

Planning with Reachable Distances: Fast Enforcement of Closure Constraints

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

Technical Report TR06-008
Parasol Lab.
Department of Computer Science
Texas A&M University

September 18, 2006

Abstract

Motion planning for closed-chain systems is particularly difficult due to additional constraints, so-called *closure constraints*, placed on the system. In fact, the probability of randomly selecting a set of joint angles that satisfy the closure constraints is zero. We overcome this challenge by considering a representation of the chain as a hierarchy of sub-chains, each with its own reachable distance range, and instead of randomly sampling in the joint angle space, we recursively sample in the reachable distance space. This provides two distinct advantages over traditional approaches: (1) joint angles are quickly and easily calculated using basic trigonometry relationships instead of using more expensive inverse kinematics solvers, and (2) configurations are guaranteed to satisfy the closure constraints.

In this paper, we describe this hierarchical chain representation and give a sampling algorithm with complexity linear in the number of links in the chain. Our method can be used to significantly improve sampling-based planners for closed-chain systems by overcoming the difficulty of sampling and satisfying closure constraints. We provide the necessary motion planning primitives (namely sampling and local planning) to implement most sampling-based motion planners. Our experimental results show 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. It is easy to implement and general. It is also extendible to more distance-related constraints besides the ones demonstrated here.

1 Introduction

Closed chain systems are involved in many applications in robotics and beyond, such as the Stewart Platform [18], parallel robots [13], closed molecular chains [17], animation [5], reconfigurable robots [8, 15], and grasping for single and multiple robots [7]. However, motion planning for closed-chain systems is particularly challenging due to additional constraints, called *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 [10].

Instead of randomly sampling in the joint angle space to find closed configurations, we overcome this challenge by precomputing the subspace where closed constraints are satisfied and then directly sampling in this subspace. We represent the chain as a hierarchy of sub-chains by recursively breaking down the problem into smaller subproblems. Each sub-chain in the hierarchy may be partitioned into other, smaller sub-chains forming a closed loop. For any sub-chain, we can compute the attainable distance (reachable distance or length) between its two endpoints recursively. With this information, we simply sample distances in these ranges, and then use simple calculations to compute a closed configuration realizing these distances. Thus, we can directly sample in the “reachable” space, which is very expensive to compute explicitly. While a linkage’s reachable distance has been used in other computations, to the best of our knowledge it has not been used to guide sampling in sampling-based planners. Our hierarchical representation provides a way to explore configurations in this reachable space and can be applied to any articulated system.

We will show this method provides two distinct advantages over traditional approaches:

- joint angles are quickly and easily calculated using basic trigonometry relationships instead of using more expensive inverse kinematics solvers, and
- configurations are guaranteed to satisfy the closure constraints.

In this paper, we formally describe this new chain representation and give a recursive sampling algorithm with complexity linear in the number of links in the chain. This algorithm explores the closure constraint surface by recursively sampling in the feasible reachable distance range for each sub-chain. It can quickly determine whether or not it is possible to satisfy the closure constraints of a sub-chain’s children. We note that our method is not guaranteed to generate collision-free configurations, so the closed-configurations it generates must be checked for collision afterwards.

Our method can be used to significantly improve the performance of sampling-based planners for closed-chain systems such as *Probabilistic Roadmap Methods* (PRMs) [6] and *Rapidly-Exploring Randomized Trees* (RRTs) [11], which have been widely successful in solving a variety of other high degree of freedom (dof) problems. However, the traditional representation using joint angles makes it difficult for these sampling-based methods to sample closed configurations. Several strategies [10, 21, 3, 20, 2, 19] have been developed to optimize sampling-based planners for closed chain systems. While they use heuristics to improve the probability of sampling closed configurations, it is still difficult and expensive to sample closed-configurations for large closed-chain systems. Our method can also be applied to these methods to further improve performance. In this paper, we provide examples of the necessary motion planning primitives (namely sampling and local planning) to implement most of these sampling-based motion planners.

Our method is easy to implement and general — it can be applied to any articulated system. It is also extendible to more distance-related constraints besides the ones here. For example, it can be used to sample configurations with the end effector in specific regions.

Our experimental results show 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.

2 Problem Formulation

This paper focuses on closed chain systems. A closed chain system differs from other systems in that it must also satisfy certain closure constraints. Figure 1 gives examples of different types of closed chain systems. Each system has a different set of closure constraints to satisfy.

