The International Journal of Robotics Research

http://ijr.sagepub.com/

Reachable Distance Space: Efficient Sampling-Based Planning for Spatially Constrained Systems

Xinyu Tang, Shawna Thomas, Phillip Coleman and Nancy M. Amato

The International Journal of Robotics Research 2010 29: 916 originally published online 25 January 2010

DOI: 10.1177/0278364909357643

The online version of this article can be found at:

http://ijr.sagepub.com/content/29/7/916

Published by:

SAGE

http://www.sagepublications.com

On behalf of:

ijrr

Multimedia Archives

Additional services and information for *The International Journal of Robotics Research* can be found at:

Email Alerts: http://ijr.sagepub.com/cgi/alerts

Subscriptions: http://ijr.sagepub.com/subscriptions

Reprints: http://www.sagepub.com/journalsReprints.nav

Permissions: http://www.sagepub.com/journalsPermissions.nav

Citations: http://ijr.sagepub.com/content/29/7/916.refs.html

>> Version of Record - May 21, 2010

OnlineFirst Version of Record - Jan 25, 2010

What is This?

Downloaded from ijr.sagepub.com at TEXAS A&M UNIV on February 24, 2012

Xinyu Tang
Google,
1600 Amphitheatre Parkway,
Mountain View,
CA 94043,
USA
xtang@google.com

Shawna Thomas
Phillip Coleman
Nancy M. Amato
Department of Computer Science
and Engineering,
Texas A&M University,
College Station,
TX 77843,
USA
{sthomas, pcoleman}@cs.tamu.edu
amato@tamu.edu

Reachable Distance Space: Efficient Sampling-Based Planning for Spatially Constrained Systems

Abstract

Motion planning for spatially constrained robots is difficult due to additional constraints placed on the robot, such as closure constraints for closed chains or requirements on end-effector placement for articulated linkages. It is usually computationally too expensive to apply sampling-based planners to these problems since it is difficult to generate valid configurations. We overcome this challenge by redefining the robot's degrees of freedom and constraints into a new set of parameters, called reachable distance space *(RD-space), in which all configurations lie in the set of constraint-satisfying subspaces. This enables us to directly sample the constrained subspaces with complexity linear in the number of the robot's degrees of freedom. In addition to supporting efficient sampling of configurations, we show that the RD-space formulation naturally supports planning and, in particular, we design a local planner suitable for use by sampling-based planners. We demonstrate the effectiveness and efficiency of our approach for several systems including closed chain planning with multiple loops, restricted end-effector sampling, and on-line planning for drawing/sculpting. We can sample single-loop closed chain systems with 1,000 links in time comparable to open chain sampling, and we can generate samples for 1,000-link multi-loop systems of varying topologies in less than a second.*

KEY WORDS—Motion Planning; Sampling-Based Planners; Closed Chain Systems; Spatial Constraints.

The International Journal of Robotics Research
Vol. 29, No. 7, June 2010, pp. 916–934
DOI: 10.1177/0278364909357643
© The Author(s), 2010. Reprints and permissions:
http://www.sagepub.co.uk/journalsPermissions.nav
Figures 1, 8–10, 12–17 appear in color online: http://ijr.sagepub.com

1. Introduction

Spatially constrained systems are systems with constraints such as requiring certain parts of the system to maintain contact or to maintain a particular clearance from each other. They have many applications in robotics and beyond, such as parallel robots (Merlet 2002), grasping for single and multiple robots (Khatib et al. 1996), reconfigurable robots (Kotay et al. 1998; Nguyen et al. 2001), closed molecular chains (Singh et al. 1999), and computer animation (Kallmann et al. 2003). Figure 1 gives some examples of spatially constrained systems.

Motion planning for these systems is particularly challenging due to the additional constraints placed on the system. Since the complexity of deterministic algorithms (Reif 1979; Latombe 1991) is exponential in the number of degrees of freedom (DOFs) of the robot, they are impractical for most systems of interest. While sampling-based planning methods are successful for many high DOF systems, their performance degrades for spatially constrained systems in which the set of

Downloaded from ijr.sagepub.com at TEXAS A&M UNIV on February 24, 2012

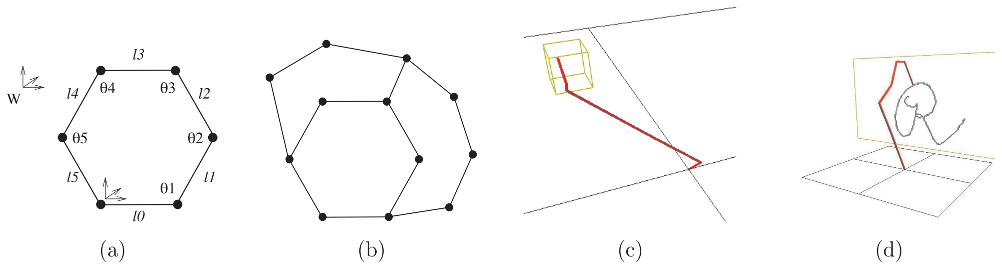

Fig. 1. Examples of spatially constrained systems. Each must satisfy certain closure constraints. (a) Single loop: loop must remain closed. *W* is the world coordinate frame. (b) Multiple loops: all loops must remain closed. (c) End-effector restrictions: the end effector must remain within the specified boundary (displayed in wireframe). (d) Drawing/sculpting: the end effector of the robotic arm must follow a desired drawing trajectory while remaining in contact with the canvas.

valid configurations occupies a small volume of configuration space (the set of all configurations, valid or not). In these situations, the probability of randomly sampling a configuration that also satisfies the spatial constraints approaches zero (e.g. consider closed chain systems (LaValle et al. 1999)).

Previously, we presented a reachable distance representation for handling single-loop closed chains (Tang et al. 2007) and later for handling other types of constraints including multiple-loop closed chains and end-effector position placement (Tang et al. 2008). In this paper, we integrate our prior work and provide a comprehensive presentation of our new representation for spatially constrained systems, called *reachable distance space* (RD-space), RD-space enables more efficient sampling of valid configurations. It is defined by a set of reachable distances for the robot instead of, e.g., by joint angles, as in configuration space. Instead of sampling in the configuration space, we sample in the RD-space. Our formulation handles a wide range of systems including both 2D and 3D chains, single and multiple loops, end-effector placement/trajectory requirements, and prismatic joints.

We describe our RD-space formulation in Section 3 and give a recursive sampling algorithm with complexity linear in the system's DOFs in Section 4. Our method can guarantee that a sampled configuration satisfies the specified spatial constraints or report that one cannot be obtained. We also describe a local planning method that operates in RD-space that is suitable for use in sampling-based planners. Then, we present applications of our approach for three types of spatially constrained motion planning problems, including multiple-loop closed chain planning (Section 5), restricted end-effector sampling (Section 6), and on-line planning for drawing or sculpting (Section 7). We demonstrate that our method is scalable to thousands of DOFs and show that it outperforms other randomized sampling methods such as PRM (Kavraki et al. 1996) and RRT (LaValle and Kuffner 2001) and other specialized methods for closed chain planning.

2. Related Work

In theory, exact motion planning algorithms can handle systems with spatial constraints by explicitly computing the constraint-satisfying subspaces in configuration space, i.e. the set of all robot configurations, valid or not (Reif 1979; Latombe 1991). However, since these algorithms are exponential in the dimension of configuration space, they are generally impractical for systems with more than four or five DOFs. This is amplified for spatially constrained systems where the set of constraint-satisfying configurations typically occupies a small volume of configuration space.

Randomized algorithms such as Probabilistic Roadmap Methods (PRMs) (Kavraki et al. 1996) and Rapidly-exploring Randomized Trees (RRTs) (LaValle and Kuffner 2001) are widely used today for many applications. These methods randomly sample the configuration space, retaining valid configurations, and connect these samples together to form a roadmap (i.e. a graph or tree) that represents the connectivity of the valid configuration space. While successful for many high-DOF systems, their performance degrades for spatially constrained systems as the probability of randomly sampling a configuration satisfying the constraints approaches zero (LaValle et al. 1999).

Much work has been done to optimize these sampling-based planners for closed chain systems. Closed chain systems are a special type of spatially constrained system where the linkage must satisfy the closure constraints at all times during planning. In this case, the valid configurations will be on the constraint surface in C-space. The first sampling-based method for closed chains randomly samples configurations and then applies an iterative random gradient descent method to push the configuration to the constraint surface (LaValle et al. 1999; Yakey et al. 2001). The approach solved planar closed chains with up to eight links and two loops in several hours with PRM (LaValle et al. 1999) and several minutes with RRT (Yakey et al. 2001). Kinematics-based PRM (KBPRM) (Han and Amato 2001) first builds a roadmap ignor-

Downloaded from ijr.sagepub.com at TEXAS A&M UNIV on February 24, 2012

ing all obstacles, then populates the environment with copies of this kinematics roadmap, removes invalid portions, and connects copies of configurations with a rigid-body local planner. This method can solve closed chain problems with seven to nine links in under a minute. KBPRM has been extended to better handle larger linkages (Xie and Amato 2004; Bayazit et al. 2005), reduce running time (Cortés et al. 2002; Cortés and Siméon 2005), and deal with multiple loops (Cortés and Siméon 2003) and singular configurations (Gharbi et al. 2008). PRM-MC combines PRMs and Monte Carlo simulations to guarantee closure constraints (Han 2004). They can generate 100 samples of a 100-link closed chain and of a two-loop system containing 16 links in under a minute. Trinkle and Milgram (2002) proposed a path planning algorithm based on configuration space analysis Milgram and Trinkle (2004) that does not consider self-collision. They extend this method to handle planar closed chains with point-obstacles (Liu and Trinkle 2005) and planar parallel star-shaped manipulators (Liu et al. 2006; Shvalb et al. 2007). A set of geometric parameters for closed chain systems was proposed such that the problem can be reformulated as a system of linear inequalities (Han et al. 2006). They extend this work to handle multiple loops (Han and Rudolph 2009). They show results for a 1,000-link closed chain and a 1,000-loop system containing 3,000 links. However, they do not discuss the algorithm's complexity or provide an experimental performance study. Their work is similar to ours in that both methods reframe the original joint-based problem into another set of parameters where satisfying a set of spatial constraints is easier. Our work is particularly suited to sampling-based planners because it proposes a different set of parameters that enables efficient sampling of configurations and naturally results in a simple local planner to connect configurations, the two primitive operations necessary for sampling-based planning.

