

# Efficient Planning of Spatially Constrained Robot Using Reachable Distances

Xinyu Tang                      Shawna Thomas                      Nancy M. Amato  
xinyut@cs.tamu.edu    sthomas@cs.tamu.edu    amato@cs.tamu.edu

Technical Report TR07-001  
Parasol Lab.  
Department of Computer Science  
Texas A&M University

September 18, 2006

## Abstract

Motion planning for spatially constrained robots is particularly 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. In fact, for many spatially constrained systems, the probability that a randomly sampled robot configuration satisfies the additional constraints approaches zero. It is usually too computationally expensive to compute the portion of configuration space that lies on the constraint surface.

We overcome this challenge by redefining the robot's degrees of freedom into a new set of sampling parameters that enable us to sample the constraint surface directly. For example, articulated linkages are typically sampled by randomly selecting values for each joint angle. Instead, we redefine the articulated linkage and its additional constraints into *reachable distance* space. The unique property of reachable distance space is that all configurations in that space lie on the constraint surface. Thus, we sample in reachable distance space instead of configuration space to directly sample on the constraint surface.

In previous work, we demonstrated that this reachable distance formulation is extremely efficient for sampling and planning closed chain systems. Here, we generalize our reachable distance framework to include other types of spatially constrained systems. We demonstrate its utility on a variety of spatially constrained systems including restricted end effector sampling for articulated linkages, on-line planning for drawing (or sculpting), and closed chain planning. In particular, we show that we can sample the constraint surface with complexity linear in the complexity of the robot system. Our method outperforms other randomized sampling methods such as PRM and RRT for sampling and planning for these spatially constrained systems.

# 1 Introduction

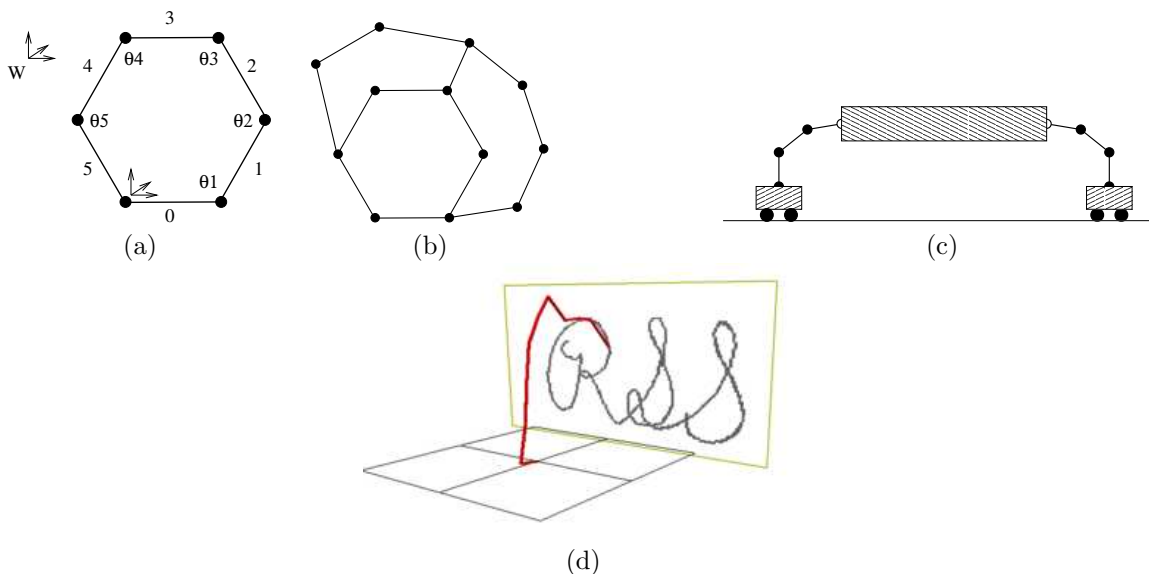


Figure 1: Examples of spatially constrained systems. Each must satisfy certain additional constraints. (a) Single loop closed chain: joint angles  $\theta_1 \dots \theta_5$  must keep the chain closed. (b) Multiple loop closed chain: generalization of single loop closed chain, all loops must remain closed. (c) Multiple robot coordination: both robots work together to transport an object; robot arms must remain in contact with object while their bases stay on the ground. (d) Drawing/sculpting: the end effector of the robotic arm must follow a desired drawing trajectory while remaining in contact with the canvas.

