Randomized Path Planning for Linkages with Closed Kinematic Chains

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

Abstract—
We present a set of primitives that can be used to 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. The dimension of the above problems is typically very high which suggests the use of randomized path planners for their solution. Currently, randomized techniques have been successfully applied to robots with open kinematic chains. One major difficulty when dealing with closed kinematic mechanisms is the fact that a parameterization of the set of configurations that satisfy closure constraints is usually not available. Rather than developing completely new planning algorithms, we present a set of primitives that can be used to extend existing randomized path planners to the case of linkages with closed kinematic chains. We focus on three operations that are commonly used by randomized planners: (a) the generation of random free configurations, (b) the generation of incremental paths and (c) the optimization of paths. To demonstrate the utility and generality of our primitives, we show their application to recently developed randomized planners and present several computed results for high-dimensional problems.

Keywords: Motion planning, randomized path planning

Submission Type: Regular paper

I. Introduction

This paper addresses the problem of path planning for multi-body systems that contain flexible kinematic loops, in an environment that contains obstacles, as shown in Figure 1. In other words, we consider general linkages that have closed kinematic chains, in addition to the usual motion planning constraints. Our motivation for considering this problem comes from the spectrum of applications that could benefit from such a planner.

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], [35]. Many of the existing methods for manipulation planning plan separately for the object and the robots [35] and require inverse kinematics solutions for the robots. Such methods, although very successful in a variety of problems, are limited by the decoupling of the problem and the use of inverse kinematics.

Dept. of Computer Science, Iowa State University, Ames, IA 50011 USA. E-mail: yakeyj@cs.iastate.edu

Dept. of Computer Science, Iowa State University, Ames, IA 50011 USA. E-mail: lavalle@cs.iastate.edu

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

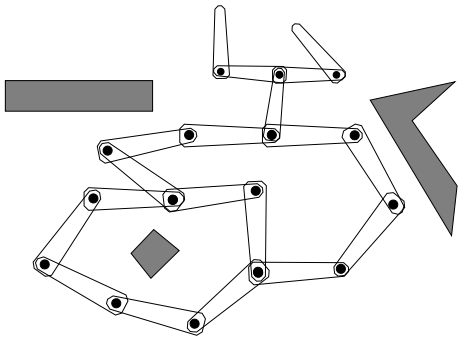

Fig. 1. We investigate path planning for linkages that have closed kinematic chains and must avoid static obstacles.

Regrasping is needed because one or more of the manipulators often attain a singular configuration [52]. The ability to plan for closed linkages eliminates the need of inverse kinematics solutions and could the 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 for *reconfigurable* robots. Typically, this type of robot is composed of multiple, independent robots that can connect and disconnect from one another [37], [54], [63]. This results in their ability to dynamically reconfigure themselves, depending on the current situation for the robot. The ability to change their connectivity gives reconfigurable robots multiple modes of locomotion [37], [63]. Closed linkages often occur during locomotion or reconfiguration of the robots, and such situations could be simplified with the addition of a planner for closed linkages.

Many of the concepts used in path planning for robotics can also be applied to computer graphical animation [11]. Human-like characters can naturally be modeled as linkages. It is desirable to automate the lengthy and complicated task of animating these models, for which motion planning techniques are well suited [64]. 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 [34], 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. To reduce the expense of engineering new products, computer simulations are often executed to determine the quality of a design. Motion planning algorithms have been

used to achieve this end [16]. For designs that include closed linkages, a corresponding planner could automate testing, and potentially avoid constructing physical prototypes.

Our work was first inspired by the need in computational chemistry to query databases of flexible molecules that satisfy a given pharmacophore [42]. A pharmacophore is a set of atoms (or more generally features) of the molecule whose relative spatial positions are constrained [17]. Drug design involves finding molecules that contain a pharmacophore while maintaining a low energy configuration and a legal chemical structure (e.g., no bonds are 'broken' for the molecule to 'stretch' and satisfy the pharmacophore). Molecules often have closed loops (rings) and are straightforward to model as linkages. Also, pharmacophore constraints define closed loops when, for example, they impose distance constraints between two atoms. Besides our earlier work in finding molecules that satisfy pharmacophoric constraints [42], related work on finding configurations of closed rings were presented in [19]. In that paper, two methods were described to algebraically express the geometry of 6-atom ring molecules (cyclic molecules represented with six linkages) so that their configurations could be enumerated. Extensions were also proposed for the 5 and 7 atom rings, which are more difficult to solve. The work in this paper, although different in spirit, could benefit in part from the results in [19] for 5-, 6-, or 7-linkage chains.

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, it is always assumed that a parameterization of the configuration space is available. For example, for a rigid body in $\mathbb{R}^3$, the configuration space is often parameterized by three real coordinates for translation and one quaternion for rotation, resulting in the manifold $\mathbb{R}^3 \times P^3$. If closure constraints exist, a parameterization is usually not available (except for special mechanisms), and the set of valid configurations is generally not even a manifold.[1]

In this paper we do not develop a new method for planning for systems with closed kinematic chains. Instead, we propose a number of primitives that can be used to extend randomized path planners to handle closed kinematic chains and we demonstrate the application of our primitives to existing planners. Our work is motivated by the fact that it is computationally prohibitive to compute the exact topological structure of the set of valid configurations when complex mechanisms with closed loops are considered [19]. Our work is also based on the belief that, among existing approaches, randomized techniques offer the most promise to solve efficiently high-degree-of-freedom problems with closed kinematic chains. Given the success of randomized path planning techniques [2], [6], [9], [15], [23], [29], [31], [33], [40], [41], [43], [48], [59] at addressing problems without closure constraints, it seems sensible to introduce primitives that would allow these existing techniques to be extended to the case of closed linkages. Other researchers are moving in this direction too [24]. In this paper, we focus on studying (a) the generation of random free configurations, (b) the generation of incremental paths and (c) the optimization of paths. All above operations are routinely performed by randomized planners and their study could lead to the development of randomized methods for all types of linkages.

The paper is organized as follows. After the related literature and the problem definition in Sections II Sections III, the following three sections each introduce a primitive that can be used as a building block in a randomized path planner. More precisely, each can be used to upgrade a component of a standard randomized path planner, to enable to inclusion of closed kinematic chains. Section IV addresses the problem is generating a collection of random samples that satisfy the closure constraints. Section V addresses the problem of generating small motions in the vicinity of a given, valid configuration. Section VI presents a simple technique for optimizing a given path. Section VII illustrates the application of the primitives from Sections IV-VI in two different randomized path planners, a version of a PRM [33] and a version of an RRT-based planner [41], [43]. We show experimental results for several challenging problems. Finally, conclusions are presented in Section VIII.

II. Related Literature

A. Kinematics

One of the difficulties when planning for closed chains is the difficulty of computing inverse kinematics. We provide some background. *Forward kinematics* determines the location in the world of a linkage when given a specific *configuration* defining its position and orientation. The *inverse kinematic* problem deals with finding a configuration that will put a point of the linkage (usually an end-effector, in the context of robotic manipulators) at a specified location in the world. Typically, the solution to an inverse kinematics problem will not be unique. If the linkage is *redundant* (i.e., it has more degrees of freedom than the minimum needed to complete its task), there will be an infinite number of solutions [12], while *nonredundant* linkages will have a finite number of solutions to the inverse kinematics problem. This topic is of interest because a closed linkage can be modeled as a special solution to the inverse kinematics problem, where each loop is considered as an open linkage with its end-effector permanently attached to another joint of the loop, or a point in space.