In addition to closed chain systems, there has been work on other types of spatially constrained systems. Constrained dynamics have been used to plan motions to ensure constraints such as joint connectivity, the spatial relationship between multiple robots, or obstacle avoidance (Garber and Lin 2003). Results are shown for a six-DOF robot arm. Other work plans articulated motions where the end effector must travel along a given trajectory for fixed-base manipulators (Oriolo et al. 2002) and mobile manipulators (Oriolo and Mongillo 2005). It is extended to probabilistic planning methods for this system by generating samples that satisfy the end-effector trajectory constraint. They present results for up to a six-DOF robot arm on a mobile planar base. This work is extended to allow for robot self-motions, i.e. motions which do not affect end-effector placement or orientation (Yao and Gupta 2006). Two approaches, alternate task-space and configuration-space exploration (ATACE) and randomized gradient descent, plan paths for manipulators with general end-effector constraints (Yao and Gupta 2005); results are presented for up to a nine-DOF robot. Task space coordinates have been introduced to constrain portions of the robot including end-effector placement and closure constraints (Stilman 2007). These coordinates yield a distance metric that is used to quantify the error in the sampled configuration that does not satisfy the constraints. Results are presented for a six-DOF robot arm on a mobile base. Han et al. unify their work on serial chains (Han and Rudolph 2007a) and closed chains (Han et al. 2006) to provide inverse kinematic solutions that satisfy three-, five-, and six-dimensional end-effector placement constraints (Han and Rudolph 2007b). They can sample a single configuration for a 1,000-link arm in 19 ms. There has also been work on planning paths for objects under manipulation constraints (e.g. contact points) of an elastic plate (Kavraki et al. 1998; Lamiraux and Kavraki 1999, 2001) and flexible rope/wire (Moll and Kavraki 2006; Saha and Isto 2006). In related work, Ladd and Kavraki (2004) studied knot tying and untying problems.

3. Reachable Distance Formulation

In this section, we describe a reachable distance data structure for planning the motion of general spatially constrained systems. The reachable distance data structure (or RD-tree) is a hierarchy of reachable distances defined by recursively partitioning the original system into smaller sub-systems. This is a generalized version of the data structure we developed in our previous work for single closed-loop systems (Tang et al. 2007). The main advantage of this representation over the traditional joint angle representation is that it encodes spatial constraint information. For a given set of constraints, this new representation helps us quickly determine whether the robot is able to satisfy those constraints, and if so, generate configurations that satisfy them.

We begin with a simple example illustrating the main ideas of our approach. Figure 2 presents a simple robot with two variable length links, a and b. Suppose that we are given a spatial constraint where the distance between the base and the end effector needs to be c. To satisfy this constraint, the length of each link has to be in an appropriate range to satisfy the triangle inequality: $|a - b| \leq |c| \leq |a + b|$. We call this range the *available reachable range* (ARR), i.e. the set of distances/lengths which allow the spatial constraints involved in the sub-chain to be satisfied. In this case, the ARR of link a, for example, can be calculated from the constraint c and the *reachable range* of the other link b. We can first sample a length for link a and update the ARR of the other link b, and then sample b. Once we find valid values of a and b, then a, b, and c form a triangle and we can calculate the joint angle between link a and link b to find a configuration satisfying the spatial constraint c.

In the following, we show how to extend this sampling strategy to handle more general spatially constrained systems. We note that this scheme only ensures that the spatial constraints are satisfied — collision checking is still required to

Downloaded from ijr.sagepub.com at TEXAS A&M UNIV on February 24, 2012

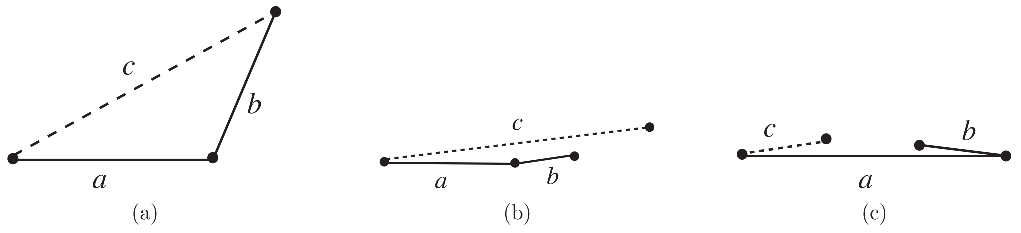

Fig. 2. Three configurations of an articulated system composed of two variable length links a and b where the distance between the base and the end effector needs to satisfy spatial constraint c. (a) Here a and b are of appropriate length to satisfy constraint c. (b) The combined lengths of a and b are too short to satisfy constraint c. (c) The combined lengths of a and b are too long to satisfy constraint c.

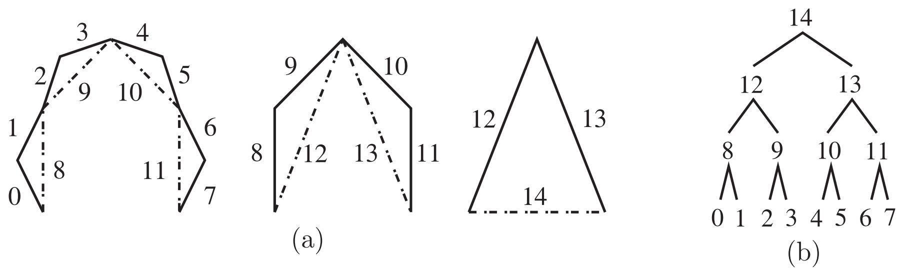

Fig. 3. (a) Articulated linkage with eight links (labeled 0–7). Two consecutive links form a parent vlink (dashed lines). This repeats to form a hierarchy (b) where the parents in one level become the children in the next level. (b) The tree represents the entire reachable distance hierarchy where the nodes correspond to the vlinks.

determine the configuration's validity. To simplify the exposition, here we consider articulated robots with spatial constraints where the distance between the base and the end effector is required to be in a certain range. However, our reachable distance representation is general and may be applied to other robotic systems and other spatial constraints. Our formulation can handle many types of systems including both 2D and 3D chains, single and multiple loops, end effector requirements, and prismatic joints.

3.1. Articulated Linkage as a Hierarchy of Virtual Links

In an articulated linkage, two (or more) consecutive links (called *children*) form a *virtual link* (called the *parent*). A *subchain* is composed of the virtual link, or *vlink*, and its children. For example, in Figure 3(a), the vlink 8 is the parent of the two actual links 0 and 1, while vlink 12 is the parent of vlinks 8 and 9. We iteratively build higher-level vlinks until we obtain a single vlink (14) at the highest level, see Figure 3(b).

3.2. Reachable Range of a Virtual Link

For a particular configuration of a vlink, the *reachable distance* of a vlink (sub-chain) is the distance between the endpoints of the sub-chain it represents. It has different values for different configurations. We call the range of those different values the *reachable range* (RR) of the vlink. For example, the RR of the "root" vlink is the RR of the entire chain, while the RR of an actual link is simply the range of its length.

The RR of a parent can be calculated from the RRs of its children. If we always build a vlink using zero or two children, RRs are computed as follows. Consider a vlink with no children. It has only one link. Let $l_{\min}$ and $l_{\max}$ be the minimum and maximum allowable values of the link's length, respectively. If the link is not prismatic, then $l_{\min} = l_{\max}$. The RR is then $[l_{\min}, l_{\max}]$.

Now consider the vlink l with two children, a and b, in Figure 4(a). They form a triangle. Let $[a_{\min}, a_{\max}]$ be the RR of the first child and $[b_{\min}, b_{\max}]$ be the RR of the second child. If a or b is an actual link, then its RR is defined by the problem, otherwise it is a function of the RRs of its children. RRs are defined from the bottom up, recursively. The RR of the parent is $[l_{\min}, l_{\max}]$ where

$$l_{\min} = \begin{cases} \max(0, b_{\min} - a_{\max}), & a_{\min} < b_{\min}, \\ 0, & a_{\min} = b_{\min}, \\ \max(0, a_{\min} - b_{\max}), & a_{\min} > b_{\min}, \end{cases} \quad (1)$$

and

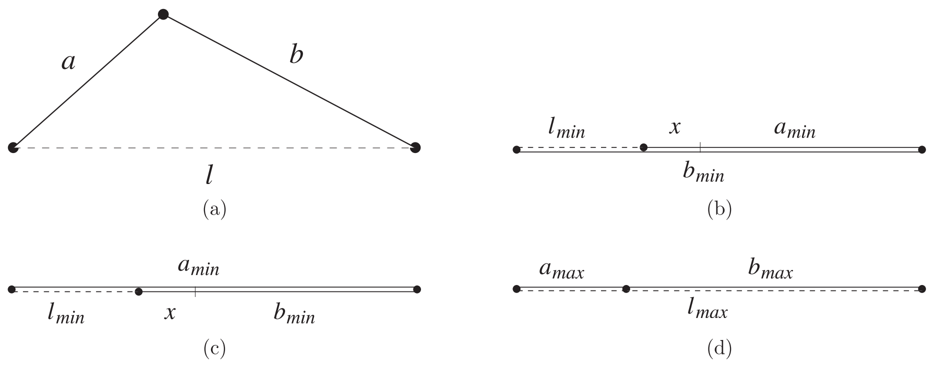

Fig. 4. (a) A sub-chain with two children a and b and its vlink l. (b) A configuration where l is minimum when $a_{\min} < b_{\min}$. We have $x = b_{\min} - a_{\min}$ if $a_{\max} > b_{\min}$ and $x = a_{\max} - a_{\min}$ otherwise. (c) A configuration where l is minimum when $a_{\min} > b_{\min}$. We have $x = a_{\min} - b_{\min}$ if $b_{\max} > a_{\min}$ and $x = b_{\max} - b_{\min}$ otherwise. (d) A configuration where l is maximum.

$$l_{\max} = a_{\max} + b_{\max}. \tag{2}$$

Note that these equations are not limited to computing RRs of parent links. Given the RRs of any two links in the same triangular sub-chain, Equations (1) and (2) calculate the RR of the remaining link to satisfy the triangle inequality.