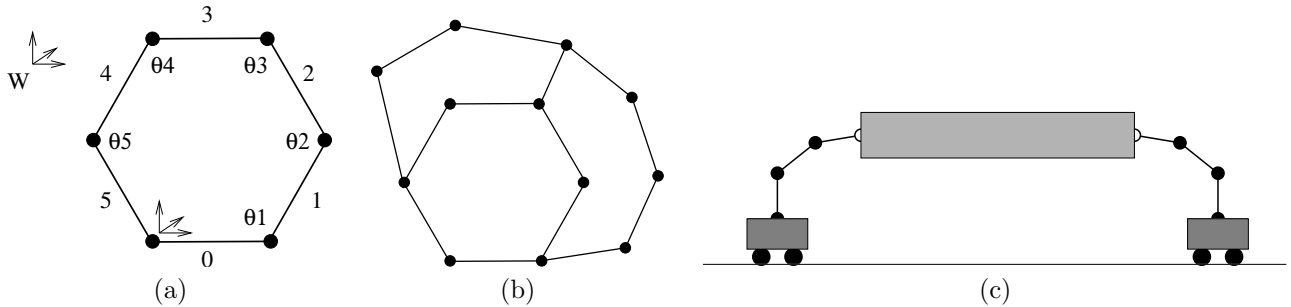


Figure 1: Examples of different closed chain systems. Each must satisfy certain closure constraints. (a) Single loop: loop must remain closed. W is the world coordinate frame. (b) Multiple loops: all loops must remain closed. (c) Multiple robot grasping: all robot end-effectors must remain in contact with the grasped object.

Closed chain systems are traditionally represented by the position and orientation of the base or a base link and one or more angles for each joint (corresponding to the joint’s dof). For example, the single loop in Figure 1(a) is represented by 11 values: 6 for the position and orientation of link l_0 and 5 for the angles $\theta_1 \dots \theta_5$. While this representation sufficiently describes a configuration, it is extremely difficult to sample these parameters randomly such that the closure constraints are also satisfied. This is because this joint angle representation does not also encode the closure constraints — they are handled separately. In fact, it has been shown [10] that the probability of randomly sampling a set of joint angles that satisfy the closure constraints is zero.

3 Related Work

In theory, exact motion planning algorithms [16, 9] can handle systems with closure constraints. However, since exact algorithms are exponential in the dimension of C-space (the set of all possible robot configurations, valid or not), they are generally impractical for large closed-chain systems.

Most general motion planners used today are randomized algorithms such as Probabilistic Roadmap Methods (PRMs) [6] and Rapidly-exploring Randomized Trees (RRTs) [12]. These methods first construct a roadmap (graph or tree) that represents the connectivity of the robot’s free C-space, and then query the roadmap to find a valid path for a given motion planning task. Initially, they were mainly limited to rigid bodies and articulated objects without closure constraints.

In the first approach to handle closure constraints, Kavraki et al.[10], sampled roadmap nodes by first sampling random nodes in C-space and then applying an iterative random gradient descent method. The method minimizes error functions that express the closure constraints. Roadmap edges are computed by executing a randomized traversal of the constraint surface between two nodes. Planar test problems with up to 8 links and two loops were solved in several hours. They extended this work in [21] to PRMs and RRTs. Problems with similar complexity were tested for both planners. The PRM-based planner achieved similar results as [10], and the RRT based planner achieved significantly better results, solving problems in 8–20 minutes that previously took several hours.

Han and Amato use a kinematics-based PRM approach (KBPRM) [3]. They first build a roadmap without any obstacles, considering only the system’s kinematics. Then they populate the environment with this kinematics roadmap, removing portions that are in-collision with environment obstacles. Planar and spatial linkages with 7–9 links were solved under a minute. In [20], Xie and Amato handle high dof closed chain systems by applying Iterative Relaxation of Constraints (IRC) [1] to KBPRM to guide sampling.

Cortes et al. [2] proposed an extension of KBPRM called Random Loop Generator (RLG) to improve the sampling of random closed-configurations. It first estimates a bounding box of the reachable workspace, and then samples parameter in that bounding box. This method is faster than the previous methods and makes the sampling more efficient. However it is still expensive to estimate the reachable workspace of a sub-chain and to use inverse kinematics. Moreover, since their estimation is conservative, they are not guaranteed to find a closed configuration if one exists. They are also unable to report if it is impossible to find a closed configuration.

