Planning with Reachable Distances

Xinyu Tang[1], Shawna Thomas[2], and Nancy M. Amato[2] *

[1] Google, 1600 Amphitheatre Parkway, Mountain View, CA 94043 USA, xtang@google.com
[2] Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843 USA, {sthomas,amato}@cse.tamu.edu

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 robot's complexity. 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 spatially constrained systems including closed chain planning with multiple loops, restricted end effector sampling, and on-line planning for drawing (or sculpting).

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 [18], grasping for single and multiple robots [13], reconfigurable robots [14, 20], closed molecular chains [25], and computer animation [11]. 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 determin-

* The work of X. Tang was done when he was a Ph.D. student in the Department of Computer Science and Engineering at Texas A&M University. This research supported in part by NSF Grants EIA-0103742, ACR-0081510, ACR-0113971, CCR-0113974, ACI-0326350, CRI-0551685, by the DOE and NERSC. Thomas supported in part by an NSF Graduate Research Fellowship, a PEO Scholarship, a Department of Education Graduate Fellowship (GAANN), and an IBM TJ Watson PhD Fellowship.

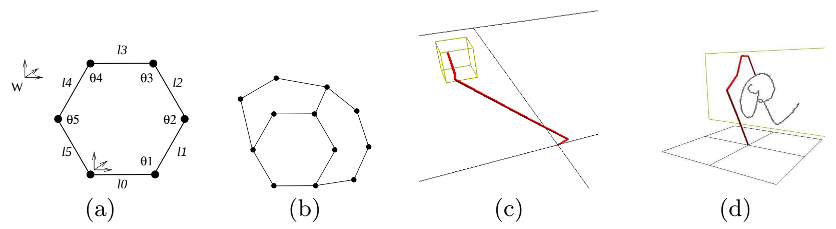


Fig. 1. Examples of different spatially constrained systems. Each must satisfy certain closure constraints. (a) Single loop: loop must remain closed. W is the world coordinate frame. (b) Multiple loops: all loops must remain closed. (c) End-effector restrictions: the end effector must remain within the 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.

istic algorithms [23, 15] is exponential in the complexity of the robot, they are impractical for most systems of practical interest. While sampling-based planning methods are successful for many high degree of freedom systems, their performance degrades for spatially constrained systems in which the set of valid configurations occupies a small volume of configuration space (the set of all robot 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 [16]).

In this paper, we generalize the reachable distance representation we proposed for single-loop closed chains [26] to handle other types of spatial constraints in addition to closure constraints and to satisfy multiple constraints simultaneously. Our framework handles a wide range of systems including both 2D and 3D chains, single and multiple loops, end effector placement/trajectory requirements, and prismatic joints. This new representation for spatially constrained systems, called *reachable distance 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 reachable distance space.

We describe our new representation, RD-space, in Section 3 and give a recursive sampling algorithm with complexity linear in the size of the system 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 on 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 degrees of freedom and show that it outperforms other randomized sampling methods such as PRM [12] and RRT[17] and other specialized methods for planning for closed chains.

2 Related Work

In theory, exact motion planning algorithms such as [23, 15] 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). However, since these algorithms are exponential in the dimension of the configuration space, they are generally impractical for systems with more than 4 or 5 degrees of freedom.

Randomized algorithms such as Probabilistic Roadmap Methods (PRMs) [12] and Rapidly-exploring Randomized Trees (RRTs) [17] 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 degree of freedom systems, their performance degrades for spatially constrained systems as the probability of randomly sampling a configuration satisfying the constraints approaches zero (see, e.g., [16]).