3.3. Available Reachable Range of a Virtual Link

The ARR of a vlink is the set of distances/lengths which allow it to satisfy the spatial constraints when only considering the other links in the same sub-chain, e.g. satisfy the triangle inequality. That is, the ARR is a subset of the RR in which the spatial constraints may be satisfied. It is a function of the ARR of the other links in the same sub-chain loop. Changes in the ARR of one link cause changes in the ARRs of the other links in the sub-chain. Note that this considers only the spatial constraints and does not consider collision detection or other requirements on validity which must be checked explicitly after sampling.

Before sampling begins (i.e. no spatial constraint is imposed on the robot), the ARR of each link is equal to its RR. For example, in the beginning, for each sub-chain with two children, the RRs of all three links can satisfy the triangle inequality and thus the ARR is the same as the RR. However, as we sample and enforce spatial constraints by fixing the length of a link, portions of the RRs of the other links in the same sub-chain may no longer be available. When this happens, we need to update the ARRs in the other links in the sub-chain. We can do this using Equations (1) and (2). Note that Equations (1) and (2) are used in a more general way here: a, b, and l can be any link in the same triangular sub-chain, l does not have to be the parent link. So, given a specified value of one link we can find the ARR of the other two in the sub-chain and sample available lengths for them. The sampling cost is discussed in Section 4.1.

This process results in a set of lengths, one for each vlink, for a configuration that satisfies the spatial constraints. We can then compute the joint angles between vlinks using only basic trigonometry functions instead of more expensive inverse kinematics solvers. We illustrate reachable distance sampling through example applications in more detail in the following sections.

Our approach is related to the two-link inverse kinematics problem (i.e. to find appropriate joint angles for a two-link manipulator given an end-effector placement) which has been well studied with closed-form, geometric solutions (Lee and Ziegler 1984). The reachable distance formulation can be thought of as a recursive hierarchy of two-link inverse kinematics problems. Thus, given a set of sampled lengths, computing the joint angles between vlinks is simply solving the corresponding two-link inverse kinematics problem.

4. Primitives for Sampling-based Planning

Here we describe two primitive operations that suffice to implement most sampling-based motion planners such as PRMs and RRTs: sampling (i.e. generating valid configurations) and local planning (i.e. finding a valid path between two samples). We show that these operations are fast and efficient, thereby allowing sampling-based motion planners to be directly applied to large spatially constrained problems.

4.1. Sampling with the RD-tree

Sampling in RD-space involves the following steps:

1. Recursively sample vlink lengths from their ARR.
2. Sample the orientation of each sub-chain.

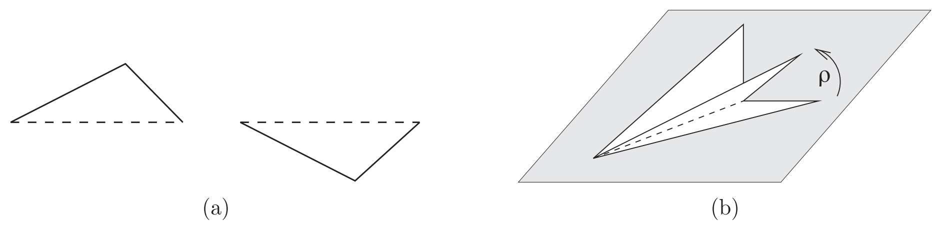

Fig. 5. (a) In two dimensions, the same vlink represents two configurations: a concave triangle and a convex triangle. (b) In three dimensions, the same vlink represents many configurations with different dihedral angles ρ. (Here ρ is the angle between the sub-chain's plane and its parent's plane.)

3. Compute appropriate joint angles from the vlink lengths and orientations.

This sampling scheme can always generate a constraint-satisfying sample or report that one does not exist. We recall that this sampling does not necessarily determine validity, i.e. the configuration will still need to be checked for (self-) collision.

This sampling scheme samples the RD-space uniformly at random. While every point in RD-space satisfies the spatial constraints, a set of samples in RD-space may not have the same distribution in the subset of joint space that also satisfies the constraints. In other words, a sampling method with a uniform distribution in RD-space does not guarantee a uniform distribution in the spatial constraint-satisfying subset of joint space.

In the following, we describe each step in more detail.

4.1.1. Recursively Sample Link Lengths

Recall that we define the ARR as the subset of the RR that satisfies the spatial constraints with respect to the rest of the sub-chain. Once we fix the length of an ARR in a sub-chain, the other ARRs may be restricted. Thus, we can generate a configuration by sampling reachable distances and updating ARRs starting at the root of the RD-tree and recursing until all sub-chain reachable distances are sampled. Algorithm 1 describes this recursive sampling strategy. Note that by sampling in this way, an ARR may never become empty; in its most constrained case, its minimum and maximum may become equal. Thus, it is always possible to sample a constraint-satisfying configuration given that one exists.

The sampling algorithm is called once for each vlink. Recall that each vlink is represented by a single internal node in the hierarchy of sub-chains. As the hierarchy of sub-chains is constructed as a binary tree, there are $O(n)$ internal nodes, where n is the number of actual links in the chain. Thus, the sampling algorithm requires $O(n)$ time to generate a configuration, i.e. time linear in the size of the system.

Algorithm 1 Sample Lengths

```
Input. A sub-chain $c$. Let $c.arr$ be $c$'s ARR, $c.left$ and
    $c.right$ be $c$'s children, and $c.len$ be the length of $c$'s
    vlink. Let $p$ be $c$'s parent and $s$ be $c$'s sibling.
1: Update $c.arr$ from $p.arr$ and $s.arr$ (from Equations (1)
   and (2) where link $l$ is $c$, link $a$ is $p$, and link $b$ is $s$ in the
   equations).
2: Randomly sample $c.len$ from $c.arr$.
3: Set $c.arr$ to $[c.len, c.len]$.
4: if $c$ has children then
5:     Sample($c.left$).
6:     Sample($c.right$).
7: end if
```

4.1.2. Sample Link Orientations

Each sub-chain forms a triangle, and there are multiple configurations with the same vlink length: there are two in two dimensions (i.e. concave and convex, see Figure 5(a)), and there are many in three dimensions depending on the dihedral angle between its triangle and its parent's triangle (see Figure 5(b)). Thus, we also sample the orientation of the vlink. This orientation sampling is done after all of the vlink lengths are sampled.

4.1.3. Calculate Joint Angles

Consider the joint angle θ between links a and b. Links a and b are connected to a vlink c to form a triangle. Let l_a, l_b, and l_c be the lengths of links a, b, and c, respectively. The joint angle can be computed using the law of cosines:

$$\theta = \arccos\left(\frac{l_a^2 + l_b^2 - l_c^2}{2 l_a l_b}\right). \tag{3}$$

4.2. Local Planning in RD-space

Based on the presentation of reachable distances, we propose a local planner for spatially constrained systems. Given

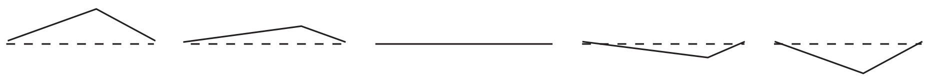

Fig. 6. In two dimensions, we need to flip the links to transform a convex configuration to a concave configuration.

two configurations, q_s and q_g, a local planner attempts to find a sequence $\{q_s, q_1, q_2, \ldots, q_g\}$ of valid configurations to transform q_s into q_g at some user-defined resolution. Here we describe a simple local planner that uses a straight-line interpolation in the RD-space to determine the sequence of configurations. While this is certainly not the only local planning scheme possible, we find it performs well in practice.

To generate the sequence of intermediate configurations, this local planner interpolates the lengths of each vlink between q_s and q_g. We then must determine the vlink's orientation sequence (i.e. concave or convex in two dimensions and the dihedral angle in three dimensions) to fully describe the sequence of configurations. In two dimensions, if the orientation is the same in q_s and q_g, we keep it constant in the sequence. If it is not the same, we must "flip" the links as described below. In three dimensions, we simply interpolate between the dihedral angle in q_s and the dihedral angle in q_g. In addition to checking collisions as with other local planners, we determine the validity of the sequence by checking that each vlink's length is in its ARR.

4.2.1. *Concave/Convex Flipping for Two-dimensional Chains*

Consider the first and last configurations of sub-chains and vlinks in Figure 6. When the orientation is different, as in this example, we need to find a transformation from one to the other. To flip the vlink, we need to expand (or open) the parent's vlink enough so its children can change orientation while remaining in their ARR. At some point, the parent's ARR must be large enough to accommodate the summation of its children's reachable distances (i.e. allow the children to be "flat"). Such a constraint on reachable distance can be easily handled by our method by first recalculating the ARR for this minimum "flat" constraint. If available, we sample a configuration where the vlinks are "flat" and try connecting it to both the start and goal configurations as described above.

5. Application: Closed Chain Planning

Closed chain systems are involved in many applications such as parallel robots (Merlet 2002), closed molecular chains (Singh et al. 1999), animation (Kallmann et al. 2003), reconfigurable robots (Kotay et al. 1998; Nguyen et al. 2001), and grasping for single and multiple robots (Khatib et al. 1996). However, motion planning for closed chain systems is particularly challenging due to additional closure constraints placed on the system. Using only the traditional joint angle representation, it is extremely difficult to randomly sample a set of joint angles that satisfy the closure constraints since the probability that a random set of joint angles lies on the constraint surface is zero (LaValle et al. 1999).