Spatially constrained systems are systems with additional constraints such as requiring certain parts of the system to maintain contact or to maintain a particular clearance from each other. Spatially constrained systems are widely involved in many applications in robotics and beyond, such as parallel robots [13], closed molecular chains [18], computer animation [5], reconfigurable robots [8, 15], and grasping for single and multiple robots [7]. Figure 1 gives some examples of spatially constrained systems. Motion planning for these systems is particularly challenging due to the additional spatial constraints placed on the system. Typically, the constraint surface occupies a small volume of configuration space. Thus, the probability of randomly sampling a configuration that also lies on the constraint surface approaches zero. In fact, for closed chain systems, this probability is zero [11].

Instead of randomly sampling in the configuration space, we sample in the *reachable distance* space. The reachable distance space is a new planning space defined by a set of reachable distances for the robot instead of by the joint angles as in configuration space. For example, consider an articulated linkage with  $n$  links of length  $l_1 \dots l_n$  and joint angles  $\theta_1 \dots \theta_{n-1}$ . We know that for all possible configurations (selections of  $\theta_1 \dots \theta_{n-1}$ ), the distance between the base and the end effector is in the range  $[l_{min}, \sum_1^n l_i]$  where  $l_{min}$  is the minimum achievable distance for the robot.  $l_{min}$  could be as small as 0. This range is its *reachable distance*. We can partition the robot recursively to build a reachable distance hierarchy. Suppose this robot has an additional spatial constraint that the distance between its base and end effector remain in the range  $[d_{min}, d_{max}]$ . The probability of randomly sampling  $\theta_1 \dots \theta_{n-1}$  to satisfy this constraint is  $(d_{max} - d_{min}) / ((\sum_1^n l_i) - l_{min})$ . Instead of sampling joint angles, we sample a set of reachable distances by first sampling a distance in  $[d_{min}, d_{max}]$  and then propagating the effects through the rest of the reachable distance hierarchy as we sample. We can then compute appropriate joint angles from the reachable distance hierarchy. Thus, we directly sample on the constraint surface and the probability of finding a configuration that satisfies the constraints is 1.

In previous work [citation removed for blind submission], we demonstrated how this approach can improve the efficiency of single-loop closed chain sampling and randomized planning such as *Probabilistic Roadmap Methods* (PRMs) [6]. Our results showed that our method is fast and efficient in practice making sampling configurations with closure constraints comparable to sampling open chain configurations that ignore closure constraints entirely.

In this paper, we generalize our method and extend it to other types of spatially constrained systems. We first describe this new representation and give a recursive sampling algorithm with complexity linear in the number of

links in the system. Then we present applications of our framework on three types of spatially constrained motion planning problems, including restricted end effector sampling, on-line planning for drawing (or sculpting), and closed chain planning. We demonstrate that our method is scalable to thousands of degrees of freedom making previously intractable problems solvable. Our also show that our method outperforms other randomized sampling methods such as PRM and RRT.

The paper is outlined as follows: Section 2 discusses related work, Section 3 describes the general reachable distance framework, and Section 4 – 6 provides algorithms that exploit the reachable distance framework and results for 3 different classes of spatially constrained systems. Section 7 draws conclusions of this work and discusses future directions.

## 2 Related Work

In theory, exact motion planning algorithms such as [17, 10] can handle systems with spatial constraints by explicitly computing the constraint surface 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 large systems, especially those with additional spatial constraints.

Randomized algorithms such as Probabilistic Roadmap Methods (PRMs) [6] and Rapidly-exploring Randomized Trees (RRTs) [12] are widely used today for many different applications. These methods randomly sample the configuration space (retaining only valid configurations) and connect these samples together to form a roadmap (graph/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 on the constraint surface approaches zero.