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. In the first sampling-based method for closed chains, Kavraki et al. [16, 29] randomly sample configurations and then apply an iterative random gradient descent method to push the configuration to the constraint surface. They solved planar closed chains with up to 8 links and 2 loops in several hours with PRM [16] and several minutes with RRT[29]. kinematics-based PRM (KBPRM) [6] first builds a roadmap ignoring all obstacles, and then populating the environment with copies of this kinematics roadmap, removing invalid portions, and connecting configuration copies with a rigid-body local planner. This method can solve closed chain problems with 7–9 links in under a minute. KBPRM has been extended to better handle larger linkages [28], reduce running time [3, 2], and deal with multiple loops [1]. PRM-MC combines PRMs and Monte Carlo simulations to guarantee closure constraints [5]. They can generate 100 samples of a 100-link closed chain and a 2-loop system containing 16 links in under a minute. Trinkle and Milgram proposed a path planning algorithm [27] based on configuration space analysis [19] that does not consider self-collision. They extend this method to handle planar parallel star-shaped manipulators [24]. Han et al. [10] proposed a set of geometric parameters for closed chain systems such that the problem can be reformulated as a system of linear inequalities. They extend this work to handle multiple loops [9]. They show results for a 1000-link closed chain and a 1000-loop system containing 3000 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. However, our work proposes a different set of parameters and provides a framework for both sampling configurations and naturally results in a simple local planner to connect configurations.

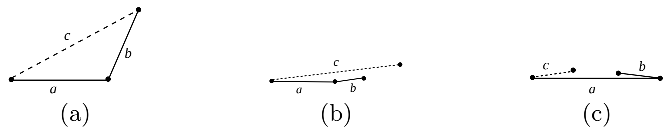


Fig. 2. Three configurations of an articulated system composed of 2 variable length links a and b where the distance between the base and the end effector needs to satisfy spatial constraint c. (a) a and b are of appropriate length. (b) The combined lengths of a and b are too short. (c) The combined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

In addition to closed chain systems, there has been work on other types of spatially constrained systems. Garber and Lin [4] use constrained dynamics to plan motions to ensure constraints such as joint connectivity, the spatial relationship between multiple robots, or obstacle avoidance. The show results for a 6 dof robot arm. Oriolo et al. plan articulated motions where the end effector must travel along a given trajectory for fixed-base manipulators [22] and mobile manipulators [21]. They extend probabilistic planning methods for this system by generating samples that satisfy the end effector trajectory constraint. They present results for up to a 6 dof robot arm on a mobile planar base. In other work, Yao and Gupta [30] propose two approaches, Alternate Task-space and Configuration-space Exploration (ATACE) and randomized gradient descent, to plan paths for manipulators with general end effector constraints. are presented for up to a 9 dof robot. Han et al. [8] unify their work on serial chains [7] and closed chains [10] to provide inverse kinematic solutions that satisfy 3D, 5D, and 6D end effector placement constraints. They can sample a single configuration for a 1000-link arm in 19 ms.

3 Reachable Distance Framework

In this section, we describe a reachable distance data structure for planning motion of general spatially constrained systems. The reachable distance data structure is a hierarchy of reachable distances defined by recursively partitioning the original robot system into smaller sub-systems. This is a generalized version of the data structure we developed in our previous work for single closed loop robots [26]. 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 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*, i.e., the set of distances/lengths which allow the spatial constraints involved in the

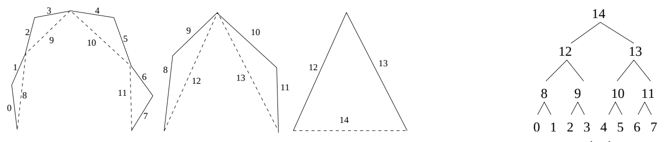


Fig. 3. (a) Articulated linkage with 8 links (labeled 0–7). Two consecutive links form a parent virtual link (dashed lines). This repeats to form a hierarchy (b) where the parents in one level become the children in the next level.

sub-chain to be satisfied. In this case, the *available reachable range* 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 *available reachable range* 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 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 framework can handle many types of systems including both 2D and 3D chains, single and multiple loops, end effector requirements, and prismatic joints.

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*). The virtual link and its children form a sub-chain. For example, in Figure 3(a), the *virtual link* 8 is the parent of the two actual links 0 and 1, while link 12 is the parent of virtual links 8 and 9. We iteratively build higher level virtual links until we obtain a single virtual link (14) at the highest level, see Figure 3(b).