We first describe how our representation of reachable distances can be used to ensure the closure constraint and how to handle simultaneous constraints such as with multiple loops. Here we show results for single-loop closed chains and for multiple-loop systems. All experiments were performed on a 3 GHz desktop computer and were compiled using gcc4 under Linux. Our current implementation supports planar joints and spherical joints.

5.1. Enforcing the Closure Constraint

A closure constraint can be considered as a special type of spatial constraint where the end effector (the last link) is always connected to the base (the first link) of the robot. Thus, to ensure the closure constraint in our formulation, we simply require that the distance between the first link and the last link is zero by making the root vlink length zero. For example, as shown in Figure 3, if we set the length of link 14 to be zero, then link 0 and link 7 will be connected and this system will remain closed.

5.2. Handling Multiple Loops

For a closed chain system that has more than one loop, we construct a RD-tree for each loop. Then the whole system can be represented by a set of RD-trees with some sub-trees in common. The common sub-chain between two loops now becomes a shared node in both trees. To make both loops closed, the common edge should satisfy the closure constraints in both trees, i.e. the ARR of a common node should be the intersection of the ARRs of the same vlink on both trees. The loops must be ordered such that each loop only has junctions with its predecessors. For instance, ear decompositions from graph theory provide such an ordering (Whitney 1932, 1933).

Figure 7(a) and (b) show an example of a closed chain system with two loops and the corresponding set of RD-trees, respectively. Two trees rooted at nodes 17 and 18, respectively, correspond to the two loops in this system. Both trees share the same node 12. When we update the ARR of 12, we should

Downloaded from ijr.sagepub.com at TEXAS A&M UNIV on February 24, 2012

Table 1. Time to generate 1,000 open/closed configurations in the RD-space without collision detection. We show both the time to purely sample a set of attainable reachable distances from the RD-space and the time including the conversion into the traditional joint angle representation.

	Open chain time (s)		Closed chain time (s)	
Number of links	Sample	Sample + convert	Sample	Sample + Convert
10	0.006	0.017	0.006	0.018
20	0.012	0.039	0.012	0.042
50	0.031	0.123	0.032	0.130
100	0.061	0.268	0.061	0.267
200	0.123	0.560	0.124	0.554
500	0.309	1.863	0.304	1.512
1,000	0.595	3.219	0.599	3.267
5,000	3.230	18.254	3.170	18.504
10,000	7.000	39.449	7.296	39.823
100,000	73.202	429.958	73.008	441.217

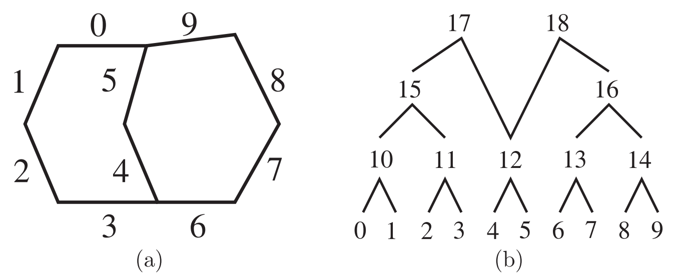

Fig. 7. (a) Multiple loop system and (b) corresponding set of RD-trees.

consider the ARR of both link 15 and link 16. Given such a set of RD-trees, we can perform the sampling and planning on each loop sequentially. As we did for a single loop, we set the length of each root link to be zero and then sample the lengths of other links in lower levels.

Even with multiple loops, the sampling algorithm is only called once for each vlink. However, after sampling each loop, a geometry calculation must be made to compute the constraint for the next loop. This is done once for each loop. Thus, the running time for the algorithm is $O(n+l)$ where n is the total number of actual links in the system and l is the number of loops.

5.3. Results

5.3.1. Single-loop Closed Chains

We first compare the performance of our sampling algorithm both with and without collision detection for open chains and single-loop closed chains of varying size. In the first set of experiments, we ignore collision detection. We measure the time to generate 1,000 configurations for chains of size 10, 20, 50, 100, 200, 500, 1,000, 5,000, 10,000, and 100,000 links. In each environment, the articulated system is composed of randomly sized links ranging from 0.1 to 1.0. Links are connected consecutively to form a long open chain or a single closed loop where the last link is connected to the first link. We compare the performance of generating both open and closed configurations using reachable distances.

As shown in Table 1 and in Figure 8, the running time has two parts: (1) the time to sample the reachable distances and link orientations and (2) the time to convert the reachable distance representation into traditional joint angles. Note that it takes much more time for conversion than for actual sampling. However, the performance of both scales well even for large numbers of links. These results show that the time required to sample closed configurations is comparable to the time to sample open chain configurations, i.e. there is negligible cost to enforce closure constraints. This demonstrates the advantage of using reachable distances to represent configurations over the traditional joint angle representation.

In the second set of experiments, we measure the performance when self-collision is considered. Again, we measure the time to sample and convert the reachable distance representation into joint angles of 1,000 self-collision-free configurations. We studied chains of size 20, 50, 100, 200 and 500 with random link lengths between 0.1 and 1.0. For each system, we measured the running time to generate open chain and closed chain configurations in RD-space. We also measure the performance using two other closed chain planners, KBPRM (Han and Amato 2001) and IRC (Bayazit et al. 2005) which were already demonstrated to perform better than ex-

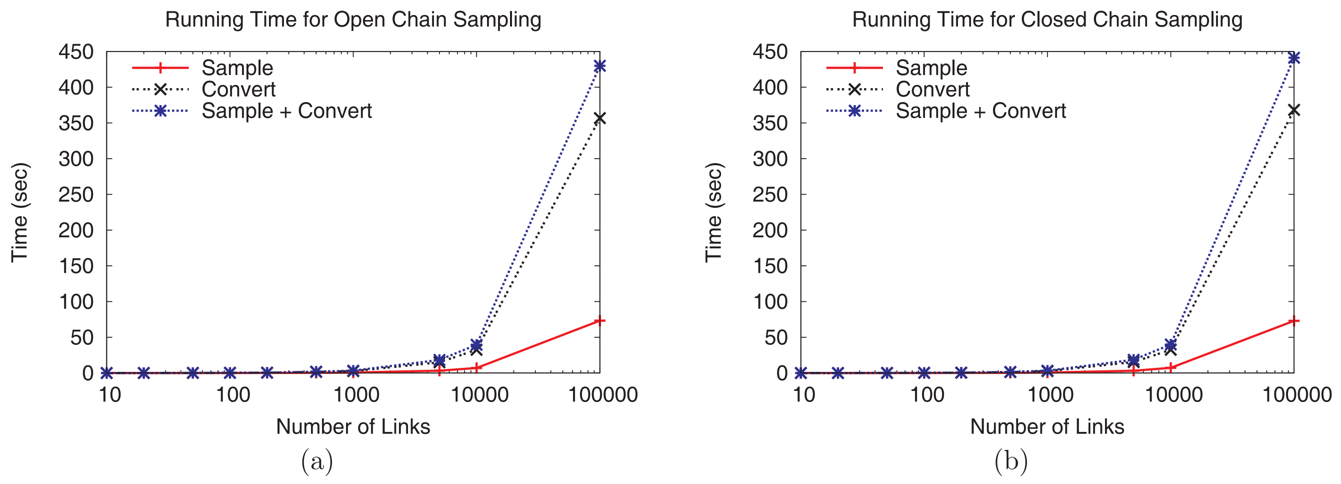

Fig. 8. Running time to generate 1,000 open (a) or closed (b) configurations in the RD-space without considering collision. Each plot shows the total running time, the pure sampling time in RD-space, and the time to convert configurations into joint-angle space. Note that the running times for closed and open configurations are similar.

isting closed chain methods such as randomized gradient descent (LaValle et al. 1999; Yakey et al. 2001). We stop an experiment after 12 hours. Note that we are unable to compare our results with those of Han and Rudolph (2007a); Han et al. (2006) because we do not have access to their implementation. They report that their Matlab implementation can generate a single closed conformation for a 1,000-link chain in 19 ms on a "desktop PC" (processor speed not reported).

Table 2 and Figure 9 show the performance results. KBPRM was only able to generate nodes for 20 links within the 12-hour time limit; it took 4.36 s without collision detection and 902.53 s with collision detection. Note that when including collision detection, the cost to use reachable distances to sample an open chain is comparable to the cost to sample a closed chain. It is slightly more expensive to sample a closed configuration because there are more self-collisions. This again shows that by directly sampling in the RD-space, closed configurations can be sampled efficiently. Our method makes it practical to solve motion planning problems for large articulated systems with distance-related constraints. Also note that our method out-performs the KBPRM method and IRC method.

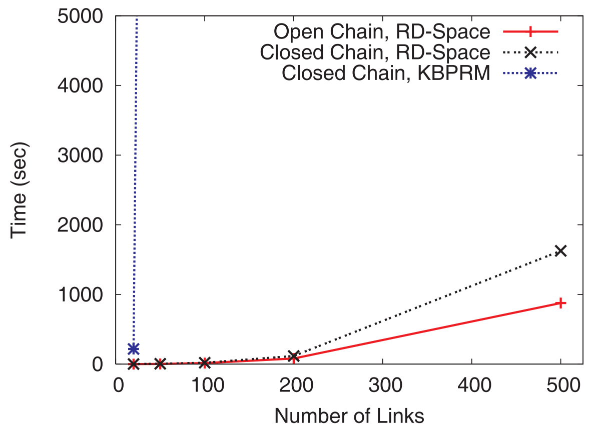

Fig. 9. Running time of open configuration sampling using reachable distances and closed configuration sampling using reachable distances all with collision detection as a function of the number of links. The performance for both methods is comparable since they are each dominated by collision detection.

5.3.2. Multiple-loop Closed Chain Systems

Figure 10 shows an example of a three-loop system with 14 links. Given only the (a) start configuration and the (e) goal configuration, our method took only 0.01 seconds to find this path containing 40 intermediate configurations.