There has been a large body of research dedicated to investigating the inverse kinematics problem. Analysis of the topological properties of the inverse kinematics for redundant manipulators were discussed in [4], [12]. Specifically, it was shown that the infinite number of solutions for a planar or spatial inverse kinematic problem can be grouped into a finite number of disjoint, continuous *self-motion manifolds*. Each of these self-motion manifolds is a set of configurations that correspond to a self-motion of

[1] Even though it can be expressed as a stratification of manifolds [14], parameterizations of the strata are still unknown.

a manipulator, which is a continuous displacement of the manipulator's joints that keeps the end-effector stationary. Many algorithms have been developed to find solutions for the inverse kinematics problem. An algorithm was presented in [47] for computing the inverse kinematic solution for a $6R$ (six revolute joints) serial manipulator in $\mathbb{R}^3$. The number of joints and the dimension of the workspace are equal in this problem, causing the system of equations that describe the constraints to be neither underconstrained nor overconstrained. The main contribution of [47] is an algorithm to reduce the inverse kinematics problem to an eigenvalue problem using matrix operations, which can then be solved efficiently. An interesting result for $6R$ manipulators is a bound of at most 16 unique solutions to the inverse kinematics problem, regardless of the manipulator's geometry [57]. An algebraic formulation of the problem and a collection of algorithms for its solution are presented in [19] for 5, 6 and 7 linkage chains. An iterative algorithm for moving the end-effector of a manipulator along a given trajectory using numerical methods was presented in [61], where the trajectory is specified as a sequence of points in the world that the end-effector is to follow.

Another area of research in kinematics is the development of *parallel manipulators*, which are closed-loop manipulators having multiple chains of links connecting an end-effector to a fixed base. Hence parallel manipulators have closed linkages. A good example is the Stewart platform, which is composed of two rigid bodies in $\mathbb{R}^3$ that are connected by a number of prismatic joints (usually 6). The kinematics for this manipulator are well known. Parallel manipulators have a close relationship with mechanism design. Usually only closed linkages with a small number of links (4 or 5) are considered in planar mechanism design because they only need a single actuator to control the entire linkage. Merlet [50] provided a method to express the forward kinematics for all architectures of planar fully parallel manipulators, which are rigid platforms with three parallel chains of links connected to them, each chain having three 1-dof joints (either revolute or prismatic, with only one of the joints being actuated). In a related paper [51], algorithms to determine various workspaces of these planar parallel manipulators were presented. Each of these workspaces defines the region of the world that a certain reference point on the manipulator can reach, given different constraints on the orientation of a linkage's platform.

All previous research on the topic of computing kinematics indicates that inverse kinematics is a very hard computational problem. In our work, we develop randomized techniques to maintain closed chains.

B. Path Planning

In general, the constraints imposed by a closed linkage form an algebraic variety and in principle complete planners such as [58], [14], [8] could be used. For example, a general algorithm to solve the motion planning problem by expressing constraints as semialgebraic sets was developed by Schwartz and Sharir [58]. Varieties can be thought as a semialgebraic sets where all the algebraic expressions are restricted to being equal to zero. However, the running time of the best known algorithm is exponential in the dimension of the configuration space [14]. Unfortunately, the high computational complexity of all of the above algorithms and our desire to address motion planning for problems with high degree of freedom makes them too prohibitive for practical use.

This has led to the development of randomized path planning methods, which produced good results in a variety of difficult problems involving robots with open kinematic linkages. Several randomized planners exist, and can be generally be categorized as *single-query* or *multiple-query* approaches. Single-query approaches are designed to quickly solve a single problem without preprocessing, and multiple-query approaches perform significant preprocessing of a given robot and environment to enable numerous initial-goal query pairs to be solved efficiently. Among the single-query planners lies the Randomized Path Planner (RPP) [5], which was one of the first highly successful randomized planners. It solved problems for robots with more than 60 degrees of freedom [7], [34]. The planner uses a potential field as a guidance towards the goal, and random walks to escape local minima. Another interesting approach was presented in [49], [48] – the Ariadne's clew algorithm. Considering the initial configuration as a landmark, the algorithm incrementally builds a tree of feasible paths as follows. Genetic optimization is used to search for a collision-free path from one of the landmarks to a point as far as possible from previous landmarks. A new landmark is then placed at this point, and a path to the goal configuration is searched. New landmarks are placed until the goal configuration can be connected to the tree. Other randomized planning approaches that grow trees of feasible paths include [29], [30], [44], [43]. The planners in [44], [43] are based on the notion of a Rapidly-exploring Random Tree (RRT).

The Probabilistic Roadmap Planner PRM [32], [33], [53] was one of the first multiple-query planners. It captures the connectivity of the free configuration space by a random network, a roadmap, whose nodes correspond to randomly selected configurations and whose edges are local path segments. The initial version of PRM [33] is considered a multiple-query planner since after the random roadmap has been constructed, multiple initial-goal query pairs can solved efficiently. Several planners have been inspired by PRM. Narrow passages are notoriously difficult to find at random and several planners have developed to address this issue. Creating nodes in narrow passages has been the main motivation of the enhancement step in [32], the generation of nodes near the configuration space obstacles in [3], the penetration of obstacles in [28], the Gaussian sampling in [10], the retraction to the configuration space medial axis in [3], and the use of the workspace medial axis in [13] and [55]. Many recent randomized planners focus on single-query performance although they can be used for multiple queries too. Some of them select the nodes of their roadmaps using a lazy evaluation [9] or a visibility-based selection method [40].

In all of the above planners, primitives such as the generation of random configurations or the creation of local paths are frequently used. One could argue that these primitives form the basis of several randomized planners. That is why in this paper we investigate how these primitives, together with path smoothing, can be extended to handle closed kinematics chains. Our hope is that the newly developed primitives could be used in the randomized methods described above, creating new randomized planners that are suitable for a variety of problems, as their open-chain counterparts are.

In Section VII we demonstrate the generality and utility of our primitives by applying them to extend two existing planners. The first is a multiple-query planner based on the PRM, and the second is a single-query planner based on the RRT. 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 [36], [35]. One is a previous paper of ours [45], which presents a subset of the work in our current paper. The other is a paper by Han and Amato [24]. 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. Both [24] and our work advance the state-of-the-art in using randomized planners for planning for mechanisms with closed loops.

III. Problem Formulation

In this section, a formal definition of a closed linkage will be presented, and the motion planning problem is formulated in the context of these linkages. Mathematical concepts that are commonly used in motion planning will be briefly introduced, and the notation used throughout the remainder of this paper will be standardized as well.

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*[2]. 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[3], which are shown in Figure 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. It is interesting to note that if G_M is a tree, then all configurations of $\mathcal{M}$ yield an acceptable position and orientation for each of the links, if we ignore any collisions. This is due to the fact that there are no loops that would create restrictions on the valid configurations of a linkage.

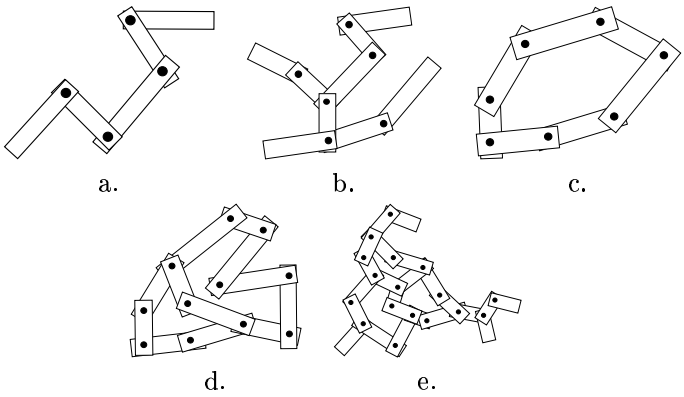

Fig. 2. Nomenclature: (a) Open chain linkage (b) open linkage (c) closed chain linkage (d) closed linkage and a (e) compound linkage.

