlier work in [15]. One is a previous paper of ours [20], which presents a subset of the work in our current paper. The other is a paper by Han and Amato [10]. In that paper, the authors show how to develop a PRM-based planner for closed kinematic chains. They break the closed chains into a set of open chains, apply standard PRM random sampling techniques and forward kinematics to one subset of the subchains, and then use inverse kinematics on the remaining subchains to enforce the closure constraints. While this approach has been shown to perform well with a robot consisting of a single chain of varied length [10], experiments are not reported for the performance of the approach for general chain systems. Undoubtedly, both [10] and our work advance the state-of-the-art in using randomized planners for planning for mechanisms with closed loops and we hope that further research will result in efficient randomized planners for closed kinematics systems.

II. Problem Formulation

In this section, a formal definition of a closed linkage will be presented, and the path planning problem is formulated in the context of these linkages.

A. Definition of a Linkage

Our problem will be defined in a bounded two or three dimensional world, $\mathcal{W} \subset \mathbb{R}^N$, such that $N = 2$ or $N = 3$. A *link*, L_i, is a rigid body in the world, which represents a closed, bounded point set. Let $\mathcal{L} = \{L_1, L_2, \ldots, L_{n_l}\}$ denote a finite collection of n_l links. A *joint* J_k contains the following information:

1. a subset of links $\{L_i, L_j, \ldots, L_m\} \subseteq \mathcal{L}$ connected by J_k
2. the point of attachment for each L_i
3. the type of joint (revolute, spherical, etc.)
4. the range of allowable motions

Let $\mathcal{J}$ be a collection of n_j joints, each of which connects various links in $\mathcal{L}$. We then define $\mathcal{M} = (\mathcal{L}, \mathcal{J})$ to be a *linkage*[1]. It will sometimes be convenient to consider $\mathcal{M}$ as a graph in which the joints correspond to vertices and the links correspond to edges. Therefore, let G_M denote the underlying graph of $\mathcal{M}$. The special case of unary links (a link connected to a single joint) in $\mathcal{M}$ needs to be addressed, since the edge corresponding to these links will only connect one vertex. An artificial vertex needs to be created in G_M for each unary link, and it will be connected only to the edge corresponding to the unary link. According to the connectivity of G_M, we will then group linkages into classes[2]. If G_M is a tree, then we will consider this type of linkage to be *open*. A special case of an open linkage is an *open chain linkage*, in which all the vertices of G_M have degree less than three. In the case where G_M is cyclic and all vertices have degree greater than one, we will call this a *closed linkage*. We define a *closed chain linkage* to be a closed linkage in which all the vertices have degree exactly two. The last class is the *compound linkage*, in which G_M is cyclic with at least one vertex having degree one.

B. Kinematic Closure Constraints

The kinematics of $\mathcal{M}$, are expressed using standard parameterizations for chains [9], [12]. The *configuration* of $\mathcal{M}$ is a vector, q, of real-valued parameters that uniquely determine the position and orientation of all links. The dimension of q is the number of degrees of freedom of $\mathcal{M}$.

In this paper, we are primarily concerned with the case in which $\mathcal{M}$ is a closed or compound linkage, implying that G_M contains cycles. For this case, there will generally exist configurations that do not satisfy *closure constraints* of the form $f(q) = 0$. These constraints can be defined by breaking each cycle in G_M at a vertex, v, and writing the kinematic equation that forces the pose of the corresponding joint to be the same, regardless of which of the two paths were chosen to v. Let $\mathcal{F}$ represent the set $\{f_1(q) = 0, f_2(q) = 0, \ldots, f_m(q) = 0\}$ of m closure constraints, whose formulation will be formally defined in Section III-A. In general, if n is the dimension of $\mathcal{C}$, then $m < n$. Let $\mathcal{C}_{cons} \subset \mathcal{C}$ be defined as:

$$\mathcal{C}_{cons} = \{q \in \mathcal{C} \mid \forall f_i \in \mathcal{F}, f_i(q) = 0\}, \tag{1}$$

which denotes the set of all configurations that satisfy the constraints in $\mathcal{F}$.

A collision is defined for $\mathcal{M}(q)$ if any of the links of $\mathcal{M}(q)$ collides with any of the workspace obstacles or the other links in $\mathcal{L}$. Consecutive links usually do not give rise to collisions. Let $\mathcal{W}$ include a set of obstacles $\mathcal{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_{n_b}\}$, which are each a closed subset of $\mathcal{W}$. Using standard terminology, let $\mathcal{C}_{free}$ denote the set of all configurations such that $\mathcal{M}(q)$ is not in collision. Formally, this is:

$$\mathcal{C}_{free} = \{q \in \mathcal{C} \mid (\mathcal{M}(q) \cap \mathcal{B} = \emptyset) \wedge (\forall L_i, L_j \in \mathcal{M}(q), L_i \cap L_j = \emptyset)\}, \tag{2}$$

where L_i, L_j are nonconsecutive links.

In addition to the usual complications of path planning for articulated linkages having many degrees of freedom, we are faced with the additional challenge of keeping the configuration in $\mathcal{C}_{cons}$. Let $\mathcal{C}_{sat} = \mathcal{C}_{cons} \cap \mathcal{C}_{free}$ define the

[1] We use the more general definition of linkage that includes open and closed kinematic chains [9], rather than a linkage that contains only closed chains [11].

[2] Note that these classes deviate from the standard terminology used in mechanism design [11]. Our intent was for a *chain* to imply linearity of the linkage, and for *closed* to mean that the linkage contains no unary links.

set of configurations satisfying both closure and collision constraints.