To demonstrate the efficiency and scalability of this approach for multiple loops, we study the time required to generate 1,000 samples. Here we look at three different multiple-loop topologies, see Figure 11. The multiple-loop system can be partitioned into ears (indicated by the arrows). Topology 1 arranges the ears such that the endpoints of ear e_i connect to ear e_{i-1}. Topology 2 arranges the ears such that the endpoints of ear e_i connect to ears e_{i-1} and e_{i-2}. Topology 3 arranges the ears in a "honeycomb" pattern. For topologies 1 and 2, we vary the number of loops in the system. For topology 3, we vary the number of rings in the honeycomb pattern from one to four.

Tables 3 and 4 summarize the results. All results are averaged over 10 runs. Even with 1,024 links and 256 loops to close in topologies 1 and 2, our method takes less than 90 sec-

Downloaded from ijr.sagepub.com at TEXAS A&M UNIV on February 24, 2012

Table 2. Time, in seconds, to generate 1,000 open configurations in the RD-space. We show the time it takes to sample with and without collision detection (CD).

		Number of links				
Method	CD	20	50	100	200	500
RD-space	N	0.04	0.11	0.25	0.53	1.50
Open	Y	0.56	2.96	14.30	80.54	877.34
RD-space	N	0.04	0.11	0.25	0.52	1.43
Closed	Y	0.69	4.58	19.25	116.53	1626.17
IRC	N	61.32	1416.96	3129.16	20737.20	n/a
Closed	Y	216.99	40089.20	n/a	n/a	n/a

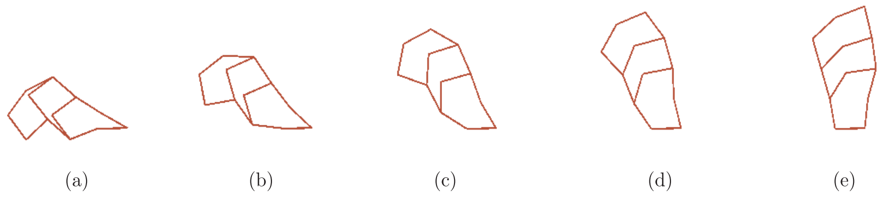

Fig. 10. A path for a three-loop closed chain system from (a) to (e).

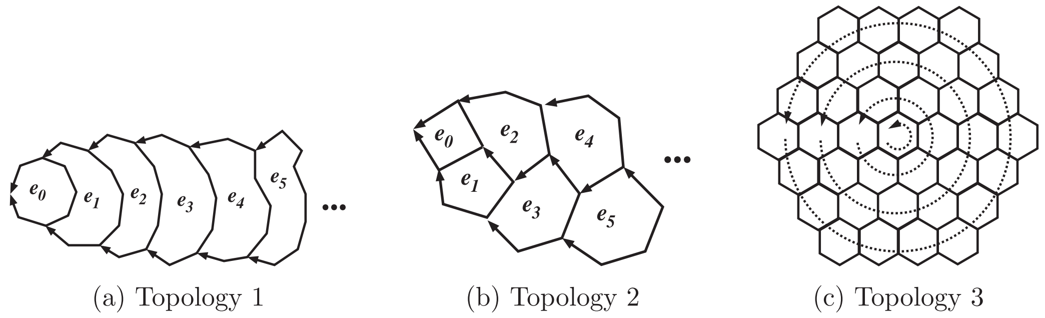

Fig. 11. Multiple-loop topologies studied. (a) Topology 1: example with eight-link ears (indicated with arrows). The ears are arranged such that the endpoints of ear e_i connect to ear e_{i-1}. (b) Topology 2: example with four-link ears (indicated with arrows). The ears are arranged such that the endpoints of ear e_i connect to ears e_{i-1} and e_{i-2}. (c) Topology 3: "honeycomb" example with four rings (indicated with dashed arrows).

onds to generate 100 samples. Figure 12 displays the performance results for topologies 1 and 2 ignoring collisions. Running time grows linearly with the number of loops when the total number of links in the system are fixed.

Table 4 shows that our method performance is only somewhat affected by system topology when collision is ignored. However, different topologies require different numbers of attempts to generate collision-free configurations because they place the links in different proximities to each other. For example, topology 2 requires many more attempts with only 30 links because the links in the inner loops must be very close together to close, see Figure 13(a). We were unable to directly compare to Han and Rudolph (2009) because we do not have access to their Matlab implementation. They report that they can generate a constraint-satisfying configuration for a 1000 loop chain with 3 links per loop (running time and topology not reported).

Downloaded from ijr.sagepub.com at TEXAS A&M UNIV on February 24, 2012

Table 3. Time to generate 100 samples of multiple loop systems with links of varying length, averaged over 10 runs for topologies 1 and 2 (T1 and T2) with 1,024 links. Sample standard deviations are shown in italics.

Number of loops	Loop size	Time (s)		Standard deviation	
		T1	T2	T1	T2
1	1,024	5.085		*0.017*	
2	512	9.403		*0.019*	
4	256	12.422	12.332	*0.032*	*0.175*
8	128	16.059	15.640	*0.042*	*0.176*
16	64	17.770	16.598	*0.046*	*0.091*
32	32	24.910	22.038	*0.075*	*0.130*
64	16	38.526	33.053	*0.178*	*0.186*
128	8	57.139	49.272	*3.945*	*0.200*
256	4	88.542	80.222	*0.325*	*1.316*

Table 4. Time to generate 100 samples of multiple-loop systems with links of varying length, averaged over 10 runs for all three topologies, both with and without collision detection (CD). Sample standard deviations are shown in italics.

Number of links	Number of loops	Time (s)					
		Without CD			With CD		
		T1	T2	T3	T1	T2	T3
6	1		0.010			0.017	
Standard deviation			*0.002*			*0.003*	
30	7	0.324	0.321	0.320	1.595	417.117	1.611
Standard deviation		*0.003*	*0.003*	*0.002*	*0.140*	*35.056*	*0.100*
73	19	1.137	1.140	2.707	12.670	68.097	181.678
Standard deviation		*0.007*	*0.006*	*0.321*	*0.696*	*5.189*	*20.153*
133	37	2.623	2.415	10.167	144.879	1,182.302	55,067.010
Standard deviation		*0.094*	*0.010*	*1.063*	*13.333*	*145.220*	*3,760.144*

6. Application: Restricted End-effector Sampling

Here we discuss how to apply this reachable distance formulation to efficiently sample configurations of an articulated linkage when its end effector is required to remain within a specified boundary, such as a work area or safe zone. We also demonstrate how to use this reachable distance formulation to constrain the orientation of the end effector, in addition to constraining its position.

Consider the robot system in Figure 1(c). The fixed base manipulator end effector is restricted to remain inside the box B. Randomly sampling such a configuration in joint space has near zero probability. Recall that the RR of this robot is $[l_{\min}, l]$ where l is the sum of its link lengths, and $l_{\min}$ is the minimum ARR of the robot. Let the *range* of the robot be the distance from the base to the end effector. Observe that all configurations with end effectors inside the boundary have ranges $[d_{\min}, d_{\max}]$ where $d_{\min}$ ($d_{\max}$) is the minimum (maximum) distance between the robot's base and B. The range $[d_{\min}, d_{\max}]$ is much smaller than the original range $[l_{\min}, l]$. We can take advantage of this information by restricting the ARR of the end effector from $[l_{\min}, l]$ to $[d_{\min}, d_{\max}]$ before we sample the rest of the RD-tree.

6.1. Enforcing End-effector Placement

We can easily restrict the end-effector distance by setting the ARR of the vlink connecting the base and the end effector (i.e. the root of the RD-tree) to $[d_{\min}, d_{\max}]$ and calling the above sampling scheme. However, this alone does not guarantee that the end effector will be in B. A simple way to guarantee this

Downloaded from ijr.sagepub.com at TEXAS A&M UNIV on February 24, 2012

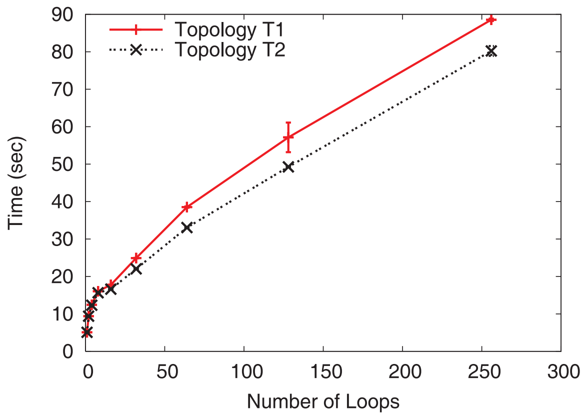

Fig. 12. The time to generate 100 samples of multiple-loop systems with 1,024 links and a varying number of loops, ignoring collisions. Results are averaged over 10 runs. Error bars indicate the sample standard deviation.

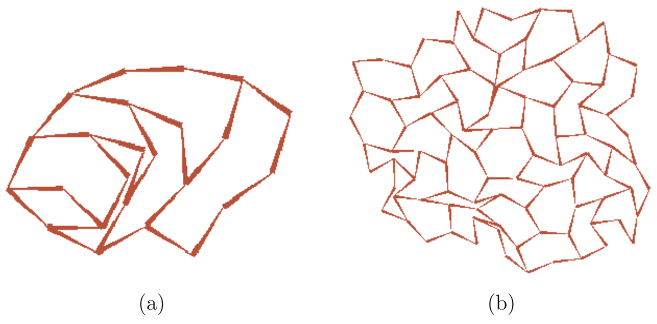

Fig. 13. Example configuration for (a) topology 2 with 30 links and (b) topology 3 with 133 links.

is to first randomly sample a point b in B. Then we calculate the distance between the robot's base and b. We set the ARR of the vlink connecting the base and the end effector to $[d_b, d_b]$ and sample as before. Let e be the resulting end-effector placement. We then compute a rotation θ along a vector υ to rotate e to b and apply this rotation to the robot base. Let υ_e be the normalized vector from the robot base to e and υ_b be the normalized vector from the robot base to b. The rotation axis υ is then $|\upsilon_b \times \upsilon_e|$, and the rotation angle θ is calculated from $\cos(\theta) = \upsilon_e \bullet \upsilon_b$. In this way we can easily sample configurations that keep the end effector inside B.