B. Kinematics

To consider the kinematics of $\mathcal{M}$, we use standard parameterization techniques to express the *configuration* of $\mathcal{M}$ as a vector, q, of real-valued parameters. The configuration is used to uniquely specify the position and orientation of $\mathcal{M}$ in $\mathcal{W}$. Let $\mathcal{M}(q)$ denote the transformation of $\mathcal{M}$ to the configuration in the world given by q.

To determine the position and orientation of each L_i with respect to L_{i-1} in an open chain of links $L_1, L_2, \cdots, L_i$, we will use a "homogeneous" transform matrix T_i that encodes both translation and rotation in $\mathcal{W}$. Each L_i will have associated with it an independent origin $\mathcal{O}_{L_i}$ and a coordinate frame $\mathcal{F}_{L_i}$, which will also be independent of $\mathcal{W}$'s coordinate frame. Then, since the linkage is open and

[2]We use the more general definition of linkage that includes open and closed kinematic chains [20], rather than a linkage that contains only closed chains [25].

[3]Note that these classes deviate from the standard terminology used in mechanism design [25]. Our intent was for a *chain* to imply linearity of the linkage, and for *closed* to mean that the linkage contains no unary links.

G_M will be a tree, we can choose one of the links to be the root link. It is often convenient to assume that this link is attached to a point in the world, no longer allowing $\mathcal{M}$ to freely translate in the world. This point of attachment can be thought of as a zero length link in our linkage, which we will denote as L_0. These transformations pertain to open chains only. To calculate the kinematics of an open linkage, each chain of links in G_M that begins at a leaf of the tree and ends in the root vertex can be considered as an independent open chain, effectively decomposing the linkage into multiple open chains.

For the planar case ($N = 2$) we have the following transformation matrix:

$$T_i = \begin{pmatrix} \cos q_i & -\sin q_i & \ell_{i-1} \\ \sin q_i & \cos q_i & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{1}$$

in which ℓ_i is the distance between the joints for link L_i and $\ell_0 = 0$ (the length of the artificial link L_0). The parameters q_i assume that each of the joints connecting the links is revolute. Since a prismatic joint is essentially a variable length link, this transform matrix can be used to represent this type of joint by varying ℓ_i. The transformation of any link is given by the product, $\forall (x, y) \in L_i$ (with respect to L_i's coordinate frame),

$$\begin{pmatrix} x(q) \\ y(q) \\ 1 \end{pmatrix} = T_1 T_2 \cdots T_i \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \tag{2}$$

This transformation rotates each link L_i by the parameter q_i and translates L_i to its position in the chain $L_1, L_2, \ldots, L_i$.

As in the 2D case, a homogeneous transformation matrix can be defined for the 3D case by using the Denavit-Hartenburg parameterization [21], [26]:

$$T_i = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & \ell_{i-1} \\ \sin\theta_i \cos\alpha_{i-1} & \cos\theta_i \cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1} d_i \\ \sin\theta_i \sin\alpha_{i-1} & \cos\theta_i \sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3}$$

in which there are four parameters involved: ℓ_i, α_i, d_i and θ_i. Assuming once again that each of the joints in the linkage is revolute, then the joint connecting links L_i and L_{i-1} will have an axis of rotation. The four parameters can be put into two groups, denoting a length and angle with respect to either L_i or the axis of rotation. The parameter ℓ_i denotes the length of L_i as before, and α_i is the angle between L_i and L_{i-1}, with respect to L_i. Along the axis of rotation, d_i is the distance between L_i and L_{i-1}, and θ_i is the angle between them. In the case of a prismatic joint, either ℓ_i or d_i can be varied. The Denavit-Hartenburg parameterization is capable of representing the other lower-pair joints by creating additional links and setting certain parameters equal to 0. To compute the kinematics for the entire linkage, a product similar to the one in Equation 2 can be used.

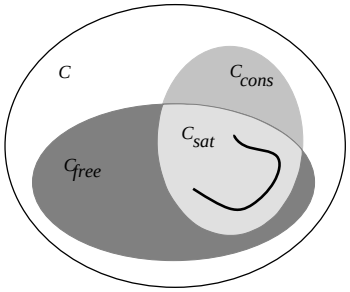

Fig. 3. The relationship between $\mathcal{C}_{free}$, $\mathcal{C}_{cons}$, and $\mathcal{C}_{sat}$.

C. Expressing Closure Constraints

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 IV-A. In general, if n is the dimension of $\mathcal{C}$, then $m < n$. Let $\mathcal{C}_{cons} \subset \mathcal{C}$ be defined as:

$$\mathcal{C}_{cons} = \{q \in \mathcal{C} \mid \forall f_i \in \mathcal{F}, f_i(q) = 0\}, \tag{4}$$

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 the world have a set of obstacles $\mathcal{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_{n_b}\}$, which are each a closed but not necessarily bounded 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:

$$\mathcal{C}_{free} = \{q \in \mathcal{C} \mid (\mathcal{M}(q) \cap \mathcal{B} = \emptyset) \wedge (\forall L_i, L_j \in \mathcal{M}(q), L_i \cap L_j = \emptyset)\}, \tag{5}$$

where L_i, L_j are nonconsecutive links.

In addition to the usual complications of path planning for articulated linkage 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 set of configurations satisfying both closure and collision constraints, pictured in Figure 3.

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 [39], 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 [62]. Because of the nature of these closure constraints, we will assume that we have no *a priori* knowledge of a parameterization for the variety.

We previously illustrated how a parameterization for S^1 is created, and we were able to use this parameterization to reduce the dimension of the composite configuration space. However, since we have no known parameterization for the variety defining $\mathcal{C}_{cons}$, we can not reduce the dimensionality of $\mathcal{C}$. Herein lies the difficulty of path planning for closed linkages. 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 $\mathcal{C}_{cons}$ to simplify path planning. This tolerance will be the subject of Section IV-A.

Finding a Path. We now come to the definition for the path planning problem. 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 $\mathcal{C}_{sat}$.

D. A Specific 2D Model

The model described up to this point is fairly general, and is used to express the primitives that we introduce. We now give a specialized model that will facilitate some of the later discussion and will be used in our simulation experiments.

Suppose the following:

1. $\mathcal{L}$ is a collection of line segments in a 2D world.
2. Joints only attach links at their endpoints.
3. Every joint is revolute.
4. There are joint limits (e.g., joints are not allowed to rotate into the range $0 \pm \delta$, where δ is a parameter for the joint limit).
5. One of the joints attaches a link to the origin $(0,0)$ in the world, $\mathcal{W}$.
6. The obstacle region is polygonal.

IV. Generating Random Samples

One of the most basic operations in many randomized planners is the construction of random configurations. This is particularly true of multiple-query approaches. For example, the basic PRM approach 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 $\mathcal{C}_{sat}$.

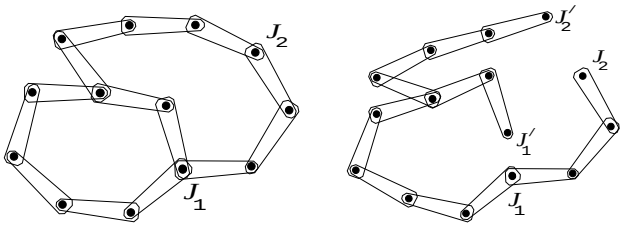

Fig. 4. An example of breaking cycles in a linkage.