Although $\mathcal{C}$ is typically a manifold, $\mathcal{C}_{cons}$ will be more complicated. Each of the holonomic constraints in $\mathcal{F}$ is a smooth function with a non-zero derivative. Using stereographic projection, these constraints can be reformulated as polynomial equations, and together these constraints form a system of equations that characterize the configurations satisfying the closure constraints. A *real algebraic variety* can be defined by the polynomial equations $f_1(q) = \ldots = f_m(q) = 0$. The surfaces defined by these varieties are not smooth in general, and can contain singular points. Therefore, a variety is not necessarily a manifold, although a real algebraic variety can be split into a finite number of manifolds [26]. Because of the nature of these closure constraints, we will assume that we have no *a priori* knowledge of a parameterization for the variety.

Our problem reduces to path planning in a space with lower dimension than $\mathcal{C}$, due to the fact that the equality constraints in $\mathcal{F}$ reduce the dimensionality of $\mathcal{C}$. Since we have no efficient way to reduce the number of parameters needed to specify the configuration for a closed linkage, we allow a tolerance for $\mathcal{C}_{cons}$, which means that the constraints will be satisfied to within some numerical precision. This tolerance will be the subject of Section III-A.

C. Finding a Path

Our problem reduces to path planning in $\mathcal{C}_{sat}$, which has lower dimension than $\mathcal{C}$. Initially we are given $q_{init} \in \mathcal{C}_{sat}$ and $q_{goal} \in \mathcal{C}_{sat}$, the *initial configuration* and *goal configuration*, respectively. The task is to find a continuous path $\tau : [0,1] \rightarrow \mathcal{C}_{sat}$ such that $\tau(0) = q_{init}$ and $\tau(1) = q_{goal}$. For a path to exist between q_{init} and q_{goal}, it will be necessary that they are both contained within the same connected component of $\mathcal{C}_{sat}$.

The existence of closed kinematic chains greatly increases the difficulty of path planning because the set of configurations that satisfy closure constraints is usually expressed in terms of implicit equations. In the traditional path planning problem, a parameterization of the configuration space is available. If closure constraints exist, a parameterization is usually not available (except for some specific mechanisms), and the set of valid configurations is generally not even a manifold.[3]

D. A Specific 2D Model

The following model will be used to facilitate later concepts and for our implementation: 1) $\mathcal{L}$ is a collection of line segments in a 2D world; 2) joints are revolute and attach links at their endpoints; 3) there are joint limits (e.g., joints are not allowed to rotate into the range $0 \pm \delta$, in which δ is a parameter for the joint limit); 4) one of the joints attaches a link to the origin $(0,0)$ in the world, $\mathcal{W}$; 5) the obstacle region is polygonal.

[3]Even though it can be expressed as a stratification of manifolds [6], parameterizations of the strata are still unknown.

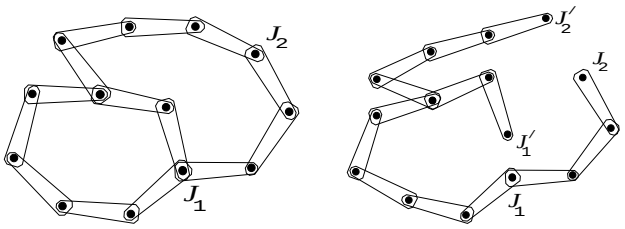


Fig. 2. An example of breaking cycles in a linkage.

III. Generating Random Samples

One of the most basic operations in many randomized planners is the construction of random configurations. For example, the basic PRM approach [13] uses randomly-generated configurations that lie in $\mathcal{C}_{free}$. These can be found by simply generating configurations in $\mathcal{C}$, and rejecting those in collision. The problem is considerably more complicated for closed kinematic chains because all samples must lie in $\mathcal{C}_{cons}$, satisfying closure constraints. This section provides a general approach to generating random samples in $\mathcal{C}_{sat}$. The use of kinematic error is an integral component of our approach.

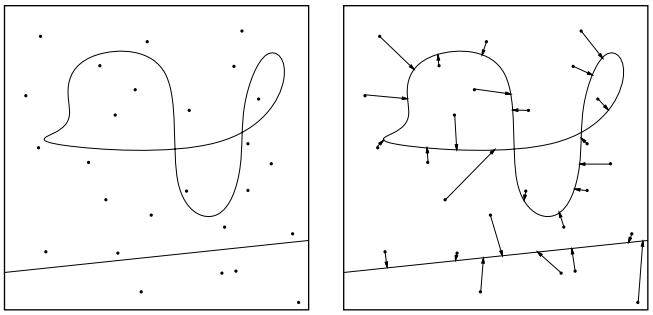
A. Kinematic Error

To handle the closure constraints in $\mathcal{F}$, we define a new linkage, $\mathcal{M}' = (\mathcal{L}', \mathcal{J}')$, which is obtained by breaking cycles in the underlying graph G_M of $\mathcal{M}$. Let the set of links be the same, $\mathcal{L}' = \mathcal{L}$. Let $\mathcal{J}'$ be a superset of $\mathcal{J}$ and contain $n_j + m$ joints, where a new joint is added for each of the m cycles in G_M. For each cycle in G_M, the joint where the break occurs can be selected arbitrarily, and will be denoted by J_k. There will be two links from the cycle in G_M that are attached by J_k. For one of these links, disconnect it from J_k and form a new joint J_k' on the link where J_k was formerly attached. If this insertion of joints is performed for each cycle of G_M, the result will be a linkage $\mathcal{M}'$ which has no cycles ($G_{M'}$ is a tree). An example of "breaking" the loops in a linkage is shown in Figure 2. In $\mathcal{M}'$, the configuration of any link can be determined by applying the forward kinematic equations to the sequence of links on the unique path to L_0.