6.2. Enforcing End-effector Orientation

Using the method described in Section 6.1, we can guarantee that the end effector will be in B, but cannot guarantee its orientation. We can restrict its orientation as follows. First we randomly sample a point b in B as before. Using b and the restrictions on the end effector's orientation, we calculate the position of the end effector's other endpoint, c. We then set the ARR of the vlink connecting the base and the second to last link to $[d, d]$ where d is the distance between the base and c, and solve for the RD-tree that does not include the last link. After we rotate the second to last link to c as described in Section 6.1, we use the orientation of the links to calculate the angle between the last two links. Using this method, we are able to sample configurations that maintain a given placement and orientation for the end effector.

6.3. Results

Here we compare the performance of our sampling scheme in RD-space to uniform random and RRT-style sampling (LaValle and Kuffner 2001) in joint space. (For RRT-style sampling, we do not count the time to check edge validity.) All samplers use the same validity checker: first checking the end-effector placement and then a collision check. The robots have 3 to 100 links of varying length, while the sum of each robot's link lengths is the same. All results were performed on a 3 GHz desktop computer, and all implementations were compiled using gcc4 under Linux.

Table 5 shows how each sampler performs in practice for restricting the end-effector placement. Each sampler was asked to generate 1,000 valid configurations within 1 hour. A sample environment is shown in Figure 1(c). Results are averaged over 10 runs. Neither uniform random sampling or RRT were able to generate a single sample in 1 hour for 50 and 100 links. Reachable distance sampling was not only able to generate samples for 100 links, it could do so faster than RRT for *any* robot and faster than uniform random sampling for any robot other than three links. Clearly, reachable distance sampling out-performs the other randomized sampling strategies for restricted end effector sampling.

Note that RRT performs significantly slower for the three-link robot than for the other larger robots. The success of RRT depends on the placement of the starting configuration. For the three-link robot, minor changes in joint angles of the starting configuration pull the end effector outside the restricted area. Thus, RRT spends more time on the initial tree samples for the three-link robot.

Table 6 shows how each sampler performs in practice for restricting the end-effector orientation as described in Section 6.2. As previously, each sampler was asked to generate 1,000 valid configurations within 1 hour. Results are averaged over 10 runs. Restricting the end-effector orientation only had a marginal effect on the time compared with those seen in Table 5. We see an improvement in all three methods, which is expected. By restricting the orientation of the last link, we are decreasing the number of DOFs that must be considered by each method. This reduced complexity would have a positive effect on all three methods. Even with the slight improvement,

Downloaded from ijr.sagepub.com at TEXAS A&M UNIV on February 24, 2012

Table 5. The time and number of attempts required to generate 1,000 samples when the end-effector placement is restricted, averaged over 10 runs. The results from the uniform random sampling and RRT methods are shown for comparison. Sample standard deviations (SD) are shown in italics.

		Time (s)			
Method	Number of links	Average	SD	Samples generated	Sample attempts
Reachable Distance	3	0.02	*0.00*	1,000.0	1,000.5
	10	0.11	*0.00*	1,000.0	1,817.8
	20	0.50	*0.01*	1,000.0	4,311.9
	50	7.13	*0.29*	1,000.0	24,486.2
	100	29.29	*0.77*	1,000.0	51,835.3
Uniform Random	3	13.89	*0.48*	1,000.0	1,326,106.9
	10	142.04	*2.76*	1,000.0	5,418,904.7
	20	3,572.60	*15.95*	82.1	9,568,969.6
	50	n/a	n/a	n/a	n/a
	100	n/a	n/a	n/a	n/a
RRT	3	519.11	*710.33*	1,000.0	392,073.6
	10	49.67	*2.18*	1,000.0	106,202.3
	20	66.11	*4.42*	1,000.0	121,840.4
	50	n/a	n/a	n/a	n/a
	100	n/a	n/a	n/a	n/a

Table 6. The time and number of attempts required to generate 1,000 samples when the end-effector orientation is restricted, averaged over 10 runs. The results from the uniform random sampling and RRT methods are shown for comparison. Sample standard deviations (SD) are shown in italics.

		Time (s)			
Method	Number of links	Average	SD	Samples generated	Sample attempts
Reachable Distance	3	0.02	*0.00*	1,000.0	1,012.4
	10	0.09	*0.01*	1,000.0	1,753.8
	20	0.48	*0.03*	1,000.0	4,211.5
	50	6.88	*0.17*	1,000.0	24,048.7
	100	28.15	*0.69*	1,000.0	50,513.3
Uniform Random	3	11.54	*0.79*	1,000.0	1,027,366.3
	10	129.20	*2.23*	1,000.0	4,717,493.7
	20	3,286.61	75.3	21.19	9,084,321.1
	50	n/a	n/a	n/a	n/a
	100	n/a	n/a	n/a	n/a
RRT	3	423.41	*473.41*	1,000.0	337,192.8
	10	45.81	*1.93*	1,000.0	100,367.2
	20	63.04	*6.46*	1,000.0	108,589.3
	50	n/a	n/a	n/a	n/a
	100	n/a	n/a	n/a	n/a

Downloaded from ijr.sagepub.com at TEXAS A&M UNIV on February 24, 2012

our reachable distance approach still out performs both RRT and uniform random sampling, especially for robots with a significantly large number of links.

7. Application: Drawing/Sculpting

An interesting application is robotic drawing and sculpting. Figure 1(d) displays an articulated linkage drawing the letter "R" on a canvas. There are many applications in robotics where the manipulator needs to follow a trajectory, e.g., painting. This problem is more constrained than the previous application of restricted end-effector sampling. Here it is not sufficient for the planner to keep the robot's end effector in a restricted space (e.g. the canvas). It must also follow a specific trajectory composed by a sequence of points.

7.1. Enforcing the End-effector Trajectory

Again we can take advantage of the RD-tree. We propose a recursive algorithm that consists of three phases. The entire algorithm is given in Algorithm 2.

- **Phase I.** Let $d_{\min}$ be the minimum distance between the base and any point in the drawing trajectory and let $d_{\max}$ be the maximum distance between the base and any point in the drawing trajectory. With a local planner, we find a valid path between $q_{\min}$ and $q_{\max}$ where $q_{\min}$ is a configuration with end-effector distance $d_{\min}$ and $q_{\max}$ is a configuration with end-effector distance $d_{\max}$. This path is *self*-collision-free. See Algorithm 2, lines 1–3.
- **Phase II.** We then use the pre-computed path to follow the drawing trajectory by selecting the configuration in the path with the appropriate end-effector length and rotating the configuration to align with the drawing target for each point along the drawing trajectory. (Rotating the configuration is performed in the same way as described in Section 6.1.) See Algorithm 2, line 4, and Algorithm 3.
- **Phase III.** If any segments of the rotated path are in collision with objects in the environment, we recursively call the algorithm on the colliding portions. We then use a local planner to connect the resulting segments together. See Algorithm 2, lines 5–21, and Algorithm 4.

Note that if the local planner returns a path that contains configurations with end effectors outside the range $[d_{\min}, d_{\max}]$, we simply clip these portions out of the path. We are then potentially left with multiple path segments. In these results, we simply select the shortest of these path segments for the second phase. However, one could also join these paths together to form a roadmap for additional planning.

Note that if the trajectory lies on a two-dimensional planar canvas, only the first two phases are required and no recursive calls are made.

Algorithm 2 Find_Path

```
Input. A trajectory $T$ and a roadmap $R$.
Output. A collision-free path $P$ of the robot in which the end
    effector follows the trajectory $T$.
 1: Let $d_{\min}$ ($d_{\max}$) be the minimum (maximum) distance from
    the robot base to any point in $T$.
 2: Sample configuration $q_{\min}$ ($q_{\max}$) with end-effector distance $d_{\min}$ ($d_{\max}$).
 3: Find a path $P_{lp}$ from $q_{\min}$ to $q_{\max}$ using the reachable distance local planner. Prune $P_{lp}$ such that the end effector
    remains in $[d_{\min}, d_{\max}]$.
 4: Let $P_{rot}$ = Rotate_Path($P_{lp}$, $T$).
 5: Let $P = \emptyset$.
 6: Partition $P_{rot}$ into collision-free and colliding segments $S$.
 7: for each $s \in S$ do
 8:   if $s$ is colliding then
 9:     Let $T'$ be the subset of $T$ corresponding to $s$.
10:     Let $s$ = Find_Path($T'$, $R$)
11:   end if
12:   Add $s$ to $R$.
13:   if $P \neq \emptyset$ then
14:     Let $P_{conn}$ = Connect_Path($R, q_a, q_b$) where $q_a$ is
        the last configuration in $P$ and $q_b$ is the first
        configuration in $s$.
15:     if $P_{conn} == \emptyset$ then
16:       return $\emptyset$.
17:     end if
18:     Add $P_{conn}$ to $P$.
19:   end if
20:   Add $s$ to $P$.
21: end for
22: return $P$.
```

Algorithm 3 Rotate_Path

```
Input. A path $P$ and a trajectory $T$.
Output. A path $P_{rot}$ rotated onto the trajectory $T$.
 1: Let $P_{rot} = \emptyset$.
 2: for each $t \in T$ do
 3:   Let $d_t$ be the distance from the robot base to $t$.
 4:   Find the configuration $p \in P$ with end-effector distance
      $d_t$.
 5:   Find a rotation $R$ that rotates the end effector in $p$ to $t$.
 6:   Apply $R$ to $p$ and add it to $P_{rot}$.
 7: end for
 8: return $P_{rot}$.
```

7.2. Results

We applied our drawing algorithm to robots with 3, 5, 10, 20, 50, and 100 links. Figure 14 shows the planning results of an articulated robotic arm drawing the character "R" on a can-

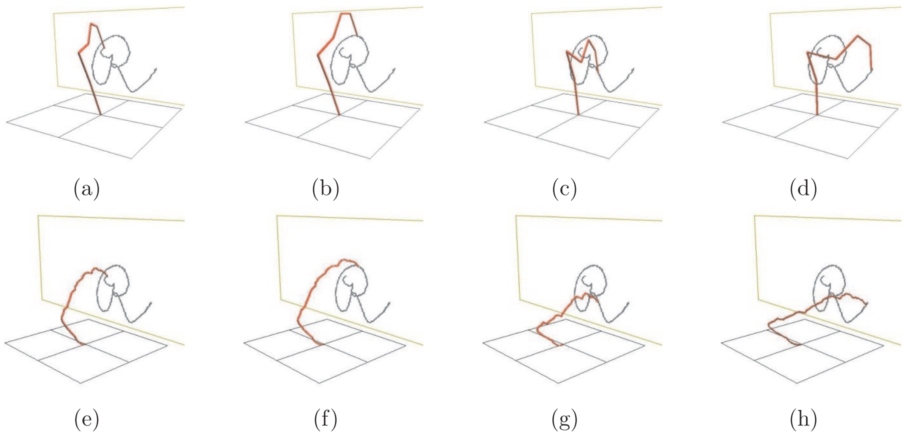

Fig. 14. A robotic arm drawing the letter “R” on the planar canvas: (a)–(d) show the drawing process of a 6-link robotic arm; (e)–(h) show the drawing process of a larger robot with 100 links.