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

The reachable range of a parent can be calculated from the ranges of its children. If we always build a virtual link using 0 or 2 children, reachable ranges are computed as follows. Consider a virtual link with no children. It has only 1 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, $l_{min} = l_{max}$.) The reachable range is then $[l_{min}, l_{max}]$.

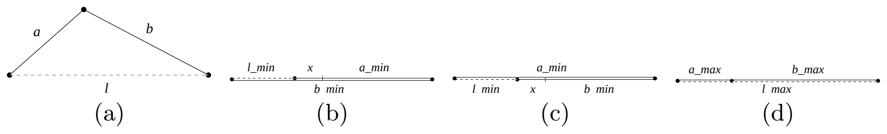


Fig. 4. (a) A sub-chain with 2 children a and b and its virtual link l. (b) A configuration where l is minimum when $a_{min} < b_{min}$. $x = b_{min} - a_{min}$ if $a_{max} > b_{min}$ and $x = a_{max} - a_{min}$ otherwise. (c) A configuration where l is minimum when $a_{min} > b_{min}$. $x = a_{min} - b_{min}$ if $b_{max} > a_{min}$ and $x = b_{max} - b_{min}$ otherwise. (d) A configuration where l is maximum.

Now consider the virtual link l with 2 children a and b in Figure 4(a). They form a triangle. Let $[a_{min}, a_{max}]$ be the reachable range of the first child and $[b_{min}, b_{max}]$ be the reachable range of the second child. (If a or b is an actual link, its reachable range is defined by the problem, otherwise it is a function of the reachable ranges of its children.) The reachable range of the parent is then $[l_{min}, l_{max}]$ where

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

and

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

Note that these equations are not limited to computing reachable ranges of parent links. Given the reachable ranges of any two links in the same triangular sub-chain, these equations calculate the reachable range of the remaining link to satisfy the triangle inequality.

Available Reachable Range of a Virtual Link. The *available reachable range* of a virtual link 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 available reachable range is a subset of the reachable range in which the spatial constraints may be satisfied. It is a function of the available reachable ranges (or lengths) of the other links in the same sub-chain loop. Changes in the available reachable range of one link cause changes in the available reachable ranges 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 available reachable range of each link is equal to its reachable range. However, as we sample and enforce spatial constraints by fixing the length of a link, portions of the reachable ranges of the other links in the same sub-chain may no longer be available. When this happens, we need to update the available reachable ranges in the other links in the sub-chain. We can do this using Equations 1 and 2. So, given a specified value of one link we can find the available reachable range of the other two in the sub-chain and sample available lengths for them.

Now we have a set of lengths, one for each virtual link, for a configuration that satisfies the spatial constraints. We can then compute the joint angles between virtual links using only basic trigonometry functions. We will illustrate reachable distance sampling through example applications in more detail in the following sections.

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 Reachable Distance Tree

Sampling in the reachable distance space involves recursively sampling virtual link lengths from their available reachable range and updating the available reachable ranges in the sub-chain containing the virtual link. This ensures that the resulting set of lengths satisfies any spatial constraints defined by the reachable distance tree. Once all the lengths are sampled, we then sample the orientation of each sub-chain (e.g., concave or convex for sub-chains in the plane, dihedral angles for non-planar sub-chains). We can then compute appropriate joint angles from the virtual link lengths and orientations. Note that this sampling does not necessarily determine validity, i.e., it will still need to be checked for 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. Exploring this relationship is the subject of future work. We describe each step below.

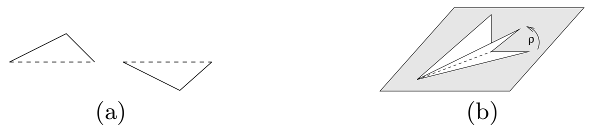


Fig. 5. (a) In 2D, the same virtual link represents two configurations: a concave triangle and a convex triangle. (b) In 3D, the same virtual link represents many configurations with different dihedral angles ρ.