Trinkle and Milgram proposed a path planning algorithm [19] based on the analysis of the C-space of the closed-chain system [14]. Their method does not consider self-collision but still may be applied as a local planner

by PRM or RRT.

Han et al. [4] proposed a set of geometric parameters for closed-chain systems. Those parameters are proved to be piece-wise convex so 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. They do not discuss their algorithm’s complexity. They do not consider self-collisions while sampling. It is not clear how difficult it would be to consider collisions in their path planning.

Our method is also very fast and has linear complexity of the number of links. It guarantees that a closed configuration will be generated or reports that one cannot be obtained. Moreover, our method is general and easy to implement on sampling-based planners. It is extendible to more distance-related constraints than the ones considered here.

4 Reachable Distance Representation

In this section we describe our hierarchical representation based on reachable distances. The main advantage of this representation over the traditional joint angle representation is that it also encodes the closure constraints. This new representation allows us to easily randomly sample closed configurations.

Intuitively, for any closed-chain system, the length of each link has to be in an appropriate range to satisfy the closure constraints. For example, Figure 2 shows a simple triangular closed-chain with 3 links a , b and c of variable length. To sample a closed configuration, we have to make sure the length of each link is in an “appropriate” (feasible) range. In other words, the length of each link needs to satisfy the triangle inequality: $|c| \geq |a + b|$ and $|c| \leq |a - b|$. To sample a closed configuration, we first can calculate the link’s remaining available range based on the available ranges of the other links. In this way we can eventually sample a valid length for each link to satisfy the closure constraints or report that it is impossible to find a closed configuration. For the triangular case in Figure 2, once we get a valid length for each link, we get a triangular configuration that is guaranteed to be closed. Below we show how we extend this sampling strategy to handle a general closed-chain system. This scheme only ensures that the closure constraints are satisfied. Collision checking must still be performed to determine the configuration’s validity.

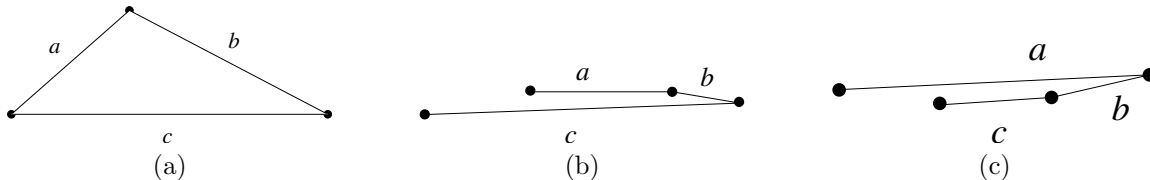


Figure 2: Three configurations of an articulated system composed of 3 links a , b and c where each link has variable length. (a) Each link has an appropriate length to make the configuration closed. (b) c is too long to make a closed configuration. (c) c is too short to make a closed configuration.

4.1 Closed Chain as a Hierarchy of Sub-Chains

For simplicity, here we consider a single loop closed chain. However, this representation is general and may be applied to any closed chain system. The chain (called the *parent*) may be partitioned into several sub-chains (called the *children*) where each sub-chain is a set of consecutive links and the union of the children represent all the links in the parent. For example, the *virtual link* 16, see Figure 3(a), connects the two actual links 0 and 1. In this way, the parent is closed if and only if the virtual links and its children form a closed chain. We recursively partition sub-chains until all the children have only 1 link. This defines a hierarchy of sub-chains and virtual links, as displayed in Figure 3(b).

4.2 Reachable Range of a Sub-Chain

The *reachable distance* of a sub-chain is the distance between the endpoints of its virtual link. Different sub-chain configurations will have different reachable distances. The set of all reachable distances possible for the sub-chain