Algorithm 4 Connect_Path

```
Input. A roadmap $R$ and configurations $q_a$ and $q_b$ where the end effector distances in $q_a$ and $q_b$ are equal.
Output. A path $P$ from $q_a$ to $q_b$.
1:  Let $N_{att}$ be the maximum number of connection attempts allowed.
2:  Let $d$ be the end effector distance in $q_a$.
3:  while $q_a$ and $q_b$ are not connected in $R$ and the number of attempts is $< N_{att}$ do
4:      Sample configuration $q$ with end effector distance $d$.
5:      Add $q$ to $R$.
6:      Let $Q$ be the $k$-closest configurations (in RD-space) in $R$ to $q$.
7:      for each $q_k \in Q$ do
8:          Let $d_k$ be the end-effector distance in $q_k$.
9:          if $d_k == d$ and there is a collision-free path between $q$ and $q_k$ using the reachable distance local planner then
10:             Add the edge $(q, q_k)$ to $R$.
11:         end if
12:     end for
13: end while
14: if the number of attempts $\geq N_{att}$ then
15:     return $\emptyset$.
16: end if
17: Let $P$ be the shortest path in $R$ from $q_a$ to $q_b$.
18: return $P$.
```

vas for a 6-link robot (Figure 14(a)–(d)) and a 100-link robot (Figure 14(e)–(h)). The target trajectory is composed of 560 strokes (or planning milestones) generated from a scanned in drawing.

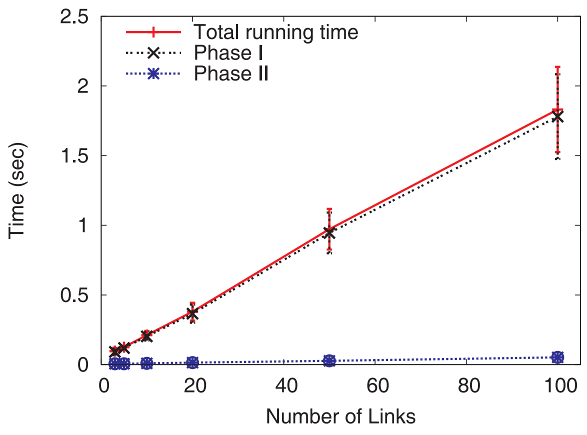

Fig. 15. The time, in seconds, for each drawing robot, averaged over 10 runs. Error bars indicate the sample standard deviation.

Table 7 shows the running time needed for each robot studied. Phase I is the time to perform the local planning and phase II is the time to morph the path to the drawing trajectory. Recall that for a two-dimensional trajectory, only the first two phases of the algorithm are needed. In all cases, the total time is very small and phase II planning is short enough for on-line applications such as drawing, monitoring, or chasing. Figure 15 shows how the planning time scales with the robot’s DOFs. As expected, phase I is linear in the number of robot links and dominates the overall planning. Phase II remains nearly constant and is thus well-suited for on-line applications. Note that other randomized planners such as PRMs or RRT would be infeasible for this application since it is even more constrained than restricted end-effector sampling where they could not generate a single valid sample for a 50-link robot in 1 hour.

Downloaded from ijr.sagepub.com at TEXAS A&M UNIV on February 24, 2012

Table 7. The time, in seconds, for each phase of the drawing process, averaged over 10 runs. Phase I consists of the local planning and phase II consists of morphing the path to the drawing trajectory. Sample standard deviations are shown in italics.

	Number of links					
	3	5	10	20	50	100
Phase I	0.091	0.119	0.203	0.366	0.945	1.780
Standard deviation	*0.023*	*0.012*	*0.030*	*0.065*	*0.146*	*0.304*
Phase II	0.006	0.006	0.009	0.014	0.027	0.052
Standard deviation	*0.001*	*0.001*	*0.000*	*0.001*	*0.001*	*0.003*
Total	0.097	0.125	0.212	0.380	0.972	1.831
Standard deviation	*0.023*	*0.012*	*0.030*	*0.064*	*0.146*	*0.306*

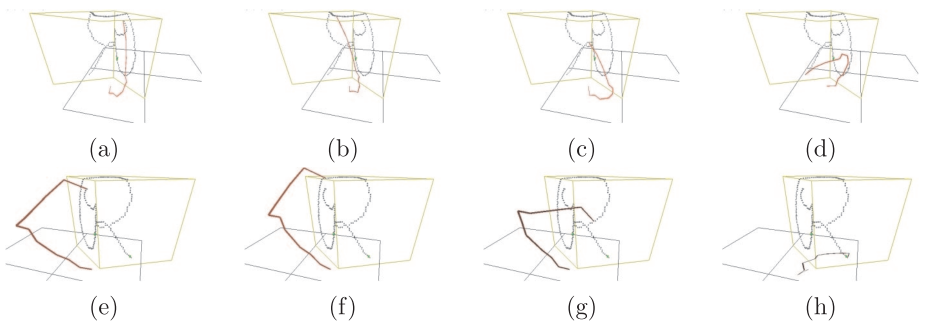
Fig. 16. A 10-link robotic arm drawing the letter "R" on the inside (a)–(d) and outside (e)–(h) of the corner of a box.

Figure 16 shows the planning results of an articulated robotic arm with 10 links drawing the same character "R" on the inside (Figure 16(a)–(d)) and outside (Figure 16(e)–(h)) of the corner of a three-dimensional box. Unlike the previous example, this trajectory necessitates the third phase of the algorithm, correcting colliding segments of the path. Figure 17 shows an example where the initial path contained three segments: a free segment, followed by a colliding segment and then another free segment. The endpoints of the colliding segment are circled. Figures 17(a) and (b) demonstrate how, when two segments are connected, the end effector is kept in the same position.

Table 8 shows the running time needed for each robot studied. The overall increase in running time between the planar example above and this 3D example results from correcting colliding segments along the initial local planner path. For the planar example, phase III is not required, however for 3D trajectory following, phase III may cause the algorithm to be recursively called many times to correct colliding segments. This also explains the large variability in running time between robots and between the two 3D examples. Even though the times have increased, all problems were solved in just a few minutes.

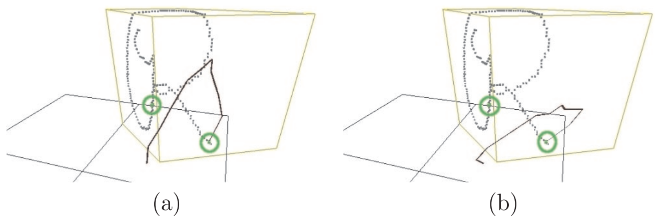
Fig. 17. The initial path for the 10-link robotic arm (from phases I and II) contained one colliding segment, endpoints circled. An example of the connecting path found during phase III is shown in (a) and (b). This path connects two path segments together while keeping the end effector in the same position.

8. Conclusion

We have presented a new method to plan the motion of spatially constrained systems based on a hierarchical representation as a set of RD-trees. We have then shown how this RD-space formulation can be applied to efficiently sample and plan motions for closed chain systems (with single and multiple

Downloaded from ijr.sagepub.com at TEXAS A&M UNIV on February 24, 2012

Table 8. Running time of different drawing robots on the inside and outside of a box, averaged over 10 runs. Sample standard deviations are shown in italics.

	Number of links					
	3	5	10	20	50	100
Inside corner	14.901	4.982	34.767	351.294	114.904	49.579
Standard deviation	*9.055*	*6.107*	*17.163*	*271.400*	*69.771*	*21.944*
Outside Corner	1.015	5.605	18.041	350.245	149.606	86.357
Standard deviation	*0.345*	*3.160*	*14.700*	*195.146*	*92.157*	*37.318*

loops), restricted end-effector sampling, and robotic arm drawing/sculpting. For all system types, our experimental results show that this formulation is fast and efficient in practice, making the cost of generating constraint satisfying configurations comparable to traditional sampling without constraints. In the future, we plan to study how our method performs for other spatially constrained systems not discussed here and in particular investigate systems with joint limits.

Acknowledgments

This research supported in part by NSF Grants EIA-0103742, ACR-0113971, CCR-0113974, ACI-0326350, CRI-0551685, CCF-0833199, CCF-0830753, by Chevron, IBM, Intel, HP, and by King Abdullah University of Science and Technology (KAUST) Award KUS-C1-016-04. S. Thomas was supported in part by a NSF Graduate Research Fellowship, a PEO Scholarship, a Department of Education Graduate Fellowship (GAANN), and an IBM T.J. Watson PhD Fellowship. The work of X. Tang was performed when he was a PhD student in the Department of Computer Science and Engineering at Texas A&M University.