Recursively sample link lengths. Recall that we define the available reachable range as the subset of the reachable range that satisfies the spatial constraints with respect to the rest of the sub-chain. Once we fix the length of an available reachable range in a sub-chain, the other available reachable ranges may be restricted. Thus, we can generate a configuration by sampling reachable distances and updating available reachable ranges starting at the the root of the hierarchy and recursing until all sub-chain reachable distances are sampled. Note that by sampling in this way, an available reachable range may never become empty; in its most constrained case, its minimum and maximum may become equal.

The sampling algorithm is called once for each virtual link. Recall that each virtual link is represented by a single internal node in the hierarchy of sub-chains. Because 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 complexity of the problem.

Sample link orientations. Each sub-chain forms a triangle, and there are multiple configurations with the same virtual link length: there are two in 2D (i.e., concave and convex, see Figure 5(a)), and there are many in 3D 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 virtual link.

Calculate joint angles. Consider the joint angle θ between links a and b. Links a and b are connected to a virtual link 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: $\theta = \arccos(\frac{l_a^2+l_b^2-l_c^2}{2l_a l_b})$.

4.2 Local Planning

Based on the presentation of reachable distances, we propose a local planner for spatially-constrained systems. Given 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 reachable distance space to determine the sequence of configurations.

To generate the sequence of intermediate configurations, this local planner interpolates the lengths of each virtual link between q_s and q_g. We then must determine the virtual link's orientation sequence to fully describe the sequence of configurations. In 2D, 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 3D, we simply interpolate between the dihedral angle in q_s and the dihedral angle in q_g. In addition to checking collision as with other local planners, we determine the validity of the sequence by checking that each virtual link's length is in its available reachable range.

Concave/convex flipping for 2D chains. Consider the two configurations in Figure 5(a) with different orientations. To flip the virtual link, we need to expand (or open) the parent's virtual link enough so their children can change orientation while remaining in their available reachable range. In other words, at some point the parent's available reachable range 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 available reachable range for this minimum "flat" constraint. If available, we sample a configuration where the virtual links 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 [18], closed molecular chains [25], animation [11], reconfigurable robots [14, 20], and grasping for single and multiple robots [13]. 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 [16].

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. In our preliminary work [26], we demonstrated how this method efficiently samples closed configurations for single-loop closed chains with thousands of links in just seconds and can be used in a complete motion planning framework. Here we show results for a multiple loop system. All experiments were performed on a 3GHz desktop computer and were compiled using gcc4 under linux. Our current implementation supports planar joints and spherical joints.

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 framework, we simply require that the distance between the first link and the last link is zero by making the root virtual link length zero.

Handling multiple loops. For a closed chain system that has more than one loop, we construct a reachable distance tree for each loop. Then the whole system can be represented by a forest of reachable distance trees. 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 available reachable range of a common node should be the intersection of the available reachable ranges of the same virtual link 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.

Figure 6 shows an example of a closed chain system with two loops (a) and the corresponding forest (b). Two trees rooted at node 17 and 18, respectively, correspond to the two loops in this system. Both trees share the same node 12. When we update the available reachable range of 12, we should consider the available reachable range of both link 15 and link 16. Given such a reachable distance forest, we can perform the sampling and planning on each loop

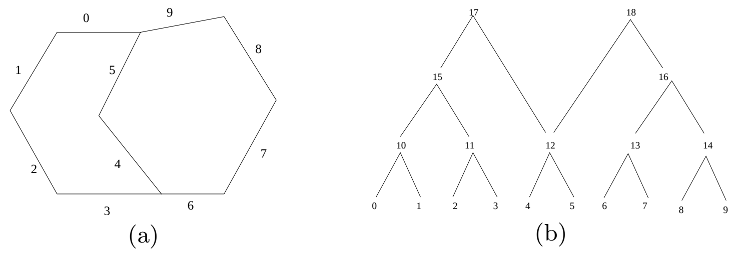


Fig. 6. (a) Multiple loop system and (b) corresponding forest hierarchy.