Neglecting self-collision, note that M' can achieve any configuration in $\mathcal{C}$. If J_k and J_k' have the same position in $\mathcal{W}$, then a closure constraint from $\mathcal{M}$ is satisfied. If this is true for all joints in $\mathcal{J}' \setminus \mathcal{J}$, then the configuration lies in $\mathcal{C}_{cons}$. The closure constraint $f_i(q)$ can be written by subtracting the kinematic expression for $J_k(q)$ from the expression for J_k' using the equations from Section II, and will be done as follows. Let $B \subset \{1, \ldots, n_j\}$ be the indices of the set of joints that were broken in $\mathcal{M}$ to form $\mathcal{M}'$. A kinematic error function can be defined as:

$$e(q) = \sum_{k \in B} \|J_k(q) - J_k'(q)\|^2. \tag{3}$$

Alternatively, the maximum (or any L^p norm) can be used to combine the error from each broken loop. This error



a. b.

Fig. 3. (a) The curves depict $\mathcal{C}_{cons}$, and configurations are chosen at random in $\mathcal{C}$. (b) Randomized error minimization is performed on the samples to force as many as possible onto $\mathcal{C}_{cons}$.

```
GENERATE_RANDOM_SAMPLE()
 1  q ←RANDOM_CONFIGURATION();
 2  i ← 0;  j ← 0;
 3  while i < I and j < J and e(q) > ε do
 4      i++;  j++;
 5      q' ←RANDOM_NHBR(q);
 6      if e(q') < e(q) then
 7          j ← 0;  q ← q';
 8  if e(q) ≤ ε then Return q
 9          else Return FAILURE
```

Fig. 4. An algorithm that iteratively attempts to reduce the kinematic error of a linkage.

function allows us to redefine $\mathcal{C}_{cons}$ as follows:

$$\mathcal{C}_{cons} = \{q \in \mathcal{C} \mid e(q) = 0\}.$$

Since the equality constraints that define kinematic closure are implicit, we allow a specified real-valued tolerance $\epsilon > 0$ to determine when the closure constraints are satisfied. This enables incremental linear motions to be made along the $\mathcal{C}_{cons}$, and gives us new definitions for $\mathcal{C}_{cons}$ and $\mathcal{C}_{sat}$ that take ϵ into consideration:

$$\tilde{\mathcal{C}}_{cons} = \{q \in \mathcal{C} \mid e(q) \leq \epsilon\},$$

$$\tilde{\mathcal{C}}_{sat} = \tilde{\mathcal{C}}_{cons} \cap \mathcal{C}_{free}.$$

By using the ϵ tolerance, we allow some freedom for the randomized algorithms as they travel on the constraint surface.

Without the tolerance, we would need to use more costly algebraic techniques to incorporate the closure constraints into our planner, which would decrease the number of allowable degrees of freedom for a linkage.

B. Gradient Descent

Figure 3 illustrates the problem of generating vertices in $\mathcal{C}_{sat}$. A random sample in $\mathcal{C}$ can easily be generated (of course, its distribution depends on the parameterization of $\mathcal{C}$), but is not very likely to be in $\tilde{\mathcal{C}}_{sat}$. The algorithm in Figure 4 gives pseudocode for a randomized descent technique that iteratively attempts to reduce the error function, $e(q)$ from Section III-A. The approach we use is to break the kinematic loops and minimize the sum of squares the Euclidean distances of each joint that is not where it should be to satisfy kinematic closure. An alternative would have been to define each of the closure constraints $f_i(q)$ in polynomial form. The algebraic distance could then be minimized, or an approximation to the Euclidean distance in $\mathcal{C}$ may easily be minimized [25].

The algorithm GENERATE_RANDOM_SAMPLE requires three constants: ϵ, which is the numerical tolerance on the error function, I, which is the maximum number of search steps, and J which is the maximum number of consecutive failures to close the kinematic chains. The function RANDOM_NHBR takes in a configuration q as a parameter, and produces a new random configuration q' in $\mathcal{C}_{free}$. The distance between the new configuration q' and q will be within a fixed amount d_{max}, which will generally be very small. RANDOM_NHBR may have to guess many nearby configurations to produce one that is collision-free. $e(q)$ measures the kinematic error of configuration q as this is specified in Eq. 3. Rather than compute a complicate gradient of $e(q)$, any random configuration q' in which $e(q') < e(q)$ is kept. This was observed in [2] to be much faster than computing an analytical gradient for high-degree-of-freedom problems. If the algorithm becomes trapped in a local minimum and returns FAILURE, then the sample is simply discarded. This has no serious effect on the overall approach, except that some computation time is wasted. Other approaches, such as the Levenberg-Marquardt [21] nonlinear optimization algorithm could be used instead of randomized descent, but one must be careful not to introduce an unwanted deterministic bias on the solutions.

C. A Computed Example

We performed the following experiment to demonstrate that the method presented above can generate a variety of samples for closed-kinematics chains. We placed obstacles in a 2D world so that there would be many distinct connected components in $\mathcal{C}_{sat}$ (see Figure 5). We then generated a PRM roadmap for this world and observed the various connected components to determine whether they were all represented. In Figure 5, it can be seen that many of the generated nodes wrap around various obstacles and have different orientations. Each of these configurations lies in a distinct connected component of $\tilde{\mathcal{C}}_{sat}$, which means that no path exists between these configurations. This experiment illustrates the ability of random sampling to simultaneously explore all components of a space, which is advantageous for PRM-type multiple-query planners.