Much work has been done to optimized these sampling-based planners for closed chain systems, a spatially constrained system where the linkage must satisfy the closure constraints at all times during planning. Kavraki et al.[11, 21] randomly samples configurations and then applies an iterative random gradient descent method to push the configuration to the constraint surface. They solved planar chains with up to 8 links and 2 loops in several hours with PRM and several minutes with RRT. [3] uses a kinematics-based PRM approach by first building a roadmap ignoring all obstacles, and then populating the environment with copies of this kinematics roadmap, removing invalid portions. This method can solve chain problems with 7 – 9 links in under a minute. [1, 20] have extended this work to increase the number of links the method can handle. Trinkle and Milgram proposed a path planning algorithm [19] based on the configuration space analysis [14]. Their method does not consider self-collision but still may be applied as a local planner. Recently, Han et al. [4] proposed a set of geometric parameters for closed-chain systems such that the problem can be reformulated as a system of linear inequalities. Then linear programming and other algorithms can be used for sampling and local planning. However, they do not discuss the algorithm’s complexity or consider collisions in planning.

In addition to closed chain systems, there has been work on other types of spatially constrained systems. Garber and Lin [2] use constrained dynamics to plan motions to ensure constraints such as joint connectivity, the spatial relationship between multiple robots, or obstacle avoidance. Oriolo et al. [16] plan articulated motions where the end effector must travel along a given trajectory. They extend probabilistic planning methods for this system by generating samples that satisfy the end effector trajectory constraint. Yao and Gupta [22] propose two approaches, ATACE and RGD, to plan paths for manipulators with general end effector constraints. RGD adapts a randomized gradient descent to search the configuration space while ATACE uses task space to guide the search.

In contrast to other approaches, our method is general and can be applied to a wide variety of spatially constrained systems, including all the ones studied above. In addition, it is easy to implement and apply to sampling-based planners. It’s fast and efficient in practice with sampling complexity linear in the size of the system. Finally, our method can guarantee that a sampled configuration satisfies the additional spatial constraints or report that one cannot be obtained.

### 3 Reachable Distance Framework

Here we describe a reachable distance data structure for planning of general spatially constrained system. The reachable distance data structure is a hierarchy of reachable distances defined by recursively partitioning the original robot system into smaller sub-systems. Previously [citation removed for blind submission], we developed a reachable distance data structure for closed chain planning and in this work we generalize it to handle other spatially constrained systems. The main advantage of this representation over the traditional joint angle representation is that it encodes information on spatial constraints. 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 valid configurations.

For example, 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, called the *available range*, to satisfy the triangle inequality law:  $|a - b| \leq |c| \leq |a + b|$ . In this triangular case, the *available 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 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  $c$ . We show as follows how we extend this sampling strategy to handle more general spatially constrained systems. This scheme only ensures that the spatial constraints are satisfied. Collision checking is still required to determine the configuration’s validity.

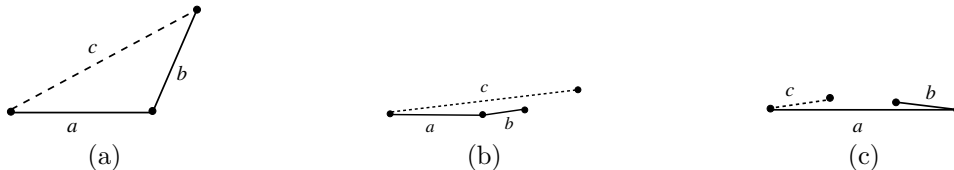


Figure 2: Three configurations of an articulated system composed of 2 variable length links  $a$  and  $b$  where the distance between the base and the end effector need to satisfy spatial constraint  $c$ . (a) Each link has an appropriate length to satisfy constraint  $c$ . (b)  $c$  is too long for the end effector to reach. (c)  $c$  is too short for the end effector to reach.

#### 3.1 Articulated Robot as Hierarchy of Virtual Links

To simplify the illustration, here we consider articulated robots with spatial constraints where the distance between the base and the end effector needs to be in a certain range. However, this representation is general and may be applied to other robotic systems and other spatial constraints.

In an articulated robot, two (or more) consecutive links (called the *children*) form a longer *virtual link* (called the *parent*). For example, the *virtual link* 8, see Figure 3(a), 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 get a single virtual link (14) at the highest level, see Figure 3(b). Note that we can consider an actual link as a virtual link without children.

#### 3.2 Reachable Range of Virtual Links

The *reachable distance* of a 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 whole chain, while the reachable range of a 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  $[l_{min}, l_{max}]$ .