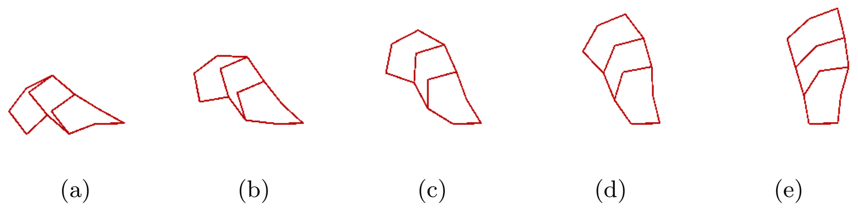


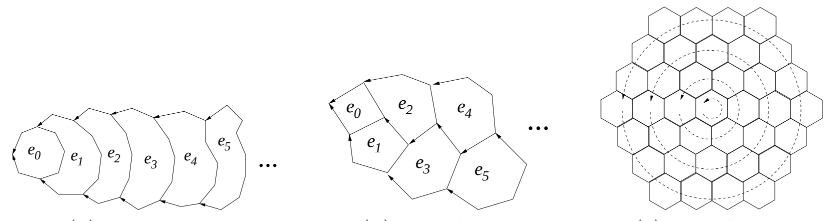
Fig. 7. A path for a three-loop closed chain system from (a) to (e).

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.

Figure 7 shows an example of a 3-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, we study the time required to generate 1000 samples for multiple loop systems. In previous work [26], we performed a similar study with single loop closed chain systems. There we compared to the original KBPRM [6] and to the extension for larger linkages [28] which was already demonstrated to perform better than existing closed chain methods such as Randomized Gradient Descent [16, 29]. Our method was able to scale to thousands of links while the KBPRM extension, the closest competitor, was only able to generate samples for loops up to 200 links in the 12 hour time limit and the original KBPRM was only able to generate samples for loops up to 20 links. We were unable to compare to [7, 10] because we do not have access to their implementation written in Matlab. They report that they can generate a single closed conformation for a 1000-link chain in 19 ms on a "desktop PC" (processor speed not reported).

Here we look at three different multiple loop topologies, see Figure 8. 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 1 to 4. Figure 9 summarizes the results.All results are averaged over 10 runs. Even with 1024 links and 256 loops to close in topologies 1 and 2, our method takes

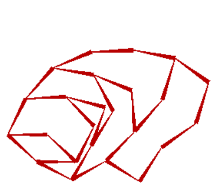


(a) Topology 1 (b) Topology 2 (c) Topology 3

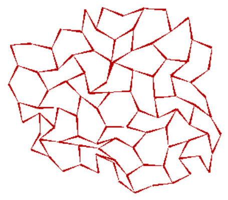
Fig. 8. Multiple loop topologies studied. (a) Example with 8-link ears (indicated with arrows). The ears are arranged such that the endpoints of ear e_i connect to ear e_{i-1}. (b) Example with 4-link ears (indicated with arrows). The ears are arranged such that the endpoints of ear e_i connect to ears e_{i-1} and e_{i-2}. (c) "Honeycomb" example with 4 rings (indicated with dashed arrows).

# Loops	Loop Size	Time (s) T1	Time (s) T2
1	1024	5.085	
2	512	9.403	
4	256	12.422	12.332
8	128	16.059	15.640
16	64	17.770	16.598
32	32	24.910	22.038
64	16	38.526	33.053
128	8	57.139	49.272
256	4	88.5424	80.222

(a)



(c)



(d)

# Links	# Loops	Time (s) No CD T1	No CD T2	No CD T3	With CD T1	With CD T2	With CD T3
6	1	0.009			0.014		
30	7	0.324	0.321	0.320	1.595	402.378	1.611
73	19	1.137	1.140	2.707	12.670	68.097	181.678
133	37	2.623	2.415	10.167	144.879	1182.302	55067.010

(b)

Fig. 9. Running time to generate 100 samples of multiple loop systems, averaged over 10 runs. (a) Results for topologies 1 and 2 (T1 and T2) with 1024 links. (b) Results for all three topologies, both with and without collision detection. Example configuration for (c) topology 2 with 30 links and (d) topology 4 with 133 links.

under 1.5 minutes to generate 100 samples. Figure 9(b) 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 9(c). We were unable to directly compare to [9] 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).