IV. Generating Local Motions

Nearly all existing randomized path planners require the generation of local motions in $\mathcal{C}_{free}$. To extend these planners, operations are needed that generate local motions in

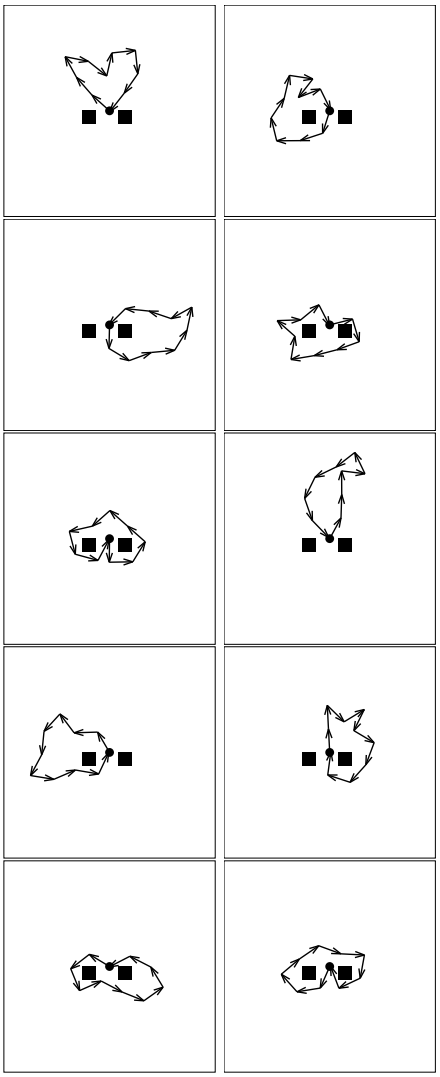


Fig. 5. All ten components were found in a 2D world that contains obstacles.

$\mathcal{C}_{cons}$ or $\mathcal{C}_{sat}$. Given a configuration $q \in \mathcal{C}_{sat}$, the task is to generate nearby configurations that also lie in $\mathcal{C}_{sat}$ and are reachable from q by a local motion.

A. Random Steps in the Tangent Space

Suppose that a configuration $q \in \mathcal{C}_{sat}$ is given. We will use random sampling to generate incremental motions. It is preferable to generate samples that locally follow the tangent space of the constraints, rather than choosing a random direction. The *tangent space* is the set of tangent vectors for some $q \in \mathcal{C}_{cons}$. Using a tolerance ϵ, each of the tangent vectors gives us a direction from q that is likely to remain in $\tilde{\mathcal{C}}_{sat}$, which we can exploit when we wish to move locally. By sampling in the tangent space when searching for configurations within a neighborhood of q, we will be more likely to generate a new configuration that satisfies all closure constraints. The differential configuration vector dq lies in the tangent space of a constraint $f_i(q) = 0$ if

$$\frac{\partial f_i(q)}{\partial q_1} dq_1 + \frac{\partial f_i(q)}{\partial q_2} dq_2 + \cdots + \frac{\partial f_i(q)}{\partial q_n} dq_n = 0. \quad (4)$$

This leads to the following homogeneous system for all of the m closure constraints:

$$\begin{pmatrix} \frac{\partial f_1(q)}{\partial q_1} & \frac{\partial f_1(q)}{\partial q_2} & \cdots & \frac{\partial f_1(q)}{\partial q_n} \\ \frac{\partial f_2(q)}{\partial q_1} & \frac{\partial f_2(q)}{\partial q_2} & \cdots & \frac{\partial f_2(q)}{\partial q_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_m(q)}{\partial q_1} & \frac{\partial f_m(q)}{\partial q_2} & \cdots & \frac{\partial f_m(q)}{\partial q_n} \end{pmatrix} \begin{pmatrix} dq_1 \\ dq_2 \\ \vdots \\ dq_n \end{pmatrix} = \mathbf{0}. \quad (5)$$

Recall that $m < n$. If the rank of the matrix is $k \leq m$, then $n - k$ configuration displacements can be chosen independently, and the remaining k parameters must satisfy Equation 5. We use singular value decomposition (SVD), to compute an orthonormal basis for the tangent space. This enables our algorithm to follow the tangent space and generate the $n - m$ random scalar displacements needed for the linear combination. This technique increases the likelihood that local motions will remain within tolerances for larger step sizes, thus improving the efficiency of our algorithms.

To use this technique, it is critical to efficiently compute the partial derivatives for each of our constraints. Each of these closure constraints is formulated by finding the algebraic equations that force J_k and J'_k at each break to have the same position in the world. J_k and J'_k can be considered as unary joints, or in other words, there is only one link attached to each of them. L_k and L'_k will denote these links for J_k and J'_k, respectively. Note that L_k and L'_k will each have a unique chain of links to the root link L_0 since the linkage is acyclic. The partial derivatives of these open chains are efficiently computed for $\mathcal{W} \subseteq \mathbb{R}^2$. We recursive formulas to compute the x and y positions for the origin of each link in $\mathcal{W}$:

$$X_n = \cos(q_n) X_{n-1} - \sin(q_n) Y_{n-1} + \ell_{n-1}, \quad (6)$$

where $X_0 = x$ and

$$Y_n = \sin(q_n) X_{n-1} + \cos(q_n) Y_{n-1}, \quad (7)$$

in which $Y_0 = y$. In Equations 6 and 7, n represents the index of the link in each open chain, and q_n represents the angle between successive links. So, L_0 will have index 0, etc. Once again, ℓ_n is the length of a link and the (x, y) values are the coordinates of a link with respect to its coordinate frame. These formulas yield an algebraic representation of the kinematics for each open chain of links, but the partial derivatives with respect to each parameter $q_i \in q$ need to be computed. For each of the above formulas, there are two cases to be considered when taking the partial derivatives: taking the derivative with respect to the parameter for link n, or for one of the other $i < n$ links:

$$\frac{\partial X_n}{\partial q_i} = \begin{cases} \cos(q_n) \frac{\partial X_{n-1}}{\partial q_i} - \sin(q_n) \frac{\partial Y_{n-1}}{\partial q_i} & i < n \\ -\sin(q_n) X_{n-1} - \cos(q_n) Y_{n-1} & i = n \end{cases} \quad (8)$$

$$\frac{\partial Y_n}{\partial q_i} = \begin{cases} \sin(q_n) \frac{\partial X_{n-1}}{\partial q_i} - \cos(q_n) \frac{\partial Y_{n-1}}{\partial q_i} & i < n \\ \cos(q_n) X_{n-1} - \sin(q_n) Y_{n-1} & i = n \end{cases} \quad (9)$$

```
CONNECT_CONFIGURATIONS(q, q')
1   i ← 0;  j ← 0;  k ← 0; L ← {q};
2   while i < I and j < J and k < K and
                    ρ(LAST(L), q') > ρ0 do
3       i++;  j++;
4       q'' ←RANDOM_NHBR(LAST(L));
5       if e(q'') ≤ ε then
6           j ← 0;  k++;
7           if ρ(q'', q') < ρ(LAST(L), q') then
8               k ← 0;  L ← L + {q''};
9   if ρ(LAST(L), q') ≤ ρ0 then Return L
10                        else Return FAILURE
```

Fig. 6. An algorithm that iteratively attempts to move a system from one vertex to another while keeping q in $\tilde{\mathcal{C}}_{sat}$.

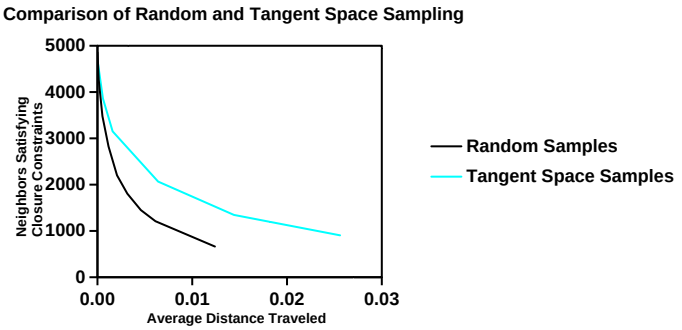


Fig. 7. Comparison between random and tangent space sampling for random neighbor generation of an 8-link closed chain linkage.

By using the recursive linkage of these equations to our advantage, memoized dynamic programming [8] can be used to efficiently evaluate these expressions for given configurations. The partial derivatives are computed iteratively starting from $n = 0$, and each value is stored in a table for reuse in later iterations. The following two equations avoid computing values for Equations 6 and 7:

$$X_n = \frac{\partial Y_n}{\partial q_n} + \ell_{n-1}, \tag{10}$$

$$Y_n = -\frac{\partial X_n}{\partial q_n}. \tag{11}$$

B. Connecting Nearby Configurations

Some randomized path planners, such as the PRM, require the generation of paths that connects nearby configurations. This can be accomplished by chaining together a sequence of local steps using the method just presented. Let q and q' be two configurations in $\mathcal{C}_{sat}$ that we wish to connect (if possible).

To describe what is meant by "nearby," a distance metric will be defined. For the experiments in Section V we use a Euclidean metric on the configuration space (appropriately adjusted for the topology). An alternative is to compute the sum of squares of the Euclidean displacements for all of the joints in $\mathcal{J}$ [13].

The algorithm in Figure 6 attempts to reduce $\rho(q, q')$, the distance from q to q', by a randomized gradient descent that simultaneously maintains the kinematic error to within ϵ and reduces ρ, but is free to travel due to the allowed tolerances on the closure constraints.

The overall structure of the CONNECT_CONFIGURATIONS algorithm is similar to GENERATE_RANDOM_SAMPLE. An additional constant K is used to terminate the search after K consecutive failures to reduce ρ, even though kinematic closure is maintained. Also, the constant ρ_0 is introduced to stop the algorithm when the path from q is sufficiently close to q'. In some cases, it might be preferable to switch the order of Lines 5 and 7, depending on whether we want to prioritize the minimization of distance over the satisfaction of the closure constraint. The success of the algorithm is based on the assumption that the selected vertices are close enough to ensure local minima and collision constraints are not likely to prevent connection.

One drawback of creating paths using randomized gradient descent is that the path needs to be stored for every edge we add to the roadmap. The reason is that there is no longer a guarantee, due to the randomization, that a path can be regenerated between these vertices at a later time. Another reason is that the gradient descent is computationally expensive to perform, and the computation required during the query phase should be minimized. However, once a path has been generated, several path optimization algorithms from Section V can be used to reduce the length of the path. As a result, the amount of space needed to store the paths in the roadmap is reduced, along with the added benefit of the higher quality paths.

C. Experiments

We again peformed experiments to demonstrate the feasibility and advantages of random sampling versus tangent space sampling when generating a random neighbor of a configuration. We generated 5000 random configurations satisfying the closure constraints, and for each of these configurations a random neighbor was computed using both the random and tangent space sampling methods. The number of random neighbors satisfying the closure constraints was recorded, as well as their average distance from the original random configuration. This experiment was performed repeatedly, changing the parameters of the two methods to vary the average distance traveled between the random configuration and its random neighbors. The chart in Figure 7 compares the two sampling methods for an 8-link closed chain linkage, and Figure 8 is a comparison for a 7-link closed linkage that has two loops (the linkage is shown in Figure 9).