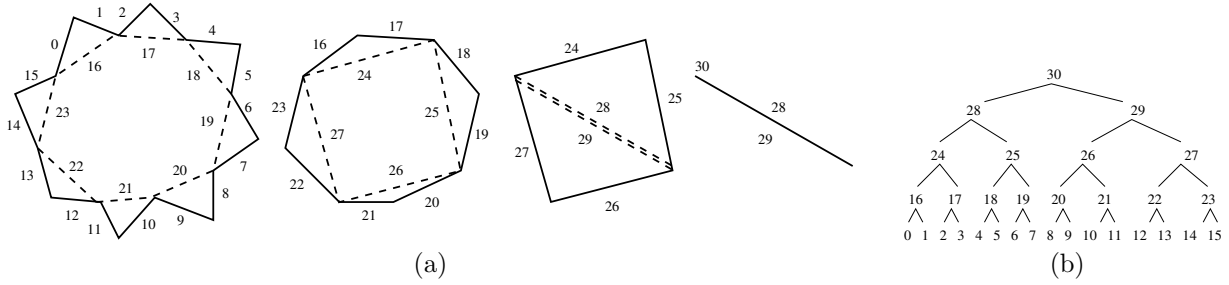


Figure 3: (a) A chain (indicated by the black links in the first image) may be partitioned into a set of sub-chains. Each sub-chain may be represented by a virtual link (indicated by the dashed links). This partitioning repeats to form a hierarchy where the virtual links in one level of the hierarchy 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.

is called its *reachable range*. For example, the reachable range of a single link is simply its length. For a parent link, its reachable range is determined by the reachable ranges of its children.

If we build the sub-chain hierarchy such that each sub-chain has only 0 or 2 children, reachable ranges are computed as follows. Consider a sub-chain 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 $[l_{min}, l_{max}]$.

Now consider the sub-chain with 2 children in Figure 4(a). The sub-chain's virtual link plus its two children 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. We also define the magnitude of the children's reachable ranges as $a_r = a_{max} - a_{min}$ and $b_r = b_{max} - b_{min}$. The reachable range of the parent is then $[l_{min}, l_{max}]$ where

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

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.

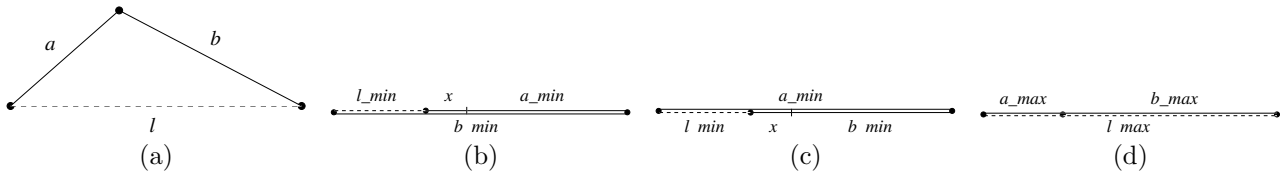


Figure 4: (a) A sub-chain with 2 children a and b and its virtual link l . (b) A configuration where l is minimum when $a_{min} < b_{min}$ and $x = \min(a_{max} - a_{min}, b_{min} - a_{min})$. (c) A configuration where l is minimum when $a_{min} > b_{min}$ and $x = \min(b_{max} - b_{min}, a_{min} - b_{min})$.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

A closed configuration is then a set of distances for each virtual link in the hierarchy such that each distance is within the virtual link's reachable range. We can then compute the joint angles for this configuration using only basic trigonometry relationships instead of more expensive inverse kinematics solvers. We discuss sampling in more detail in Section 5.1.

4.3 Available Range of a Sub-Chain

The available reachable range (available range) of a virtual link is set of distances/lengths which allow it to close with the other links in the same sub-chain (i.e., satisfy the triangle inequality). The available range is a subset of the reachable range that is a function of the available ranges (or lengths) of other links in the same sub-chain loop. Therefore, changes in the available reachable range of the other links will cause changes in the available reachable range of the link.

As shown above, in the beginning, for each sub-chain with 2 children, the reachable ranges of all 3 links can satisfy the triangle inequality and thus the available range is the same as the reachable range. However, once we fix the length of a link, portions of the reachable ranges of other links in the same sub-chain may no longer be valid. When this happens, we need to recalculate their available ranges 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.

5 Application to Sampling-Based Motion Planning

Here we describe two basic primitive operations required for most randomized sampling-based motion planners such as PRMs and RRTs: sampling and local planning (i.e., finding a valid path between two valid samples). We show that these operations are fast and efficient with our reachable distance representation, thereby allowing sample-based motion planners to be directly applied to closed chain problems.

5.1 Sampling

To sample a closed configuration and determine the corresponding joint angles, we first recursively sample the lengths of each virtual link. We then compute the orientation of each sub-chain (e.g., concave or convex for chains in the plane, dihedral angles for non-planar chains). Finally, we use the virtual link lengths and orientations to compute the appropriate joint angles. Note that this sampling only ensures that the configuration is closed — it will still need to be checked for collision to determine validity. We describe each step in more detail below.