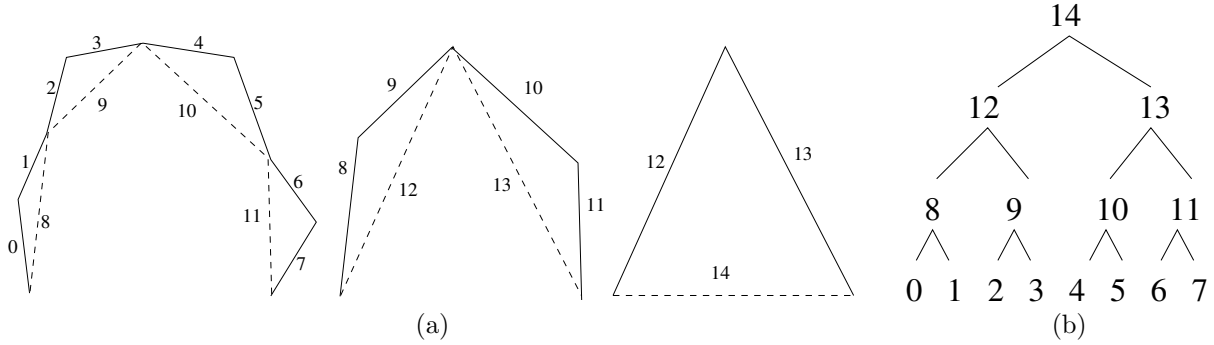


Figure 3: (a) Two consecutive links forms a virtual link (indicated by the dashed links). This grouping repeats to form a hierarchy where the virtual links in one level of the hierarchy become the actual links of the next level in the hierarchy. (b) The tree represents the entire sub-chain hierarchy where the nodes correspond to the virtual links.

Now consider the virtual link  $l$  with 2 children 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. The reachable range of the parent is then  $[l_{min}, l_{max}]$  where

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

and  $l_{max} = a_{max} + b_{max}$ . Note that Equation 1 is general. Given the reachable ranges of any two links in the same triangular sub-chain, it calculates the reachable range of the third one to satisfy the triangle inequality. Thus, for this example, given  $a$  and  $b$ , we can calculate the range of  $l$ . On the other hand, given a specified value of  $l$ , we can find the available range of  $a$  and  $b$  and sample valid lengths for them.

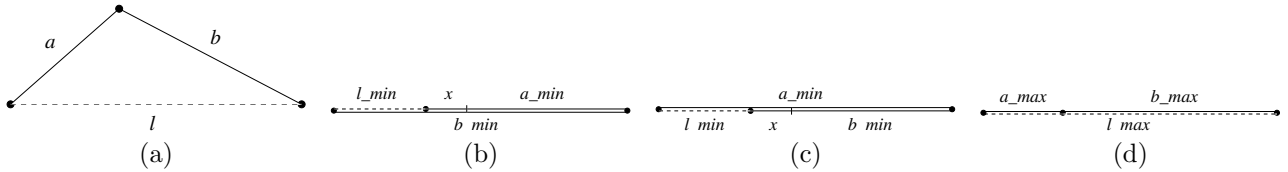


Figure 4: (a) A sub-chain with 2 children  $a$  and  $b$  and its virtual link  $l$ . (b) A configuration where  $l$  is minimum when  $a_{min} < b_{min}$  and  $x = \min(a_{max} - a_{min}, b_{min} - a_{min})$ . (c) A configuration where  $l$  is minimum when  $a_{min} > b_{min}$  and  $x = \min(b_{max}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

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 in an example application in more detail in the following section.

## 4 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 working zone or safe zone.

Consider the robot system in Figure 5. 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 attainable 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

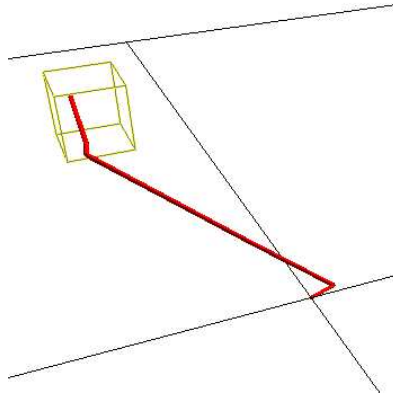


Figure 5: An articulated robot with 3 links must keep its end effector within the specified boundary  $B$  (displayed in wireframe).

much smaller than the original range  $[0, l]$ . We can take advantage of this information by restricting the *available range* of the end effector from  $[l_{min}, l]$  to  $[d_{min}, d_{max}]$  before we sample the rest of the reachable distance tree. In the following, we first define available range, then discuss how to sample using the reachable distance tree, and show how to perform restricted end effector sampling. Finally, we provide experimental results comparing the efficiency of our method to other randomized sampling schemes.