It is readily seen that for both linkages the tangent space sampling will outperform random sampling in both criteria. Tangent space sampling is more likely to produce a new configuration satisfying the closure constraints, as well as generating random neighbors along a greater distance. Both of these can improve the overall computation time because more successful random neighbor sampling leads to less wasted computation and increasing the dis-

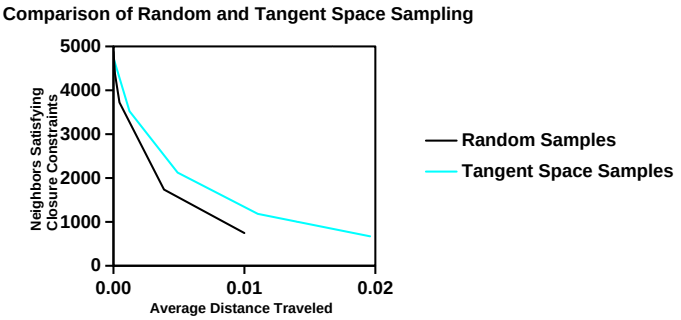


Fig. 8. Comparison between random and tangent space sampling for random neighbor generation of a 7-link, 2-loop closed linkage.

tance traveled per step speeds connection of two configurations. Computing the tangent space samples is more expensive to perform, though. The average time needed to generate a neighbor using random sampling took 6.94 microseconds, while tangent space sampling took 1.518 milliseconds. Even though the tangent space sampling is more expensive to perform, the extra distance it allows the random neighbors to travel makes up for this added expense. Another factor to be considered is the time spent performing collision detection, which usually dominates the time needed to compute the random neighbor using either random or tangent space sampling.

V. Path Planning Experiments

In this section we extend a PRM-based planner and an RRT-based planner by applying the methods introduced in Sections III- IV. Two notes are in order. First collision detection was performed naively by testing all pairs of line segments. Second, we optimize computed paths given by the methods in Section IV as follows. We repeatedly iterate over the path, analyzing every triple v_i, v_{i+1}, and v_{i+2}. We compute the distance $d = \rho(v_i, v_{i+2})$ and determine whether $d < d_{max}$. In this case, we can delete v_{i+1} without violating the maximum distance between two consecutive configurations in a path. If $d \geq d_{max}$, then we attempt to incrementally move v_{i+1} closer to the straight line between v_i and v_{i+2}, as far as possible before violating the maximum kinematic error allowance.

A. PRM Results

The implemented version of PRM is a modification of the planner presented in [13]. A large number of configurations are distributed uniformly at random in the configuration space and those that are collision-free are retained as nodes of a roadmap. A local planner is then used to find paths between each pair of nodes that are sufficiently close together. If the planner succeeds in finding a path between two nodes, they are connected by an edge in the roadmap. In the query phase, the user specified start and goal configurations are connected to the roadmap by the local planner. Then the roadmap is searched for a shortest path between the given points.

We use GENERATE_RANDOM_SAMPLE to generate configurations that lie in $\mathcal{C}_{sat}$. These serve as the vertices of the roadmap. The edges of the roadmap are generated using CONNECT_CONFIGURATIONS.

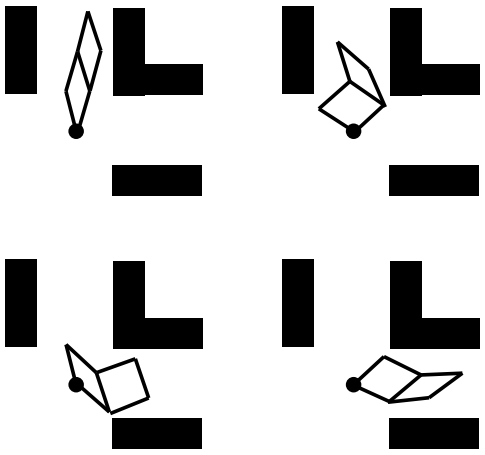


Fig. 9. Snapshots along the path of a closed linkage with two loops.

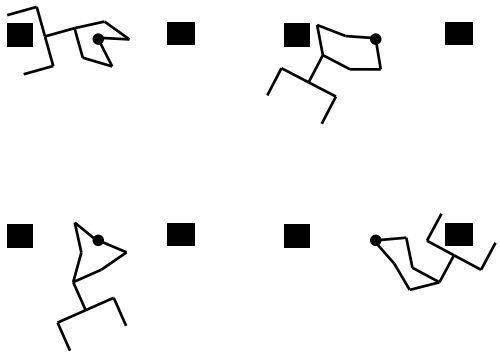


Fig. 10. Snapshots along the path of a manipulator example.

We now present three examples of linkages for which we have computed roadmaps. The first linkage is shown in Figure 9, and is composed of seven links configured into two loops. A path was generated using the roadmap, and four intermediate configurations in the path have been displayed. This linkage has three DOF: each of the loops in the linkage has a single DOF, and the base joint adds the third DOF. The next example considers a manipulator attached to a closed linkage, and is pictured in Figure 10. This linkage has 6 degrees of freedom: 5 from each link in the loop and one for the manipulator (the grippers are not able to move). The single closure constraint then reduces the total DOF to 4. Our final example in Figure 11 simulates two planar serial manipulators cooperatively grasping an object. This example has 8-DOF, because the two manipulators have 3 and 4 links plus the single DOF added by the manipulated object. Once again, the closure constraint reduces the total degrees of freedom to 6 for the linkage.