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 4. In $\mathcal{M}'$, the configuration of any link can be determined by applying the kinematic equations from Section III-B 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 III, 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:

$$e(q) = \sum_{k \in B} \|J_k(q) - J'_k(q)\|^2. \tag{6}$$

Alternatively, the maximum (or any L^p norm) can be used to combine the error from each broken loop. This error function allows us to redefine $\mathcal{C}_{cons}$ as follows:

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

Figure 5 is an example of a linkage in which there are two broken kinematic loops, where e_1 and e_2 are the gaps that need their Euclidean distance reduced. Since the equality constraints that define the closure of the kinematic loops are very restrictive, we allow a specified real-valued tolerance $\epsilon > 0$ to determine when the closure constraints are satisfied. The tolerance also allows us to reduce the effects of numerical error on our solutions. This gives us new definitions for $\mathcal{C}_{cons}$ and $\mathcal{C}_{sat}$ that take ϵ into consideration:

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

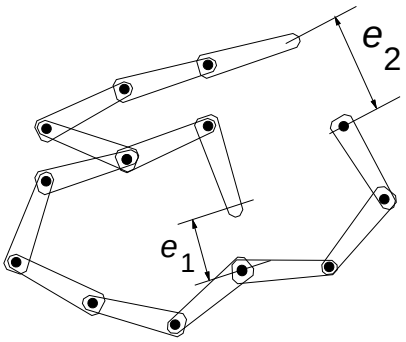

Fig. 5. Each kinematic loop is broken, and $e(q)$ is measured in terms of Euclidean distances.

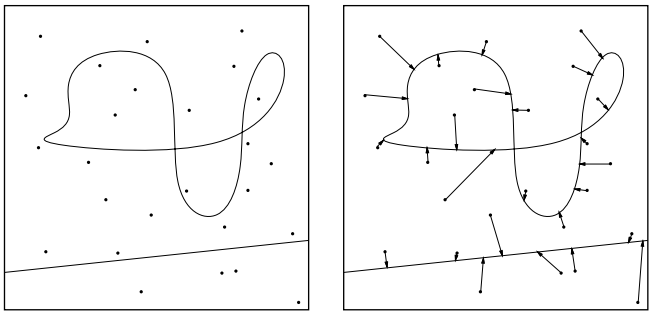

Fig. 6. (a) The curves depict $\mathcal{C}_{cons}$, and configurations are chosen at random in $\mathcal{C}$. (b) Randomized error minimization is performed on the samples to force as many as possible onto $\mathcal{C}_{cons}$.

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

By using the ϵ tolerance, we allow some freedom for the randomized algorithms as they travel on the constraint surface. Without the 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 6 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 7 gives pseudocode for a randomized descent technique that iteratively attempts to reduce the error function, $e(q)$ from Section IV-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 [60].