5.1.1 Recursively sample link lengths

Because we know the reachable range for each sub-chain in the hierarchy, sampling a *closed* 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 reachable range*. Note that an available reachable range may never become empty, at the very least its minimum and maximum may become equal. Thus, we sample reachable distances and update available reachable ranges starting at the the root of the hierarchy until all sub-chain reachable distances are sampled. Algorithm 5.1 describes this recursive sampling strategy.

Algorithm 5.1 Sample

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 $O(n)$.

5.1.2 Sample dihedral or concave/convex orientation

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 5(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 5(b)). Thus, we also sample the orientation of the virtual link (i.e., concave/convex for 2D, dihedral angle for 3D).

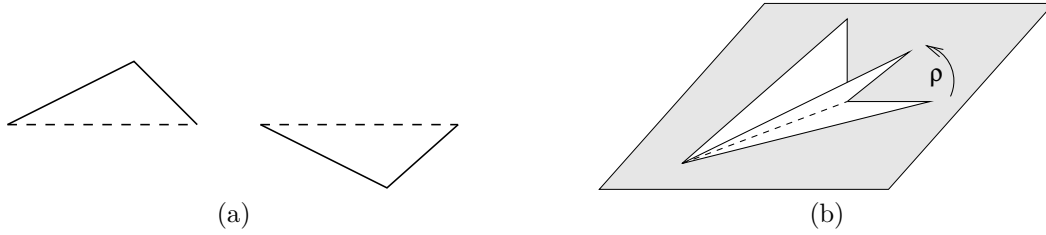


Figure 5: (a) In 2D, the same virtual link represents two configurations: a concave triangle and a convex triangle. (b) In 3D, the same virtual link represents many configurations with different dihedral angles ρ between the sub-chain’s plane and its parent’s plane.

5.1.3 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:

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

5.2 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 configuration is closed (i.e., each virtual link’s length is in its available reachable range) and collision-free.

5.2.1 Concave/convex flipping for 2D chains

Consider the first and last configurations of sub-chains and virtual links in Figure 5. When the orientation is different, as in this example, we need to find a transformation from one to the other.

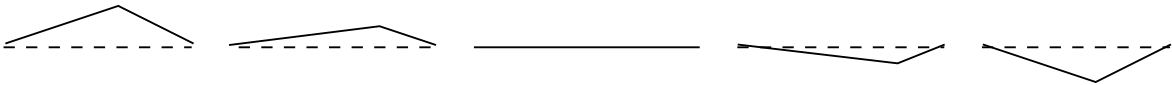


Figure 6: We need to flip the links to transform a convex configuration to a concave one first.

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.

6 Results and Discussion

In this section we demonstrate how this representation performs in practice. We give a performance study in Section 6.1 and demonstrate the method in a complete motion planning framework in Section 6.2. All experiments are performed on an 3G Hz PC desktop computer. All implementations are compiled using gcc4 under linux.

6.1 Performance Study

Here we study and compare the performance of our sampling algorithm both with and without collision detection.

In the first set of experiments, we ignore collision detection. Here we measure the running time to generate 1000 configurations in 10 different environments for chains of size 10, 20, 50, 100, 200, 500, 1000, 5000, 10000, and 100000 links. In each environment, the articulated system is composed of different sized links ranging from 0.1 to 1.0. We compare the performance of generating both open and closed configurations using reachable-distance. As shown in Table 1 and Figure 7, the running time is composed of two parts: the time to sample the configuration using reachable distances and the time to convert the representation into traditional joint angles. Note that here the running time for conversion is much more than the time needed for actual sampling. However, the performance of both scales well even for large numbers of links.

The results show that sampling configurations that satisfy closure constraints is comparable to sampling open chain configurations that ignore closure constraints entirely. This demonstrates the advantage of using reachable distances to represent configurations instead of traditional joint angle representation.

Number of Links	10	20	50	100	200	500	1,000	5,000	10,000	100,000
Sample Open Chain (sec)	0.001	0.003	0.005	0.007	0.016	0.038	0.076	0.380	0.840	9.007
Sample Open Chain and Convert (sec)	0.002	0.005	0.016	0.029	0.063	0.165	0.346	3.156	6.751	81.224
Sample Closed Chain (sec)	0.001	0.002	0.004	0.008	0.015	0.037	0.075	0.380	0.840	9.070
Sample Closed Chain and Convert (sec)	0.002	0.004	0.013	0.028	0.061	0.165	0.346	3.162	6.738	81.003

Table 1: Running time to generate 1000 open (and closed) configurations in the reachable-distance space without collision detection for open chains (and closed chains). We show both the time to purely sample a set of available reachable distances from the reachable distance space and the time including the conversion into the traditional joint angle representation.