6 Application: Restricted End Effector Sampling

Here we discuss how to apply this reachable distance framework 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. 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 is unlikely. Recall that the reachable range of this robot is $[l_{min}, l]$ where l is the sum of its link lengths, and l_{min} is the minimum available reachable range 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 *available reachable range* of the end effector from $[l_{min}, l]$ to $[d_{min}, d_{max}]$ before we sample the rest of the reachable distance tree.

Enforcing end effector placement. We can easily restrict the end effector distance by setting the available reachable range of the virtual link connecting the base and the end effector 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 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 available reachable range of the virtual link connecting the base and the end effector to $[b, b]$ and sample as before. Let e be the resulting end effector placement. We then compute a rotation R to rotate e to b and apply this rotation to the robot base. Thus, we can easily sample configurations that keep the end effector inside B.

Results. Here we compare the performance of our sampling scheme in reachable distance space to uniform random and RRT-style sampling [17] in joint space. For RRT-style sampling, we only count the time to sample nodes and check their validity and do not include the time spent generating and checking edges. 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.

Table 1 shows how each sampler performs in practice. Each sampler was asked to generate 1000 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 3 links.

Method	# Robot Links	Time (s)	Samples Generated	Sample Attempts
Reachable Distance	3	0.02	1000.0	1000.5
	10	0.11	1000.0	1817.8
	20	0.50	1000.0	4311.9
	50	7.13	1000.0	24486.2
	100	29.29	1000.0	51835.3
Uniform Random	3	13.89	1000.0	1326106.9
	10	142.04	1000.0	5418904.7
	20	358.55	61.7	7205750.0
	50	n/a	n/a	n/a
	100	n/a	n/a	n/a
RRT	3	182.76	1000.0	392550.3
	10	49.67	1000.0	106202.3
	20	61.11	1000.0	121840.4
	50	n/a	n/a	n/a
	100	n/a	n/a	n/a

Table 1. Restricted end effector sampling comparison.

Clearly, reachable distance sampling out-performs the other randomized sampling strategies for restricted end effector sampling.

Note that RRT performs significantly slower for the 3 link robot than for the other larger robots. The success of RRT depends on the placement of the starting configuration. For the 3 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 3 link robot.

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. 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.

Enforcing the End Effector Trajectory. Again we can take advantage of the reachable distance tree. We propose a two phase approach. With a local planner, we first find a valid path between q_{min} and q_{max} where q_{min} (q_{max}) is a configuration with end effector distance d_{min} (d_{max}) and d_{min} (d_{max}) is the minimum (maximum) distance between the base and any point in the drawing trajectory. We store the path's configurations in a hash table using the end effector distance as the key. We then use the pre-computed path (as stored in the hash table) 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. Note that if the local planner returns a path that contains

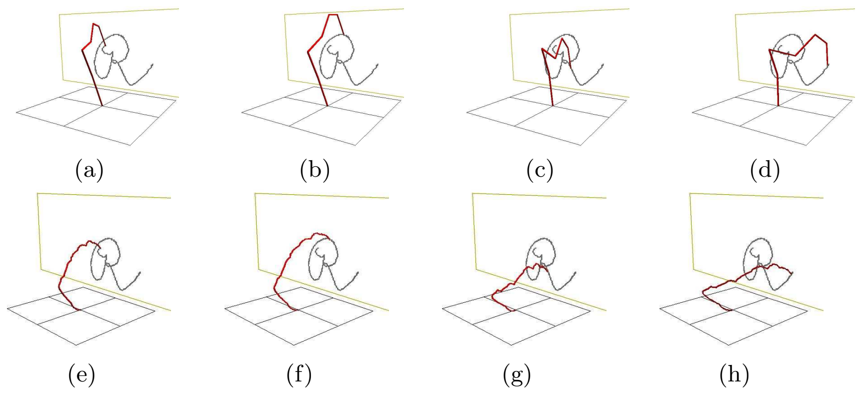
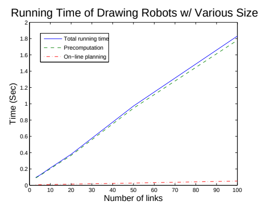