#### 4.1 Available Range of a Sub-Chain

The available range (or attainable range) of a virtual link is a set of distances/lengths which allow it to satisfy the triangle inequality defined by the other links in the sub-chain given the available ranges of the other links. The available range is a subset of the reachable range. Changes in the available range of one link cause changes in the available ranges of the other links in the sub-chain.

Before sampling begins, the available range of each link is equal to its reachable range. This, by definition, satisfies the triangle inequality. As we sample and fix the length of a link in a sub-chain, we need to update the available ranges in the other links in the sub-chain. We can do this using Equation 1. Note that Equation 1 is used in a more general way here:  $a$ ,  $b$  and  $l$  can be any link in the same triangular sub-chain.

#### 4.2 Sampling with the Reachable Distance Tree

Sampling in the reachable distance space involves recursively sampling virtual link lengths from their reachable range and updating the available 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. We describe each step in more detail below.

##### 4.2.1 Recursively sample link lengths

Because we know the reachable range for each sub-chain in the hierarchy, sampling a *spatially constrained* configuration simply becomes sampling distances in the *available* reachable ranges of each sub-chain. Once we fix the length or reachable distance of a parent sub-chain, portions of its children’s reachable ranges may become invalid. We define the remaining valid portions of their reachable range as their *available range*. Note that an available range may never become empty, at the very least its minimum and maximum may become equal. Thus, we sample reachable distances and update available ranges starting at the the root of the hierarchy until all sub-chain reachable distances are sampled or an available range becomes empty. Algorithm 4.1 describes this recursive sampling strategy.

---

**Algorithm 4.1** Sample Lengths

---

*Input.* A sub-chain  $c$ . Let  $c.arr$  be  $c$ 's available reachable range,  $c.left$  and  $c.right$  be  $c$ 's children, and  $c.len$  be the length of  $c$ 's virtual link. Let  $p$  be  $c$ 's parent and  $s$  be  $c$ 's sibling.

- 1: Update  $c.arr$  from  $p.arr$  and  $s.arr$ .
  - 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**
- 

Recall that the hierarchy of sub-chains is a binary tree and that there is one internal node for each virtual link. The sampling algorithm is called once on each virtual link. Because there are  $O(n)$  internal nodes in the binary tree (where  $n$  is the number of actual links in the chain), the sampling algorithm is  $O(n)$ .

#### 4.2.2 Sample link orientations

Each sub-chain and virtual link forms a triangle. In 2D, two different configurations have the same virtual link length: a concave orientation and a convex orientation (see Figure 6(a)). In 3D, a virtual link length can represent many configurations depending on the dihedral angle between its triangle and its parent's triangle (see Figure 6(b)). Thus, we also sample the orientation of the virtual link (i.e., concave/convex for 2D, dihedral angle for 3D).



Figure 6: (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  $\rho$  between the sub-chain's plane and its parent's plane.

#### 4.2.3 Calculate joint angles

Consider the joint angle  $\theta$  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:

$$\theta = \text{acos}\left(\frac{l_a^2 + l_b^2 - l_c^2}{2l_a l_b}\right). \quad (2)$$

### 4.3 Enforcing End Effector Placement

We can easily restrict the end effector distance by setting the available range of the virtual link connecting the base and the end effector (i.e., the root of the reachable distance 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 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 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  $B$ .

## 4.4 Results

Here we compare the performance of our sampling scheme in reachable distance space to uniform random sampling and RRT-style sampling [12] in joint space. For RRT-style sampling, only count the time to sample nodes and check their validity. We 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<sup>1</sup>. The robots contain 3 to 100 links of varying length, however the sum of each robots link lengths is the same. All results in the paper were performed on an 3G Hz PC desktop computer, and all implementations were compiled using gcc4 under linux.

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 provided in Figure 5. 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. Clearly, reachable distance sampling out-performs the other randomized sampling strategies for restricted end effector sampling.

Sampling 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.

Note that RRT performs significantly slower for the 3 link robot than for the other larger robots. The success of RRT in this situation 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. This is not the case for the larger robots. Thus, RRT spends more time on the initial tree samples for the 3 link robot than the other larger robots. As expected, it does see a performance gain over uniform random sampling for the other robots.