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. Rather than compute a complicated gradient of $e(q)$, any random configuration q' in which $e(q') < e(q)$ is kept. This was observed in [5] 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 [56] nonlinear optimization procedure could be used instead of randomized descent, but one must be careful not to introduce an unwanted deterministic bias on the solutions.

```
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. 7. An algorithm that iteratively attempts to reduce the kinematic error of a linkage.

C. A Computed Example

As an example, we tested the uniformity of a sample of nodes obtained through our randomized algorithms. Obstacles were placed in the world so that there would be many distinct connected components in $\mathcal{C}_{sat}$. We then generated a roadmap for this world and observed the various connected components to determine whether they were all represented. In Figure 8, 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.

V. 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 $\mathcal{C}_{cons}$ or $\mathcal{C}_{sat}$. Given a configuration $q \in \mathcal{C}_{sat}$, the task is to

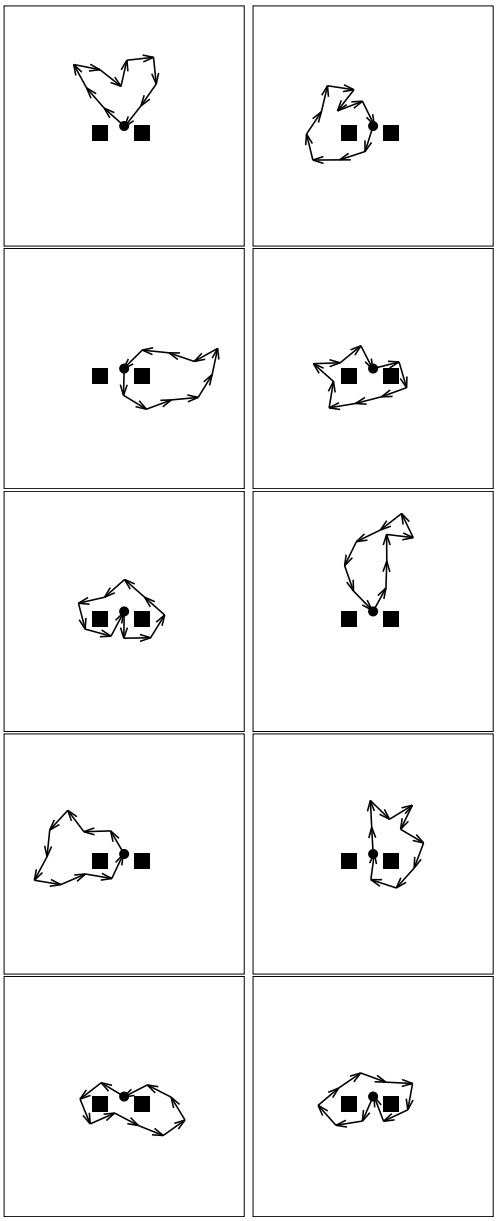

Fig. 8. All ten components were found in a 2D world that contains obstacles.

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 $q \in \mathcal{C}_{cons}$, which is depicted in Figure 9. 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

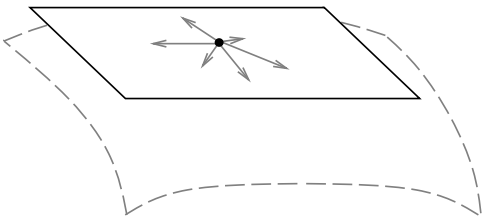

Fig. 9. Tangent space for a configuration q on $\mathcal{C}$.

differential configuration vector dq lies in the tangent space of a constraint $f_i(q) = 0$ if

$$\frac{\partial f_i(q)}{\partial q_1} dq_1 + \frac{\partial f_i(q)}{\partial q_2} dq_2 + \cdots + \frac{\partial f_i(q)}{\partial q_n} dq_n = 0. \quad (7)$$

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

$$\begin{pmatrix} \frac{\partial f_1(q)}{\partial q_1} & \frac{\partial f_1(q)}{\partial q_2} & \cdots & \frac{\partial f_1(q)}{\partial q_n} \\ \frac{\partial f_2(q)}{\partial q_1} & \frac{\partial f_2(q)}{\partial q_2} & \cdots & \frac{\partial f_2(q)}{\partial q_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_m(q)}{\partial q_1} & \frac{\partial f_m(q)}{\partial q_2} & \cdots & \frac{\partial f_m(q)}{\partial q_n} \end{pmatrix} \begin{pmatrix} dq_1 \\ dq_2 \\ \vdots \\ dq_n \end{pmatrix} = \mathbf{0}. \quad (8)$$

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 8. Using singular value decomposition (SVD) [22], [56], [46], this under-constrained homogeneous system can be efficiently converted into a more useful form. SVD will take any $m \times n$ matrix A and decompose it into three new matrices: an $m \times n$ column orthogonal matrix U, an $n \times n$ diagonal matrix W, and the transpose of an $n \times n$ orthogonal matrix V. These matrices are constructed such that $A = U \cdot W \cdot V^T$. The nonnegative values $w_1 \geq w_2 \geq \ldots \geq w_n$ that lie on the diagonal of W are the *singular values* for the homogeneous system of equations. Since $m < n$ for Equation 8, the last $n - m$ singular values will be equal to 0, although some of the first m values could be 0 as well due to degeneracies in the m equations. For all $w_i = 0$, the corresponding columns i in V will form an orthonormal basis for our tangent space, meaning that any linear combination of these vectors will lie in our tangent space. This enables our algorithm to follow the tangent space and only generate the $n - m$ random scalar displacements needed for the linear combination, effectively reducing the size of the space we need to searching for valid local motions. This technique usually 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, we need an efficient means of computing 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. As discussed in Section III-B, 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 kinematics of these open chains needs to be computed using the two or three dimensional transformation matrices from Section III-B. Below we will consider how the partial derivatives of these open chains are efficiently computed for $\mathcal{W} \subseteq \mathbb{R}^2$. First, we need to reformulate the homogeneous transform matrix in Equation 1. We do this by defining recursive formulas to compute the x and y positions for the origin of each link in $\mathcal{W}$:

$$X_n = \cos(q_n)X_{n-1} - \sin(q_n)Y_{n-1} + \ell_{n-1}, \tag{9}$$

where $X_0 = x$ and

$$Y_n = \sin(q_n)X_{n-1} + \cos(q_n)Y_{n-1}, \tag{10}$$

where $Y_0 = y$. In Equations 9 and 10, i represents the index of the link in each open chain of 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 give us 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 will be 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:

$$\frac{\partial X_n}{\partial q_i} = \begin{cases} \cos(q_n)\frac{\partial X_{n-1}}{\partial q_i} - \sin(q_n)\frac{\partial Y_{n-1}}{\partial q_i} & i < n \\ -\sin(q_n)X_{n-1} - \cos(q_n)Y_{n-1} & i = n \end{cases} \tag{11}$$

$$\frac{\partial Y_n}{\partial q_i} = \begin{cases} \sin(q_n)\frac{\partial X_{n-1}}{\partial q_i} - \cos(q_n)\frac{\partial Y_{n-1}}{\partial q_i} & i < n \\ \cos(q_n)X_{n-1} - \sin(q_n)Y_{n-1} & i = n \end{cases} \tag{12}$$

Now that we have the algebraic equations corresponding to the needed derivatives, we must now consider the task of evaluating them for a given q. Since these derivatives need to be computed quite often, it must be done efficiently. By using the recursive linkage of these equations to our advantage, memoized dynamic programming [18] can be used to efficiently perform this computation. 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 9 and 10:

$$X_n = \frac{\partial Y_n}{\partial q_n} + \ell_{n-1}, \tag{13}$$

$$Y_n = -\frac{\partial X_n}{\partial q_n}. \tag{14}$$

Using memoized dynamic programming, we can eliminate the need to recompute values that were previously computed, and we can also avoid the overhead of using the naive recursive implementation that would result in a large number of function calls in our implementation. The procedures outlined above to reformulate the kinematics and express the partial derivatives for a linkage can easily be extended to the three dimensional case. The Denavit-Hartenburg parameterization may be expressed as a recursive function, and the derivatives can be derived in a manner similar to the two dimensional case.

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

B. Connecting Nearby Configurations

Some randomized path planners, such as the PRM, are required generate a path that connects two 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 model in Section III-D, we will use the following

$$\rho(q, q') = \sum_{i=1}^{n_j} \|q_i - q'_i\|, \tag{15}$$

with the understanding that each $q_i \in S^1$. An alternative is to compute the sum of squares of the Euclidean displacements for all of the joints in $\mathcal{J}$. This metric offers some advantages, but is often too costly for frequent computation [33].

The algorithm in Figure 10 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 gradient descent is shown in Figure 11, where the random path travels between two configurations while staying within $\tilde{\mathcal{C}}_{sat}$.