Fig. 10. A robotic arm drawing letter "R". (a)-(d) show the drawing process of a 6-link robotic arm. (e)-(h) show the drawing process of a larger robot with 100 links.

	Number of Links					
Time (s)	3	5	10	20	50	100
Phase I	0.090	0.119	0.203	0.366	0.945	1.780
Phase II	0.006	0.006	0.009	0.014	0.027	0.052
Total	0.096	0.125	0.212	0.380	0.972	1.831

(a)



(b)

Fig. 11. (a) Running time of different drawing robots averaged over 10 runs. (b) Running time of drawing robot with different number of links.

configurations with end effectors outside the range $[d_{min}, d_{max}]$, we simply clip these portions out of the path. Beyond drawing, this algorithm could also be used by other on-line planning applications such as chasing or monitoring.

Results. We applied our drawing algorithm to robots with 3, 5, 10, 20, 50, 100 links. Figure 10 shows the planning results of an articulated robotic arm drawing the character "R" on a canvas for the 6 link robot (a–d) and the 100 link robot (e–h). The target trajectory is composed of 560 strokes (or planning milestones) generated from a scanned in drawing.

The table in Figure 11(a) 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. In all cases, the total time is very small and phase II planning is short enough for on-line applications. Figure 11(b) shows how the planning time scales with the robot's degrees of freedom. As expected, the phase I pre-computation 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.

8 Conclusion

We presented a new method to plan the motion of spatially constrained systems based on a hierarchical representation as or forest of reachable distances. We then show how this framework can be applied to efficiently sample and plan motions for closed chain systems (with single and multiple loops), restricted end effector sampling, and robotic arm drawing/sculpting. For all system types, our experimental results show that this framework is fast and efficient in practice, making the cost of generating constraint satisfying configurations comparable to traditional sampling without constraints.

References

1. J. Cortés and T. Siméon. Probabilistic motion planning for parallel mechanisms. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4354–4359, Taipei, Taiwan, 2003.
2. J. Cortés and T. Siméon. Sampling-based motion planning under kinematic loop-closure constraints. In *Algorithmic Foundations of Robotics VI*, pages 75–90. Springer, Berlin/Heidelberg, 2005.
3. J. Cortés, T. Siméon, and J. P. Laumond. A random loop generator for planning the motions of closed kinematic chains using PRM methods. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2141–2146, Washington, DC, 2002.
4. M. Garber and M. C. Lin. Constraint-based motion planning using Voronoi diagrams. In *Algorithmic Foundations of Robotics V*, pages 541–558. Springer, Berlin/Heidelberg, 2003.
5. L. Han. Hybrid probabilistic roadmap — Monte Carlo motion planning for closed chain systems with spherical joints. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 920–926, New Orleans, LA, 2004.
6. L. Han and N. M. Amato. A kinematics-based probabilistic roadmap method for closed chain systems. In *New Directions in Algorithmic and Computational Robotics*, pages 233–246. A. K. Peters, Boston, MA, 2001.
7. L. Han and L. Rudolph. Inverse kinematics for a serial chain with joints under distance constraints. In *Robotics Science and Systems II*, pages 177–184. The MIT Press, Cambridge, MA, 2007.
8. L. Han and L. Rudolph. A unified geometric approach for inverse kinematics of a spatial chain with spherical joints. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4420–4427, Roma, Italy, 2007.
9. L. Han and L. Rudolph. Simplex-tree based kinematics of foldable objects as multi-body systems involving loops. In *Robotics Science and Systems IV*, page to appear. The MIT Press, Cambridge, MA, 2009.
10. L. Han, L. Rudolph, J. Blumenthal, and I. Valodzin. Stratified deformation space and path planning for a planar closed chain with revolute joints. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, New York, NY, 2006.
11. M. Kallmann, A. Aubel, T. Abaci, and D. Thalmann. Planning collision-free reaching motion for interactive object manipulation and grasping. *Computer Graphics Forum*, 22(3):313–322, Sept. 2003.
12. L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, 12(4):566–580, August 1996.