## 5 Application: Drawing/Sculpting

An interesting application is robotic drawing and sculpting. Figure 1(d) displays an articulated linkage drawing the letters “RSS” on a canvas. There are more applications on robots where the manipulator need 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.

Again we can take advantage of the reachable distance tree. We propose a two phase approach:

1. With a local planner, 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

<sup>1</sup>Since the environment does not contain any obstacles, the collision detector only checks self-collisions.

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.

2. 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 then rotating the configuration to align with the drawing target for each point along the drawing trajectory. Observe that since the pre-computed path was smooth in reachable-distance space, the current drawing configuration will be continuous to the previous one.

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 left with potentially 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.

Beyond drawing, this algorithm can also be potentially used by other on-line planning applications such as chasing or monitoring. We describe the local planner in detail below.

## 5.1 Local Planning

Given two configurations,  $q_s$  and  $q_g$ , a local planner attempts to find a sequence  $\{q_s, q_1, q_2, \dots, q_g\}$  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 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.

For each virtual link, this local planner interpolates between its length in  $q_s$  and its length in  $q_g$  to get a sequence of lengths. We then must determine the virtual link’s orientation sequence (i.e., concave/convex in 2D and the dihedral angle in 3D) 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  $q_g$ . We determine the validity of the sequence by checking that each virtual link’s length is in its available reachable range and collision-free.

### 5.1.1 Concave/convex flipping for 2D chains

Consider the first and last configurations of sub-chains and virtual links in Figure 6. When the orientation is different, as in this example, we need to find a transformation from one to the other.

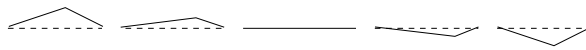


Figure 7: We need to flip the links to transform a convex configuration to a concave one first.

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 reachable range must be large enough to accommodate the summation of its children’s reachable distance (i.e., allow the children to be “flat”). Such a constraint on reachable-distance can be easily handled by our method. There are other options that are more powerful. In our current implementation, we first recalculate the available range for this minimum “flat” constraint. Then we sample an intermediate configuration where the virtual links are “flat” and try connecting it to both the start and goal configurations.

## 5.2 Results

We applied our drawing algorithm to robots with 3, 5, 10, 20, 50, 100 links. Figure 8 shows the planning results of an articulated robotic arm drawing the characters “RSS” on a canvas for the 3 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.

Table 2 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

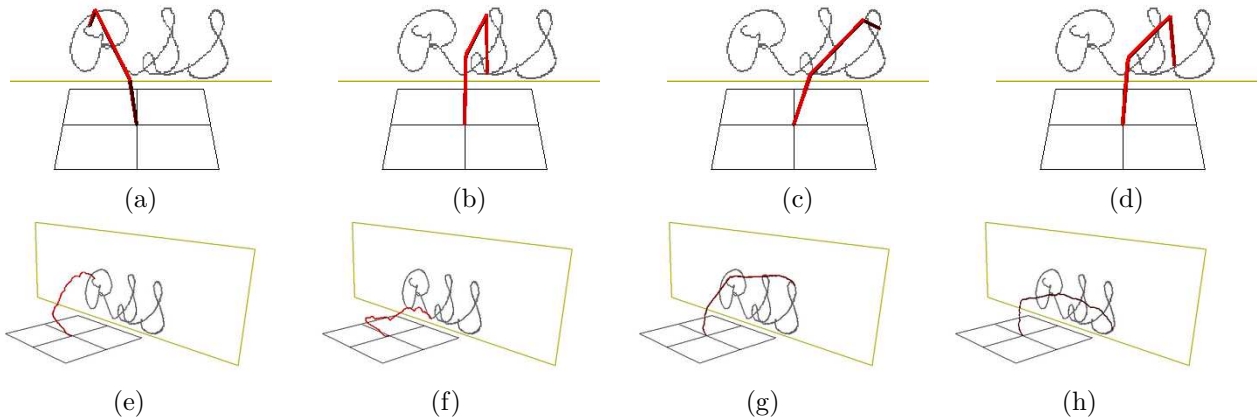


Figure 8: A robotic arm drawing letters “RSS” on the canvas. (a)-(d) show the drawing process of a 3-link robotic arm. (e)-(h) show the drawing process of a larger robot with 100 links.

very small and phase II planning is short enough for on-line applications such as drawing, monitoring, or chasing. Figure 9 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 as the robot’s degrees of freedom increase, thus it is well-suited for on-line applications. Note that other randomized planners like PRMs and 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 50 link robots in 1 hour.