We did a straightforward implementation of the primitives described in this paper and focused in displaying the feasibility of our approach without worrying about

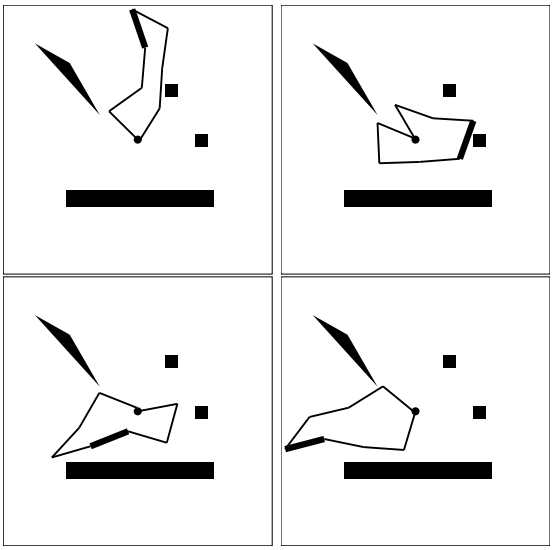


Fig. 11. Two manipulators grasping and moving an object.

performance. Indeed, all of the considered problems were solved (snapshots along computed paths are shown in Figures 9,10, and 11 but computing general PRM roadmaps required several hours of computation time resulting in roadmaps of several thousand nodes. This extensive computation time is due to the repeated execution of the GENERATE_RANDOM_SAMPLE and CONNECT_CONFIGURATIONS algorithms, which generally are very expensive. Note that the implemented version of PRM tries to generate a roadmap that captures the components of the free configuration space. The roadmap is then stored for answering multiple queries. After the roadmap has been precomputed, path queries can be run very quickly: once the initial and goal configurations have been connected to the roadmap, a simple graph search is all that is required to compute the remainder of the path.

B. RRT Results

The RRT-based planner is a modification of a planner presented in [19]. An RRT is a tree that is grown incrementally. Initially, there is a single vertex, q_{init}. In each iteration, a vertex is added to the tree by picking a random configuration, and then extending the vertex that is closest to the random sample [17], [19]. In the adaptation described here, the RRT is biased toward q_{goal} by selecting q_{goal} as a "random" sample a small percentage of the time.

We have computed several examples of paths for closed linkages using the RRT approach. Each of these examples were computed by selecting an initial configuration, and then the RRT was allowed to expand until 8000 nodes were added to the tree. The first example, shown in Figure 12, is another coordinated manipulation task for two serial manipulators grasping an object in the shape of a cross. This example has 9-DOF, but with closure constraints the number of degrees of freedom is reduced to 7. The time needed to generate this example was 1271.18 seconds.

The second example is of a snake-like compound linkage, shown in Figure 13, where the "head" of the snake

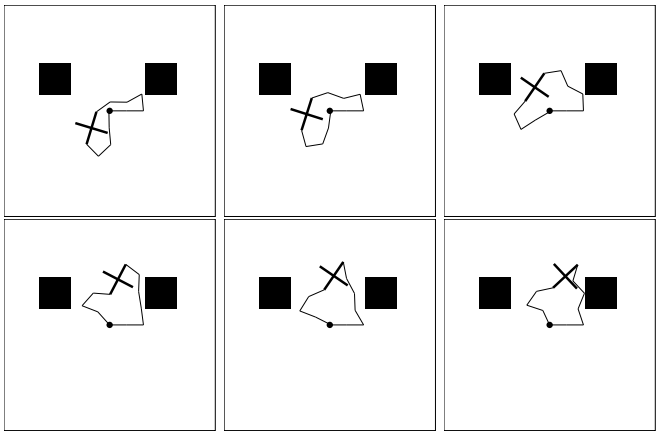


Fig. 12. Two manipulators grasping a cross-shaped object.

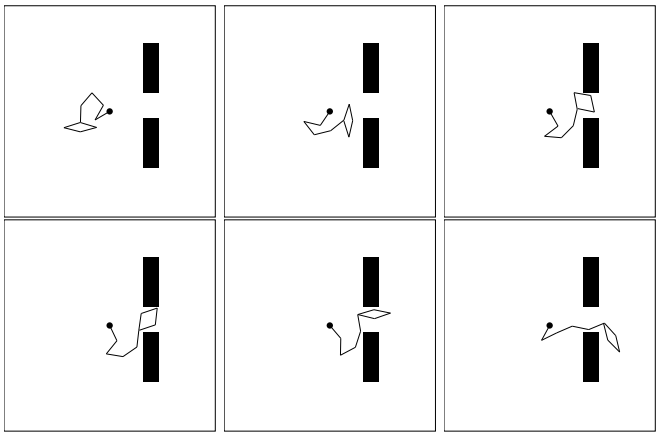


Fig. 13. A snake-like compound linkage example.

needs to compress so that it may fit through an obstacle. This linkage has 9-DOF, but again has a total of 7 degrees of freedom when closure constraints are considered. Altogether, this example required 468.7 seconds to compute.

The last RRT example is an 11-link linkage, shown in Figure 14 with 9-DOF once the closure constraints have been taken into account. The computation of this example took 888.22 seconds.

VI. Conclusions

We presented extensions of successful randomized planners to the case of linkages that have closed kinematic chains. Closure constraints are common in many applications such as robotics, computational chemistry, virtual prototyping, and computer graphics. The difficulty is that path planning must be performed in a complicated subset, $\mathcal{C}_{sat}$, of the configuration space. Our current experiments demonstrate the feasibility of our approach. We expect that substantial performance improvement can be obtained by taking the following steps: 1) using the motion primitives from this paper in recent, more-efficient planning algorithms, such as the LazyPRM [4] or RRTConCon [19]; 2) precomputing roadmaps while ignoring obstacles, as suggested in [20] and applied in [10]; 3) employing efficient nearest-neighbor algorithms and collision detection