13. O. Khatib, K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, and A. Casal. Vehicle/arm coordination and multiple mobile manipulator decentralized cooperation. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 546–553, Osaka, Japan, 1996.
14. K. Kotay, D. Rus, M. Vona, and C. McGray. The self-reconfiguring robotic molecule: Design and control algorithms. In P. K. Agarwal, L. E. Kavraki, and M. T. Mason, editors, *Robotics: The Algorithmic Perspective*, pages 375–386. A. K. Peters, Boston, MA, 1998.
15. J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, MA, 1991.
16. S. LaValle, J. Yakey, and L. Kavraki. A probabilistic roadmap approach for systems with closed kinematic chains. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1671–1676, Detroit, MI, 1999.
17. S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *New Directions in Algorithmic and Computational Robotics*, pages 293–308. A. K. Peters, 2001.
18. J.-P. Merlet. Still a long way to go on the road for parallel mechanisms. In *ASME Bienneal Mech. Rob. Conf.*, Montreal, Canada, 2002.
19. R. J. Milgram and J. C. Trinkle. The geometry of configuration spaces for closed chains in two and three dimensions. *Homology, Homotopy Appl.*, 6(1):237–267, 2004.
20. A. Nguyen, L. J. Guibas, and M. Yim. Controlled module density helps reconfiguration planning. In *New Directions in Algorithmic and Computational Robotics*, pages 23–36. A. K. Peters, Boston, MA, 2001.
21. G. Oriolo and C. Mongillo. Motion planning for mobile manipulators along given end-effector paths. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2154–2160, Barcelona, Spain, 2005.
22. G. Oriolo, M. Ottavi, and M. Vendittelli. Probabilistic motion planning for redundant robots along given end-effector paths. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 1657–1662, Lausanne, Switzerland, 2002.
23. J. H. Reif. Complexity of the mover's problem and generalizations. In *Proc. 1EEE Symp. Foundations of Computer Science (FOCS)*, pages 421–427, San Juan, Puerto Rico, Oct. 1979.
24. N. Shvalb, M. Shoham, G. Liu, and J. C. Trinkle. Motion planning for a class of planar closed-chain manipulators. *Int. J. Robot. Res.*, 26(5):457–473, 2007.
25. A. P. Singh, J.-C. Latombe, and D. L. Brutlag. A motion planning approach to flexible ligand binding. In *Int. Conf. on Intelligent Systems for Molecular Biology (ISMB)*, pages 252–261, 1999.
26. X. Tang, S. Thomas, and N. M. Amato. Planning with reachable distances: Fast enforcement of closure constraints. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2694–2699, Roma, Italy, 2007.
27. J. C. Trinkle and R. J. Milgram. Complete path planning for closed kinematic chains with spherical joints. *Int. J. Robot. Res.*, 21(9):773–789, 2002.
28. D. Xie and N. M. Amato. A kinematics-based probabilistic roadmap method for high DOF closed chain systems. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 473–478, New Orleans, LA, 2004.
29. J. H. Yakey, S. M. LaValle, and L. E. Kavraki. Randomized path planning for linkages with closed kinematic chains. *IEEE Trans. Robot. Automat.*, 17(6):951–958, 2001.
30. Z. Yao and K. Gupta. Path planning with general end-effector constraints: using task space to guide configuration space search. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 1875–1880, Edmonton, Alberta, Canada, 2005.