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

Table 2: Averaged running time of different drawing robots in 10 runs.

## 6 Application: Closed Chains

Finally, we can apply reachable distance space to closed chain planning as we did in [citation removed for blind submission]. Here, we model the closed chain as a simple articulated linkage. We enforce the closure constraint (i.e., the restriction that the chain remains closed at all times) by setting the available range of the virtual link that goes from the base to the end effector to  $[0, 0]$ . We then sample and plan as described earlier in Sections 4.2 and 5.1.

### 6.1 Results: Performance Study

Here we demonstrate that our sampling method is fast and efficient in practice. First we study the time required to generate 1000 configurations for closed chains of size 10, 20, 50, 100, 200, 500, 1000, 5000, 10000, and 100000 links. Chain links vary in size ranging from 0.1 to 1.0. Table 3 and Figure 10 compare open chain sampling to closed chain sampling in reachable distance space, ignoring collision detection costs. The results show that sampling configurations that satisfy closure constraints is comparable to sampling open chain configurations that ignore closure constraints entirely. In addition, closed chain sampling scales linearly with the size of the chain, as expected. This demonstrates the advantage of using reachable distances to represent configurations instead of the traditional joint angle representation.

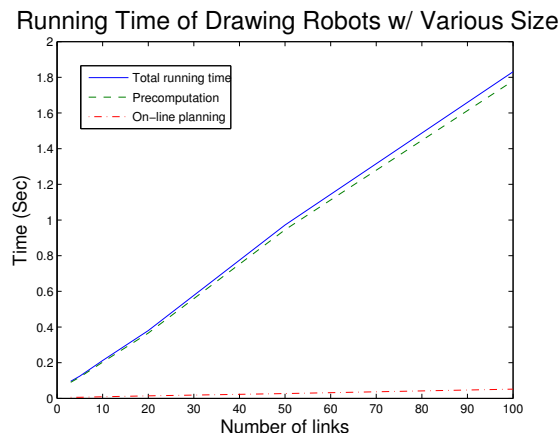


Figure 9: Running time of drawing robot with different number of links. Here we show the total running time, the pre-computation time and the actually on-line planning time.

Number of Links	Open Chain		Closed Chain (s)
	in Joint Space (s)	in RD Space (s)	
10	0.001	0.002	0.002
20	0.003	0.005	0.004
50	0.005	0.016	0.013
100	0.007	0.029	0.028
200	0.016	0.063	0.061
500	0.038	0.165	0.165
1,000	0.076	0.346	0.346
5,000	0.380	3.156	3.162
10,000	0.840	6.751	6.738
100,000	9.007	81.224	81.003

Table 3: Running time to generate 1000 open configurations in joint space and in reachable distance space and 1000 closed configurations in reachable distance space.

## 6.2 Closed Chain Planning

Here we demonstrate the ability of reachable distance planning to find a valid, closed path for a 20 link single loop, fixed base, closed chain. Figure 11 shows the start (a) and goal (d) configurations. We used the greedy RRT [9] planning strategy in conjunction with reachable distance sampling and local planning to find the path in Figure 11(a – d). We were able to find the path in 363.3 seconds with a roadmap containing 394 nodes.

## 7 Conclusion

We presented a new method to plan the motion of spatially constrained problems based on a hierarchical representation of the system as a tree of reachable distances. We then show how this framework can be applied to efficiently sample and plan motions for 3 different types of spatially constrained systems: restricted end effector sampling of an articulated linkage, robotic arm drawing/sculpting, and closed chain systems. We provide a sampling algorithm to sample constraint-satisfying configurations that is linear in the complexity of the robot. We also discuss a local planning method that exploits the reachable distance hierarchy to find paths between neighboring configurations. 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. In the future, we plan to study how our method performs for other spatially constrained systems not discussed here.