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

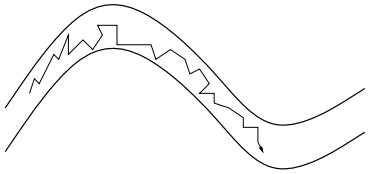

Fig. 11. A randomized descent on ρ can be performed inside $\tilde{\mathcal{C}}_{cons}$.

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. Although, once a path has been generated, the path optimization algorithms from Section VI 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 use the model given in Section III-D to demonstrate the method. The first experiment we performed compared the random sampling versus tangent space sampling when generating a random neighbor of a configuration. Our experiment was conducted by generating 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 12 compares the two sampling methods for an 8-link closed chain linkage, and Figure 13 is a comparison for a 7-link closed linkage that has two loops (the linkage is shown in Figure 15).

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 distance traveled per step speeds connection of two configurations. Computing the tangent space samples is more

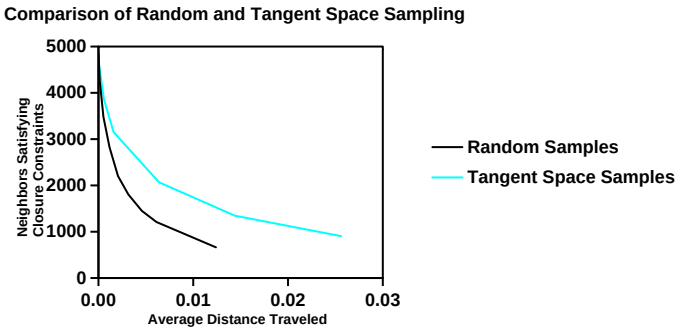

Fig. 12. Comparison between random and tangent space sampling for random neighbor generation of an 8-link closed chain linkage.

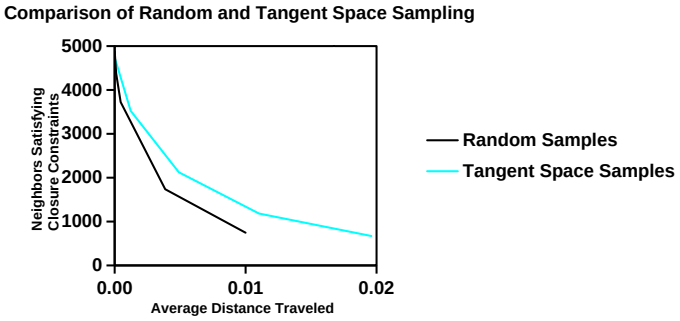

Fig. 13. Comparison between random and tangent space sampling for random neighbor generation of a 7-link, 2-loop closed linkage.

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.

VI. Optimizing Paths

Generally, the paths generated by randomized algorithms are not very smooth. In many applications it is desirable to have paths that are not jagged; thus, a post-processing step can be used to improve the quality of our paths. While improving the quality of paths is straightforward for basic path planning, path optimization is more complicated for closed linkages.

The paths produced by randomized algorithms are assumed to be of the following form:

1. The path is represented as a discretized sequence of points that lie in $\tilde{\mathcal{C}}_{sat}$, $(v_1, \ldots, v_{n_v})$, where n_v is the number of points in the path.
2. The first point in the sequence is q_{init} and the last is q_{goal}.
3. For each pair of consecutive points in the sequence, v_i and v_{i+1}, $\rho(v_i, v_{i+1}) < d_{max}$, where d_{max} is the maximum distance between configurations in the path.

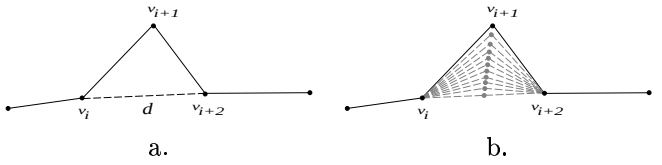

Fig. 14. Two methods for path optimization: a) point removal; b) barycentric warping.

We describe two approaches for improving the quality of the paths produced by the randomized algorithms, one that reduces the length of the path and one that "warps" the path to reduce its curvature. Typically, a combination of the two methods will provide the best results when they are repeatedly performed on a path.

Point removal. Along the path, every triple v_i, v_{i+1}, and v_{i+2} is analyzed. We compute the distance $d = \rho(v_i, v_{i+2})$ and determine whether $d < d_{max}$. Intuitively, when $d < d_{max}$ the local portion of the path with configurations v_i, v_{i+1}, and v_{i+2} is not taking the shortest route from v_i to v_{i+2}. Because of this, we can delete the configuration v_{i+1} from our path without violating the maximum distance between two consecutive configurations in a path. This effectively creates a new, straight line, local path from v_i to v_{i+2}. Collision detection can be ignored for this new local path because we have already made the assumption that there are no collisions between points in the sequence, as long as the condition $d < d_{max}$ is maintained. When this operation is performed over the entire path, a less jagged path is the result. Figure 14.a shows an example of point removal for a path. To work properly, this technique requires multiple passes over the path to ensure that all extra configurations have been removed from the path.

Barycentric warping. In this approach to path optimization, we will once again be concerned with the triple v_i, v_{i+1}, and v_{i+2}, but this method will attempt to incrementally move v_{i+1} closer to the straight line between v_i and v_{i+2}. Barycentric warping is most useful when $d > d_{max}$ because this method can reduce the length of the path without removing any points. We use *barycentric coordinates* [27] to perform this warping, or more formally, an iterative interpolation of v_{i+1}. Given a set of points $A = \{a_0, a_1, \ldots, a_k\}$ in Euclidean n-space, a k-dimensional hyperplane H exists containing A. We can define barycentric coordinates as the real numbers $\beta_0, \beta_1, \ldots, \beta_k$ such that:

$$\sum_{i=0}^{k} \beta_i = 1 \quad \text{and} \quad h = \sum_{i=0}^{k} \beta_i a_i,$$

in which h is a point with respect to A lying in H. Intuitively, the barycentric coordinates are a series of weights that allow us to interpolate a new point h from A. This weighting is used to facilitate the warping of v_{i+1}. To compute the barycentric warping, we will then let $A = \{v_i, v_{i+1}, v_{i+2}\}$, making H a two dimensional hyperplane. Since we will want the interpolated point to be in the triangle specified by v_i, v_{i+1}, and v_{i+2} on H, then $\beta_0, \beta_1, \beta_2 \geq 0$ will always be true.

The barycentric warping algorithm iteratively warps v_{i+1} towards the straight line between v_i and v_{i+2} by incrementally increasing the β_1 weight while setting $\beta_0 = \beta_2 = \frac{(1-\beta_1)}{2}$. An example of this warping is shown in Figure 14.b. The parameter $\Delta\beta$ determines the change in the weighting of v_{i+1} for each iteration of the warping algorithm. The interpolated point that is closest to the line between v_i and v_{i+2} while satisfying the closure and collision constraints will replace point v_{i+1} in the path.

VII. Path Planning Experiments

In this section, we present two path planning methods that were developed by applying the methods introduced in Sections IV-VI. Section VII-A describes an implemented PRM-based planner and shows some computed results. Section VII-B presents an implemented RRT-based planner and several computed examples.

A. PRM Results

The implemented version of PRM is a modification of the planner presented in [33]. 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_SMAPLE 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.

We now present three examples of linkages for which we have computed roadmaps. The first linkage is shown in Figure 15, 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 seven degrees of freedom when the closure constraints are ignored. Each of these constraints removes two degrees of freedom from the linkage, resulting in a total of 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 16. 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. Using a probabilistic roadmap, we created an example of how manipulation tasks with closed linkages may be computed.

Our final example in Figure 17 simulates two planar serial manipulators cooperatively grasping an ob-

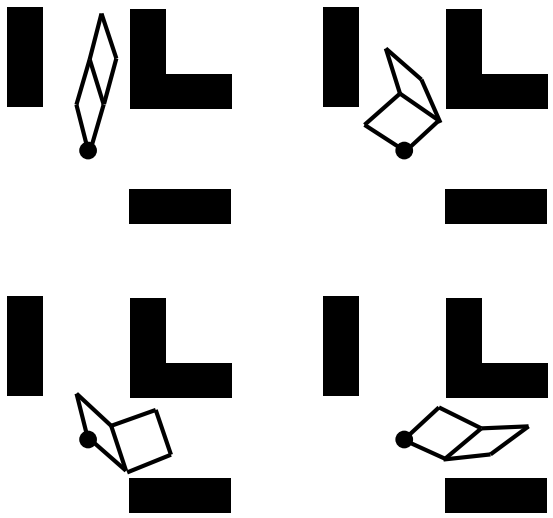

Fig. 15. Snapshots along the path of a closed linkage with two loops.