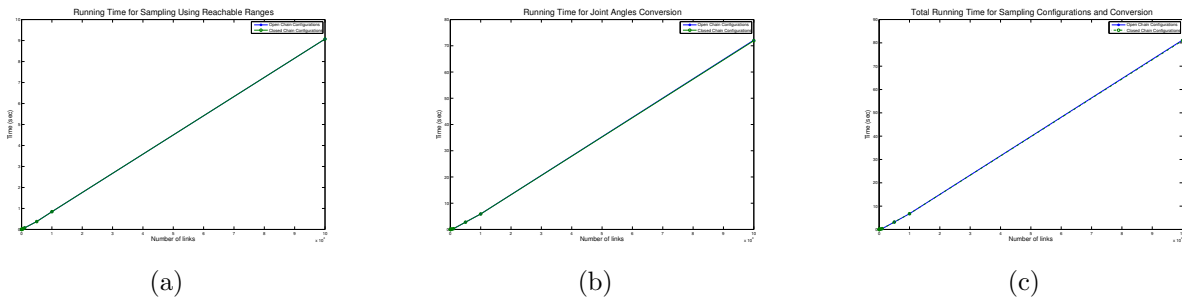


Figure 7: Running time to generate 1000 open (and closed) configurations in the reachable-distance space without considering collision for open (and closed) chains for (a) purely sampling open/closed configurations using reachable distance, (b) converting open/closed configurations from reachable distances to joint angles, and (c) both sampling and converting open/closed configurations. Note that the running time is linear in the number of links.

In the second set of experiments, we measure the performance when self-collision is considered. Again we measure the time to sample and convert 1000 self-collision-free configurations. We studied 7 different environments with chains of size 10, 20, 50, 100, 200, 500, and 1000. In each environment, we measured the running time to

generate open-chain configurations in reachable-distance space and generate closed-chain configurations reachable distance. Table 2 and Figure 8 show the performance results.

Note that when we consider the collision detection, the performance of sampling an open-chain in reachable-distance space is comparable to the performance of sampling a closed-chain in reachable-distance space. This strongly shows that by directly sampling in the reachable space, our method can sample closed configurations almost as efficiently as sampling open-chain configurations. This means that our method makes it practical to solve motion planning problems for large articulated systems with distance-related constraints.

Sampling Method	Collision Detection	10	20	50	100	200	500	1,000
Open Chain	N	0.002	0.005	0.016	0.029	0.063	0.165	0.346
	Y	0.006	0.018	0.199	1.464	10.416	125.076	627.800
Closed Chain	N	0.002	0.004	0.013	0.028	0.061	0.165	0.346
	Y	0.005	0.016	0.217	1.375	10.436	124.566	546.200

Table 2: Running times of different sampling methods both with and without collision detection for chains with 10 links up to 1000 links.

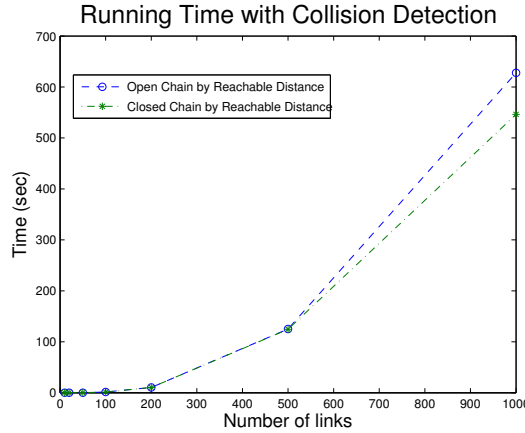


Figure 8: Running time of open configuration sampling using reachable distances and closed configuration sampling using reachable distances all with collision detection as a function of the number of links. The performance for both methods is comparable since they are each dominated by collision detection.

6.2 Application in Motion Planning Frameworks

In this section, we show how we apply the sampling and local planning primitives and use them to solve closed-chain problems.