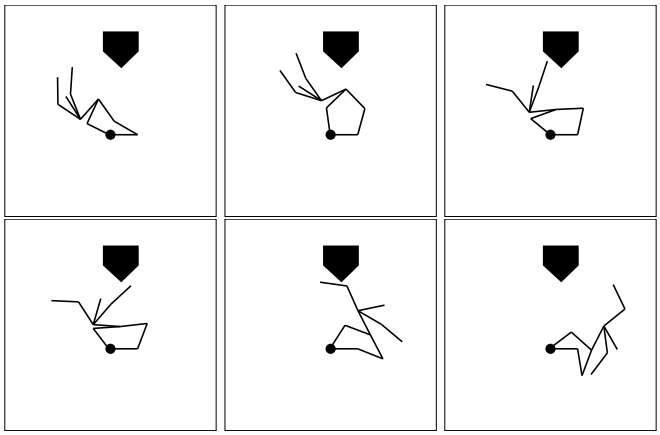


Fig. 14. An 11-link compound linkage example.

algorithms. We believe that a running-time improvement of a couple of orders of magnitude is possible; however, experimental support for this remains as a topic of future research.

Acknowledgments

We are very grateful to Jean-Claude Latombe and Paul Finn for our work in [18] which inspired the current paper. We thank Carlo Tomasi for supplying singular value decomposition code. We also thank Nancy Amato and Judy Vance for their helpful comments. Steve LaValle is partially supported by NSF CAREER award IRI-9875304 and a grant from Honda Research. Lydia Kavraki is partially supported by NSF CAREER Award IRI-970228, a Whitaker Award, an ATP Award and a Sloan Fellowship.

References

[1] R. Alami, T. Siméon, and J. P. Laumond. A geometrical approach to planning manipulation tasks. In *5th Int. Symp. Robot. Res.*, pages 113–119, 1989.

[2] J. Barraquand and J.-C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. In *IEEE Int. Conf. Robot. & Autom.*, pages 2328–2335, 1991.

[3] S. Basu, R. Pollack, and M.-F. Roy. Computing roadmaps of semi-algebraic sets on a variety. Submitted for publication, 1998.

[4] R. Bohlin and L. Kavraki. Path planning using lazy prm. In *IEEE Int. Conf. Robot. & Autom.*, 2000.

[5] R. Boulic and R. Mas. Hierarchical kinematic behaviors for complex articulated figures. In N. Thalmann and D. Thalmann, editors, *Interactive Computer Animation*, chapter 3. Prentice Hall, London, 1996.

[6] J. F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, 1988.

[7] H. Chang and T. Y. Li. Assembly maintainability study with motion planning. In *IEEE Int. Conf. Robot. & Autom.*, pages 1012–1019, 1995.

[8] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *An Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.

[9] A. G. Erdman and G. N. Sandor. *Mechanism Design: Analysis and Synthesis*. Prentice Hall, Upper Saddle, NJ, third edition, 1997.

[10] L. Han and N. M. Amato. A kinematics-based probabilistic roadmap method for closed chain systems. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 233–246. A K Peters, Wellesley, MA, 2001.

[11] R. S. Hartenberg and J. Denavit. *Kinematic Synthesis of Linkages*. McGraw-Hill, New York, NY, 1964.

[12] R. S. Hartenburg and J. Denavit. A kinematic notation for lower pair mechanisms based on matrices. *J. Applied Mechanics*, 77:215–221, 1955.

[13] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. & Autom.*, 12(4):566–580, June 1996.

[14] Y. Koga, K. Kondo, J. Kuffner, and J.-C. Latombe. Planning motions with intentions. *Computer Graphics (SIGGRAPH'94)*, pages 395–408, 1994.

[15] Y. Koga and J.-C. Latombe. On multi-arm manipulation planning. In *Proc. IEEE Int. Conf. on Rob. and Autom.*, pages 945–952, 1994.

[16] K. Kotay, D. Rus, M. Vora, and C. McGray. The self-reconfiguring robotic molecule: Design and control algorithms. In P.K. Agarwal, L. Kavraki, and M. Mason, editors, *Robotics: The Algorithmic Perspective*. AK Peters, Natick, MA, 1998.

[17] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. TR 98-11, Computer Science Dept., Iowa State University. <http://janowiec.cs.iastate.edu/papers/rrt.ps>, Oct. 1998.

[18] S. M. LaValle, P. Finn, L. Kavraki, and J.-C. Latombe. Efficient database screening for rational drug design using pharmacophore-constrained conformational search. *J. Computational Chemistry*, 21(9):731–747, 2000.

[19] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Workship on the Algorithmic Foundations of Robotics*, 2000.

[20] S. M. LaValle, J. H. Yakey, and L. Kavraki. A probabilistic roadmap approach for systems with closed kinematic chains. In *IEEE Int. Conf. Robot. & Autom.*, pages 1671–1676, 1999.

[21] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. of Appl. Math.*, 11:431–441, 1963.

[22] J.-P. Merlet, C. Gosselin, and N. Mouly. Workspace of planar parallel manipulators. *Mechanism and Machine Theory*, 33(1/2):7–20, 1998.

[23] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Boca Raton, FL, 1994.

[24] A. Pamecha, I. Ebert-Uphoff, and G. S. Chirikjian. Useful metrics for modular robot motion planning. *IEEE Trans. Robot. & Autom.*, 13(4):531–545, 1997.

[25] G. Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Trans. Pattern Anal. Machine Intell.*, 13(11):1115–1137, November 1991.

[26] H. Whitney. Elementary structure of real algebraic varieties. *Annals of Mathematics*, 66(3):545–556, November 1957.

[27] M. Yim. *Locomotion with a Unit-Modular Reconfigurable Robot*. PhD thesis, Stanford Univ., December 1994. Stanford Technical Report STAN-CS-94-1536.

[28] J. Zhao and N. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics*, 13(4):313–336, 1994.