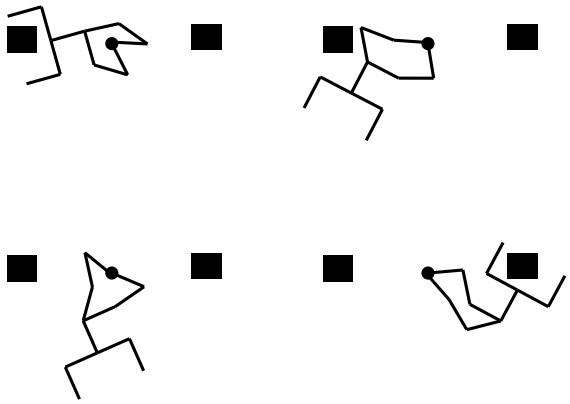

Fig. 16. Snapshots along the path of a manipulator example.

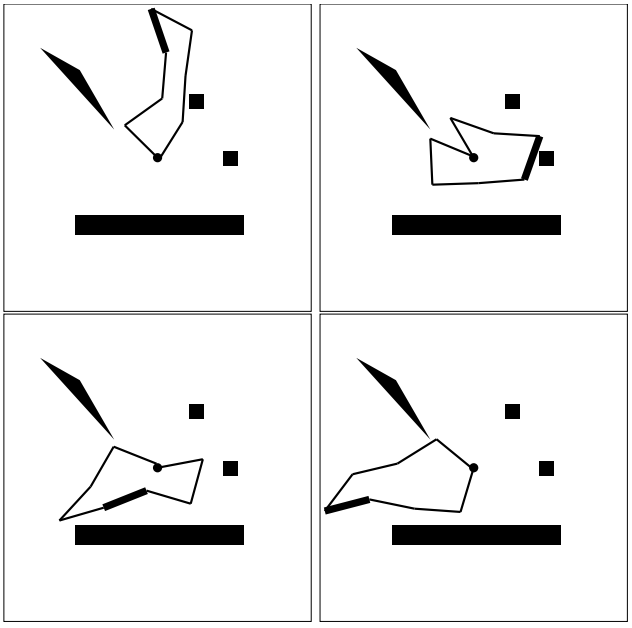

Fig. 17. Two manipulators grasping and moving an object.

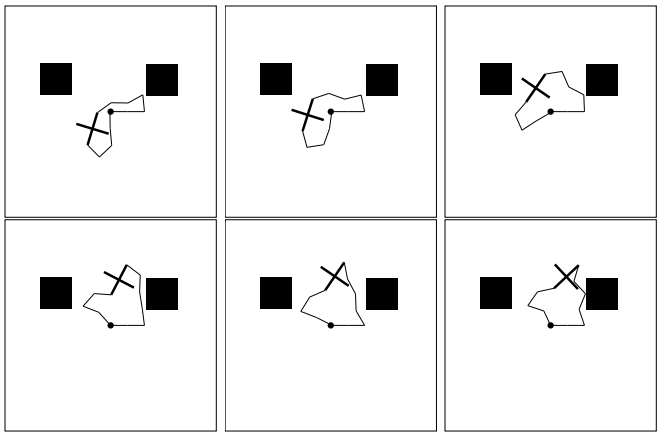

Fig. 18. Two manipulators grasping a cross-shaped object.

ject. 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. All of these examples required several hours for the computation of the roadmap. This extensive computation time is due to the repeated execution of the GENERATE_RANDOM_SAMPLE and CONNECT_CONFIGURATIONS algorithms, which generally are very time consuming. Note that the implemented version of PRM tries to generate a roadmap that captures the components of the free configuration space and this entails a long precomputation time. 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 because a simple graph search is all that is required to compute the remainder of the path. The precomputed roadmap is then stored for later use.

B. RRT Results

The RRT-based planner is a modification of a planner presented in [43]. 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 [41], [43]. 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 18, 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 19, where the "head" of the snake needs to compress so that it may fit through an obstacle.

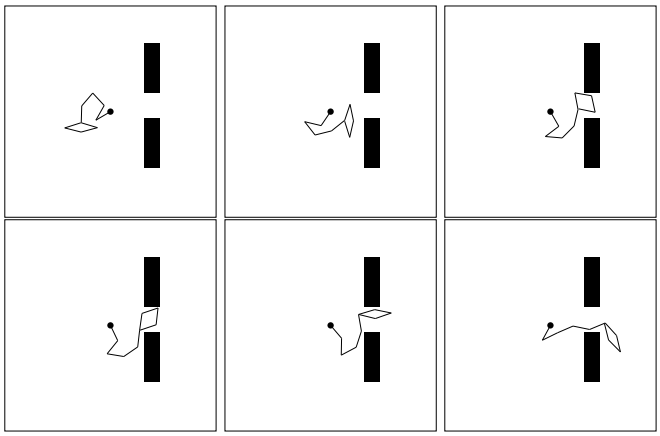

Fig. 19. A snake-like compound linkage example.

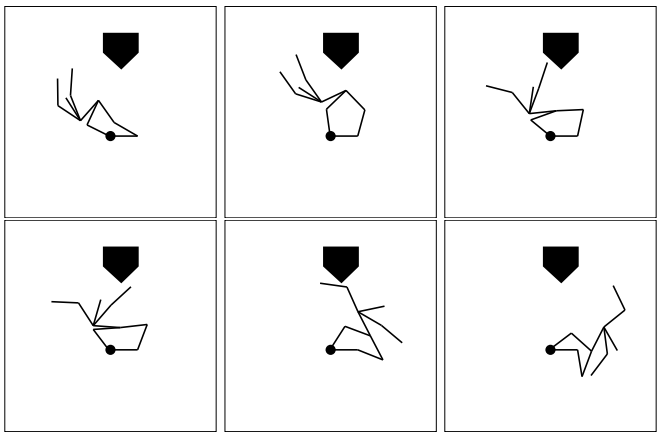

Fig. 20. An 11-link compound linkage example.

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 20 with 9-DOF once the closure constraints have been taken into account. The computation of this example took 888.22 seconds.

VIII. Conclusions

We have introduced a set of primitives that extend the applicability of randomized path planning techniques to a larger class of problems. Our work allows for the development of randomized planners that incorporate both closed and open linkages into their motion planning strategy. Our primitives are general and thus capable of being applied to a large number of problems and randomized path planning methods.

Our work is based on decomposing a closed linkage into multiple open linkages, as well as by defining the notion of a closure constraint upon these new linkages. We use random gradient descent algorithms to force and maintain the closure of these open linkages, leading to algorithms that provide a large computation time improvement over algebraic methods, although at the expense of having a weaker notion of completeness.

Using our new primitives, randomized planning methods were developed by adapting existing randomized algorithms. The PRM approach provides us with a powerful means of characterizing the entire space of valid configurations for the articulated linkage. RRT trees have also been implemented, yielding a fast algorithm to explore a single connected component of $\tilde{\mathcal{C}}_{sat}$. Our implemented PRM excels in applications where multiple path queries are needed, since the entire space is covered with the roadmap. On the other hand, when only single point to point queries are needed, the RRT planner eliminate the overhead of planning in all connected components simultaneously. The choice of a suitable randomized planning method is generally dependent on the application. Both of these approaches lack the ability to decide if a path exists between two given configurations. However, if a path does exist, then the probability that it will be found by these algorithms increases with the number of nodes.

A. Completeness Issues

For the probabilistic roadmap method a notion of probabilistic completeness has been established [33], which states that if a path exists between an initial and goal configuration, the probability that a path is found approaches one as the running time approaches infinity. Both the original PRM and RRT-based planners are probabilistically complete, and we have been careful not to introduce any unwanted biases with the modifications made to the standard probabilistic roadmap planner to ensure that probabilistic completeness is retained.

Since $\mathcal{C}_{cons}$ is an equality constraint we know it has lower dimension than $\mathcal{C}$, but the dimension of $\tilde{\mathcal{C}}_{cons}$ is equal to the dimension of $\mathcal{C}$ because of the inequality constraints (see Section IV-A). The boundaries of the ϵ tolerance for $\mathcal{C}_{cons}$ can actually be considered as obstacle boundaries, and $\tilde{\mathcal{C}}_{cons}$ is a kind of thin passageway. This enables probabilistic completeness to holds for planning in $\tilde{\mathcal{C}}_{sat}$. This property is true for $\tilde{\mathcal{C}}_{sat}$ only, and not $\mathcal{C}_{sat}$. The probabilistic completeness of our algorithm is also dependent on d_{max}, the maximum step size between two intermediate configurations in the encoding of a path.

B. Future Directions

Although our experiments demonstrate that randomized path planners can be extended to closed linkages using our framework, signficant issues remain. Our work in [45] represented the first attempt to develop randomized planners for closed kinematic chains; the current paper is an expansion and extension of that work. Recently, the approach described in [24] has led to better efficiency for some classes of problems. In particular, it was shown in [24] greater efficiency can be obtained by precomputing samples for a linkage in the absence of obstacles, and also by using fast inverse kinematics algorithms for incremental motions.

The reported computation times are encouraging, but it is expected that great improvements could be obtained by adapting more recent randomized planners. In some preliminary experients, we have observed dramatic improve-

ment (over an order of magnitude) in computation time by replacing the RRT-based planner with a bidirectional version that uses the connect heuristic described in [38]. One would also expect signficant improvement in the PRM-based approach by exploiting ideas such as lazy evaluation [9].

Sampling represents another interesting issue that remains for future work. Although it is straightforward to argue that our approach leads to probabilistic complete planners, it remains a considerable challenge to characterize the rate of convergence to a solution or the distribution of samples that fall into $\mathcal{C}_{sat}$. In the spirit of the work in [3], [10], [28], [40], it be possible to develop special sampling techniques that are aimed improving the performance for planning problems that involve closed kinematic chains.

Acknowledgments

We are very grateful to Jean-Claude Latombe and Paul Finn for our work in [42] 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, NSF CISE SA1728-21122N, 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] N. Amato, B. Bayazit, L. Dale, C. Jones, and D. Vallejo. Obprm: An obstacle-based prm for 3d workspaces. In P. Agarwal, L. Kavraki, and M. Mason, editors, *Robotics: The Algorithmic Perspective*. AK Peters, 1998.