References

Bayazit, O. B., Xie, D. and Amato, N. M. (2005). Iterative relaxation of constraints: A framework for improving automated motion planning. *Proceedings IEEE International Conference Intelligent Robotic Systems (IROS)*, Edmonton, Alberta, Canada, pp. 3433–3440.

Cortés, J. and Siméon, T. (2003). Probabilistic motion planning for parallel mechanisms. *Proceedings IEEE International Conference Robotics and Automation (ICRA)*, Taipei, Taiwan, pp. 4354–4359.

Cortés, J. and Siméon, T. (2005). Sampling-based motion planning under kinematic loop-closure constraints. *Algorithmic Foundations of Robotics VI*. Berlin, Springer, pp. 75–90.

Cortés, J., Siméon, T. and Laumond, J. P. (2002). A random loop generator for planning the motions of closed kinematic chains using PRM methods. *Proceedings IEEE International Conference Robotics and Automation (ICRA)*, Washington, DC, pp. 2141–2146.

Garber, M. and Lin, M. C. (2003). Constraint-based motion planning using Voronoi diagrams. In *Algorithmic Foundations of Robotics V*. Berlin, Springer, pp. 541–558.

Gharbi, M., Cortés, J. and Siméon, T. (2008). A sampling-based planner for dual-arm manipulation. *Proceedings IEEE/ASME International Conference Advances in Mechanics (AIM)*, Xi'an, China.

Han, L. (2004). Hybrid probabilistic roadmap—Monte Carlo motion planning for closed chain systems with spherical joints. *Proceedings IEEE International Conference Robotics and Automation (ICRA)*, New Orleans, LA, pp. 920–926.

Han, L. and Amato, N. M. (2001). A kinematics-based probabilistic roadmap method for closed chain systems. *New Directions in Algorithmic and Computational Robotics*. Boston, MA, A. K. Peters, pp. 233–246.

Han, L. and Rudolph, L. (2007a). Inverse kinematics for a serial chain with joints under distance constraints. *Robotics Science and Systems II*. Cambridge, MA, The MIT Press, pp. 177–184.

Han, L. and Rudolph, L. (2007b). A unified geometric approach for inverse kinematics of a spatial chain with spherical joints. *Proceedings IEEE International Conference Robotics and Automation (ICRA)*, Roma, Italy, pp. 4420–4427.

Han, L. and Rudolph, L. (2009). Simplex-tree based kinematics of foldable objects as multi-body systems involving loops. *Robotics Science and Systems IV*. Cambridge, MA, The MIT Press.

Han, L., Rudolph, L., Blumenthal, J. and Valodzin, I. (2006). Stratified deformation space and path planning for a planar closed chain with revolute joints. *Proceedings Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, New York, NY.

Kallmann, M., Aubel, A., Abaci, T. and Thalmann, D. (2003). Planning collision-free reaching motion for interactive ob-

Downloaded from ijr.sagepub.com at TEXAS A&M UNIV on February 24, 2012

ject manipulation and grasping. *Computer Graphics Forum*, **22**(3): 313–322.

Kavraki, L. E., Lamiraux, F. and Holleman, C. (1998). Towards planning for elastic objects. *Robotics: The Algorithmic Perspective*. Natick, MA, A. K. Peters, pp. 313–325.

Kavraki, L. E., Švestka, P., Latombe, J. C. and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, **12**(4): 566–580.

Khatib, O., Yokoi, K., Chang, K., Ruspini, D., Holmberg, R. and Casal, A. (1996). Vehicle/arm coordination and multiple mobile manipulator decentralized cooperation. *Proceedings IEEE International Conference Intelligent Robotic Systems (IROS)*, Osaka, Japan, pp. 546–553.

Kotay, K., Rus, D., Vona, M. and McGray, C. (1998). The self-reconfiguring robotic molecule: Design and control algorithms. *Robotics: The Algorithmic Perspective*, Agarwal, P. K., Kavraki, L. E. and Mason, M. T. (eds). Boston, MA, A. K. Peters, pp. 375–386.

Ladd, A. M. and Kavraki, L. E. (2004). Using motion planning for knot untangling. *The International Journal of Robotics Research*, **23**(7–8): 797–808.

Lamiraux, F. and Kavraki, L. E. (1999). Path planning for elastic plates under manipulation constraints. *Proceedings IEEE International Conference Robotics and Automation (ICRA)*, Detroit, MI, pp. 151–156.

Lamiraux, F. and Kavraki, L. E. (2001). Planning paths for elastic objects under manipulation constraints. *The International Journal of Robotics Research*, **20**(3): 188–208.

Latombe, J.-C. (1991). *Robot Motion Planning*. Boston, MA, Kluwer.

LaValle, S., Yakey, J. and Kavraki, L. (1999). A probabilistic roadmap approach for systems with closed kinematic chains. *Proceedings IEEE International Conference Robotics and Automation (ICRA)*, Detroit, MI, pp. 1671–1676.

LaValle, S. M. and Kuffner, J. J. (2001). Rapidly-exploring random trees: Progress and prospects. *New Directions in Algorithmic and Computational Robotics*, Natick, MA, A. K. Peters, pp. 293–308.

Lee, C. S. G. and Ziegler, M. (1984). Geometric approach in solving inverse kinematics of puma robots. *IEEE Transactions on Aerospace and Electronic Systems*, **20**(6): 695–706.

Liu, G. and Trinkle, J. C. (2005). Complete path planning for planar closed chains among point obstacles. *Robotics Science and Systems I*. Cambridge, MA, The MIT Press, 33–40.

Liu, G., Trinkle, J. C. and Shvalb, N. (2006). Motion planning for a class of planar closed-chain manipulators. *Proceedings IEEE International Conference Robotics and Automation (ICRA)*, Orlando, FL, pp. 133–138.

Merlet, J.-P. (2002). Still a long way to go on the road for parallel mechanisms. *ASME Bienneal Mechanisms and Robotics Conference*, Montreal, Canada.

Milgram, R. J. and Trinkle, J. C. (2004). The geometry of configuration spaces for closed chains in two and three dimensions. *Homology, Homotopy and Applications*, **6**(1): 237–267.

Moll, M. and Kavraki, L. E. (2006). Path planning for deformable linear objects. *IEEE Transactions on Robotics and Automation*, **22**(4): 625–636.

Nguyen, A., Guibas, L. J. and Yim, M. (2001). Controlled module density helps reconfiguration planning. *New Directions in Algorithmic and Computational Robotics*. Boston, MA, A. K. Peters, pp. 23–36.

Oriolo, G. and Mongillo, C. (2005). Motion planning for mobile manipulators along given end-effector paths. *Proceedings IEEE International Conference Robotics and Automation (ICRA)*, Barcelona, Spain, pp. 2154–2160.

Oriolo, G., Ottavi, M. and Vendittelli, M. (2002). Probabilistic motion planning for redundant robots along given end-effector paths. *Proceedings IEEE International Conference Intelligent Robotic Systems (IROS)*, Lausanne, Switzerland, pp. 1657–1662.

Reif, J. H. (1979). Complexity of the mover's problem and generalizations. *Proceedings IEEE Symposium on Foundations of Computer Science (FOCS)*, San Juan, Puerto Rico, pp. 421–427.

Saha, M. and Isto, P. (2006). Motion planning for robotic manipulation of deformable linear objects. *Proceedings IEEE International Conference Robotics and Automation (ICRA)*, Orlando, FL, pp. 2478–2484.

Shvalb, N., Shoham, M., Liu, G. and Trinkle, J. C. (2007). Motion planning for a class of planar closed-chain manipulators. *The International Journal of Robotics Research*, **26**(5): 457–473.

Singh, A. P., Latombe, J.-C. and Brutlag, D. L. (1999). A motion planning approach to flexible ligand binding. *International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pp. 252–261.

Stilman, M. (2007). Task constrained motion planning in robot joint space. *Proceedings IEEE International Conference Intelligent Robotic Systems (IROS)*, pp. 3074–3081.

Tang, X., Thomas, S. and Amato, N. M. (2007). Planning with reachable distances: Fast enforcement of closure constraints. *Proceedings IEEE International Conference Robotics and Automation (ICRA)*, Roma, Italy, pp. 2694–2699.

Tang, X., Thomas, S. and Amato, N. M. (2008). Planning with reachable distances. *Proceedings International Workshop on Algorithmic Foundations of Robotics (WAFR)*, Guanajuato, México.

Trinkle, J. C. and Milgram, R. J. (2002). Complete path planning for closed kinematic chains with spherical joints. *The International Journal of Robotics Research*, **21**(9): 773–789.

Downloaded from ijr.sagepub.com at TEXAS A&M UNIV on February 24, 2012

Whitney, H. (1932). Non-separable and planar graphs. *Transactions of the American Mathematical Society*, **34**: 339–362.

Whitney, H. (1933). 2-isomorphic graphs. *American Journal of Mathematics*, **55**: 245–254.

Xie, D. and Amato, N. M. (2004). A kinematics-based probabilistic roadmap method for high DOF closed chain systems. *Proceedings IEEE International Conference Robotics and Automation (ICRA)*, New Orleans, LA, pp. 473–478.

Yakey, J. H., LaValle, S. M. and Kavraki, L. E. (2001). Randomized path planning for linkages with closed kinematic chains. *IEEE Transactions on Robotics and Automation*, **17**(6): 951–958.

Yao, Z. and Gupta, K. (2005). Path planning with general end-effector constraints: using task space to guide configuration space search. *Proceedings IEEE International Conference Intelligent Robotic Systems (IROS)*, Edmonton, Alberta, Canada, pp. 1875–1880.

Yao, Z. and Gupta, K. (2006). Self-motion graph in path planning for redundant robots along specified end-effector paths. *Proceedings IEEE International Conference Robotics and Automation (ICRA)*, Orlando, FL, 2004–2009.

Downloaded from ijr.sagepub.com at TEXAS A&M UNIV on February 24, 2012