6.2.1 Multiple robot grasping

Here, two articulated robotic arms with mobile bases grasp an object and work together to move it from one end of the environment to the other (see Figure 9(a)). Here each arm is composed of 3 links. Considering the grasping object and the robot’s base, there are 8 dof in total. The robot must maneuver the object underneath the object in the ceiling to solve the query.

We use an incremental PRM method to construct the roadmap and to solve the query. It begins with a small roadmap and incrementally generates more nodes until the query is solved. Our method required 40 nodes to solve the query. The total running time was 58.4 seconds to build the map and find a valid path with 3100 intermediate configurations (see Figure 9(b)). An example configuration where the robot compresses itself underneath the obstacle is shown (c).

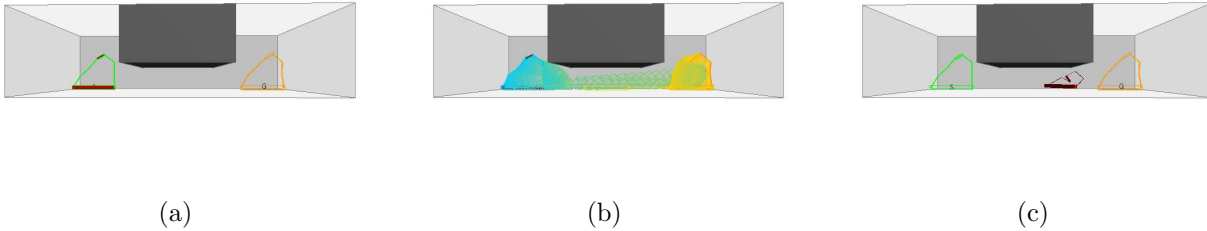


Figure 9: (a) A grasping experiment where 2 arms collaborate to grasp an object and transport it from one end of the environment to the other. (b) A path’s swept volume. (c) Intermediate configuration along the path.

6.2.2 Single-loop closed chain

A single-loop chain with 10 links of variable length must pass through a series of narrow passages to traverse the entire environment (see Figure 10). Again, we use incremental PRM map generation to build a roadmap until it solves the query. Our method successfully found a valid path with only 20 nodes in 96.2 seconds of computation time. Figure 10(b) shows the swept volume of the path with 1680 intermediate configurations.

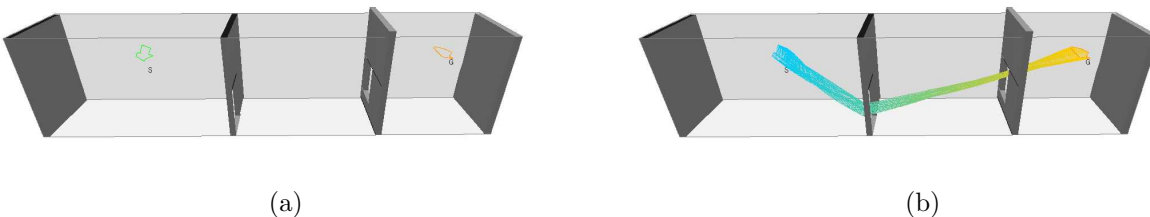


Figure 10: (a) Single-loop chain that must pass through a series of narrow passages to traverse the entire environment. (b) The swept volume of the solution path.

7 Conclusion

We proposed a new method to plan the motion of closed-chain problems based on a hierarchical representation of the chain. It can be used to quickly generate closed-configurations or determine that it is impossible to satisfy the closure constraints. We described this new representation and gave a sampling algorithm to generate closed configurations with linear complexity in the number of links. It can be used to significantly improve sampling-based planners for closed-chain systems by overcoming the difficulty of sampling closed configurations. We provide the necessary motion planning primitives (namely sampling and local planning) to implement sampling-based motion planners. Our experimental results show that our method is fast and efficient in practice, making the cost of generating closed and open configurations comparable.

References

- [1] O. B. Bayazit. *Solving Motion Planning Problems By Iterative Relaxation Of Constraints*. Ph.D. dissertation, Dept. of Computer Science, Texas A&M University, May 2003.
- [2] 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.
- [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.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.
- [10] 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.
- [11] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 473–479, 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] J. H. Reif. Complexity of the mover’s problem and generalizations. In *Proc. 20th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 421–427, 1979.
- [17] 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.
- [18] D. Stewart. A platform with six degrees of freedom. *Proc. of the Institute of Mechanical Engineering*, 180(part I(5)):171–186, 1954.
- [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.