[3] N.M. Amato, O.B. Bayazit, L.K. Dale, C. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In P. K. Agarwal, L. E. Kavraki, and M. Mason, editors, *Robotics: The Algorithmic Perspective*, pages 630–637. AK Peters, 1998.

[4] D. R. Baker and C. W. Wampler II. On the inverse kinematics of redundant manipulators. *Int. J. Robot. Res.*, 7(2):3–21, 1988.

[5] 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.

[6] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *Int. J. Robot. Res.*, 10(6):628–649, December 1991.

[7] J. Barraquand and J.C. Latombe. Robot motion planning: A distributed representation approach. *Int. J. of Rob. Research*, 10:628–649, 1991.

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

[9] R. Bohlin and L.E. Kavraki. Path planning using lazy prm. In *Proc. IEEE Int. Conf. on Rob. & Aut.*, 2000.

[10] V. Boor, M.H. Overmars, and F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proc. IEEE Int. Conf. on Rob. & Aut.*, pages 1018–1023, 1999.

[11] 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.

[12] J. W. Burdick. On the inverse kinematics of redundant manipulators: Characterization of the self-motion manifolds. In *IEEE Int. Conf. Robot. & Autom.*, pages 264–270, 1989.

[13] L. Kavraki C. Holleman. A framework for using the workspace medial axis in PRM planners. In *Proc. IEEE Int. Conf. on Rob. & Aut.*, 2000.

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

[15] D. Challou, D. Boley, M. Gini, and V. Kumar. A parallel formulation of informed randomized search for robot motion planning problems. In *IEEE Int. Conf. Robot. & Autom.*, pages 709–714, 1995.

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

[17] D. E. Clark, G. Jones, P. Willett P. W. Kenny, and Glen. Pharmacophoric pattern matching in files of three-dimensional chemical structures: Comparison of conformational searching algorithms for flexible searching. *J. Chem. Inf. Comput. Sci.*, 34:197–206, 1994.

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

[19] I. Z. Emiris and B. Mourrain. Computer algebra methods for studying and computing molecular conformations. Technical report, Institut National De Recherche En Informatique Et En Automatique, Sophia-Antipolis, France, 1997.

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

[21] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee. *Robotics: Control, Sensing, Vision, and Intelligence*. McGraw-Hill, New York, 1987.

[22] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, second edition, 1989.

[23] L. Gouzénes. Strategies for solving collision-free trajectories for mobile and manipulator robots. *Int. J. Robot. Res.*, 3(4):51–65, 1984.

[24] L. Han and N.M Amato. Kinematics-based probabilistic roadmap method for closed chain systems. In *Wokshop on the Algorithmic Foundations of Robotics*, 2000.

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

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

[27] J. G. Hocking and G. S. Young. *Topology*. Dover Publications, New York, NY, 1988.

[28] D. Hsu, L.E. Kavraki, J.C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In P. Agarwal, L. Kavraki, and M. Mason, editors, *Robotics: The Algorithmic Perspective*, pages 141–154. A K Peters, 1998.

[29] D. Hsu, R. Kindel, J.C Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. In *Wokshop on the Algorithmic Foundations of Robotics*, 2000.

[30] D. Hsu, J. C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *Proc. IEEE Int. Conf. on Rob. & Aut.*, pages 2719–2726, 1997.

[31] Y. K. Hwang and N. Ahuja. A potential field approach to path planning. *IEEE Trans. Robot. & Autom.*, 8(1):23–32, February 1992.

[32] L. Kavraki and J.-C. Latombe. Randomized preprocessing of configuration space for path planning. In *IEEE Int. Conf. Robot. & Autom.*, pages 2138–2139, 1994.

[33] 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.

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

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

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

[37] 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.

[38] J. J. Kuffner and S. M. LaValle. RRT-connect: An efficient

approach to single-query path planning. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, 2000.

[39] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[40] J.P. Laumond and T. Siméon. Notes on visibility roadmaps and path planning. In *Wokshop on the Algorithmic Foundations of Robotics*, 2000.

[41] 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.

[42] 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.

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

[44] S.M. LaValle and J.J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int. Conf. on Rob. & Aut.*, pages 473–479, 1999.

[45] S.M. LaValle, J.H. Yakey, and L.E. Kavraki. A proababilistic roadmap approach for systems with closed kinematic chains. In *Proc. IEEE Int. Conf. on Rob. & Aut.*, pages 1671–1676, 1999.

[46] A. A. Maciejewski. Dealing with the ill-conditioned equations of motion for articulated figures. *IEEE Computer Graphics & Applications*, 10(3):63–71, 1990.

[47] D. Manocha and J. Canny. Real time inverse kinematics of general 6R manipulators. In *IEEE Int. Conf. Robot. & Autom.*, pages 383–389, Nice, May 1992.

[48] E. Mazer, J.M. Ahuactzin, and P. Bessière. The Ariadne's clew algorithm. *J. of Art. Intelligence Research*, 9:295–316, 1998.

[49] E. Mazer, G. Talbi, J. M. Ahuactzin, and P. Bessière. The Ariadne's clew algorithm. In *Proc. Int. Conf. of Society of Adaptive Behavior*, Honolulu, 1992.

[50] J.-P. Merlet. Direct kinematics of planar parallel manipulators. In *IEEE Int. Conf. Robot. & Autom.*, pages 3744–3749, 1996.

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

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

[53] M. Overmars and P. Švestka. A probabilistic learning approach to motion planning. In K.Y. Goldberg, D. Halperin, J.C. Latombe, and R.H. Wilson, editors, *Algorithmic Foundations of Robotics*, pages 19–37. A K Peters, 1995.

[54] 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.

[55] C. Pisula, K. Hoff, ¡. Lin, and D. Manocha. Randomized path planning for a rigid body based on hardware accelarated voronoi sampling" booktitle = wafr, year = 2000.

[56] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, second edition, 1992.

[57] E. J. F. Primrose. On the input-output equation of the general 7R-mechanism. *Mechanism and Machine Theory*, 21(6):509–510, 1986.

[58] J. T. Schwartz and M. Sharir. On the piano movers' problem: II. General techniqies for computing topological properties of algebraic manifolds. *Communications on Pure and Applied Mathematics*, 36:345–398, 1983.

[59] P. Švestka and M. Overmars. Probabilistic path planning. In J-P. Laumond, editor, *Robot Motion Planning and Control*, pages 255–304. Lecture Notes in Control and Information Sciences, Springer, NY, 1998.

[60] 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.

[61] S.C.A. Thomopoulos and R.Y.J. Tam. An iterative solution to the inverse kinematics of robotic manipulators. *Mechanism and Machine Theory*, 26(4):359–373, 1991.

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

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

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