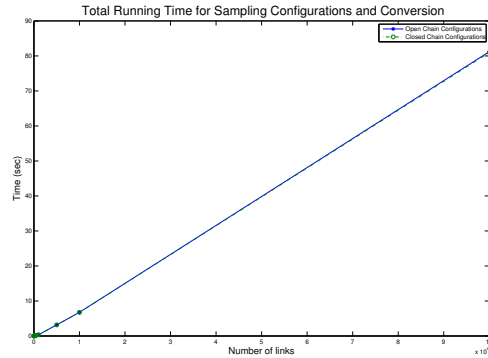


Figure 10: Running time to generate 1000 open (and closed) configurations in the reachable-distance space without considering collision for open (and closed) chains. Note that the running time is linear in the number of links.

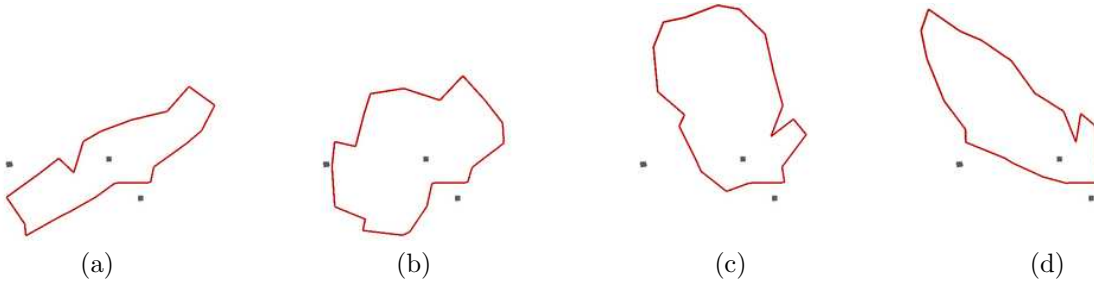


Figure 11: A single-loop, fixed base, closed chain must avoid the 3 obstacles in moving from the start (a) to the goal (d).

## References

- [1] J. Cortes, T. Simeon, 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, 2002.
- [2] M. Garber and M. Lin. Constraint-based motion planning for virtual prototyping. In *Proc. ACM/Siggraph Symp. Solid Modeling*, pages 257–264, 2002.
- [3] L. Han and N. M. Amato. A kinematics-based probabilistic roadmap method for closed chain systems. In *Robotics: New Directions*, pages 233–246, Natick, MA, 2000. A K Peters. Book contains the proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR), Dartmouth, March 2000.
- [4] 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)*, July 2006.
- [5] M. Kallmann, A. Aubel, T. Abaci, and D. Thalmann. Planning collision-free reaching motion for interactive object manipulation and grasping. In *Eurographics*, 2003.
- [6] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, 12(4):566–580, August 1996.
- [7] 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. Robot. Autom. (ICRA)*, pages 546–553, 1996.
- [8] K. Kotay, D. Rus, M. Vona, and C. McGray. The self-reconfiguring robotic molecule: Design and control algorithms. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 375–386, 1998.

- [9] J. J. Kuffner and S. M. LaValle. RRT-Connect: An Efficient Approach to Single-Query Path Planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 995–1001, 2000.
- [10] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.
- [11] 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, 1999.
- [12] S. M. LaValle and J. J. Kuffner. Rapidly-Exploring Random Trees: Progress and Prospects. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages SA45–SA59, 2000.
- [13] J. P. Merlet. Still a long way to go on the road for parallel mechanisms. In *Proc. ASME Int. Mech. Eng. Congress and Exhibition*, 2002.
- [14] R. J. Milgram and J. Trinkle. The geometry of configuration spaces for closed chains in two and three dimensions. *Homology Homotopy Appl.*, 6(1):237–267, 2004.
- [15] A. Nguyen, L. J. Guibas, and M. Yim. Controlled module density helps reconfiguration planning. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2000.
- [16] 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, 2005.
- [17] J. H. Reif. Complexity of the mover’s problem and generalizations. In *Proc. 20th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 421–427, 1979.
- [18] A. Singh, J. Latombe, and D. Brutlag. A motion planning approach to flexible ligand binding. In *7th Int. Conf. on Intelligent Systems for Molecular Biology (ISMB)*, pages 252–261, 1999.
- [19] 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.
- [20] 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, 2004.
- [21] J. H. Yakey, S. M. LaValle, and L. E. Kavraki. Randomized path planning for linkages with closed kinematic chains. *IEEE Transactions on Robotics and Automation*, 17(6):951–958, 2001.
- [22] 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, 2005.