Manipulation Planning with Directed Reachable Volumes*

Troy McMahon[1], Read Sandström[2], Shawna Thomas[2], and Nancy M. Amato[2]

***Abstract*— Motion planning for manipulators with rotational joints is challenging because the actuation range for each link is constrained by the placement and orientation of other links. Thus, finding paths that avoid self-collision is non-trivial. However, rotational joints are often used in industrial robots.**

We develop a reparameterization of the planning problem called *directed reachable volumes* that provides an explicit representation of the workspace regions that the joints and end effectors can reach given the placement *and orientation* of other links. This formulation, while similar in spirit to prior reachable volume work, does not rely on the same restrictive assumptions that preclude prior work from handling rotational joints. We provide primitive planning operations that can be used in the context of state-of-the-art motion planning methods. We present experimental validation of directed reachable volumes by demonstrating a simulated pick-and-place scenario using realistic robots with rotational joints.

I. Introduction

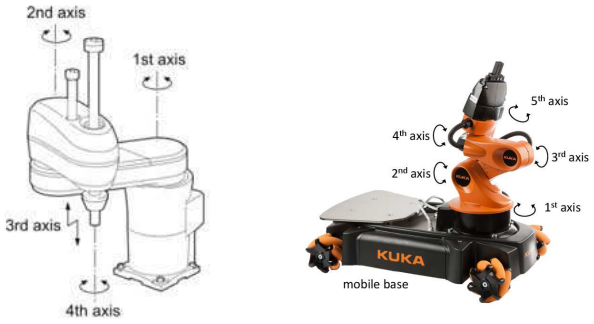
Motion planning consists of a mobile object (robot) that must find valid (e.g., collision-free) paths through an environment. This problem has applications in mobile robots, grasping and manipulating [1], [2], computational biology [3] and animation [4]. Sampling-based methods (e.g., PRM [5] and RRT [5]) are among the most widely used solutions and are capable of solving a wide variety of problems.

While generally successful, these methods do not perform as well in constrained problems such as those frequently observed in manipulation planning. The challenge comes in planning for a system that must simultaneously avoid self-collision, maintain joint limit constraints, and reach target placements for certain components such as the end-effector. The probability of randomly sampling a constraint-satisfying configuration can be very small and in some cases approaches zero [6]. In addition, many manipulators employ rotational joints because they weigh less and are more cost effective than spherical joints. For example, industrial SCARA robots [7] have 3 rotational joints mixed with a single prismatic joint, and the KUKA YouBot [8] has 5 rotational joints on a mobile base with different orientation axes (see Figure 1). Rotational joints are also found in many research platforms such as the PR2 [9] and Fetch [10].

*This research supported in part by NSF awards CNS-0551685, CCF 0702765, CCF-0833199, CCF-1439145, CCF-1423111, CCF-0830753, IIS-0916053, IIS-0917266, EFRI-1240483, RI-1217991, by NIH NCI R25 CA090301-11, and by DOE awards DE-AC02-06CH11357, DE-NA0002376, B575363.

[1]Troy McMahon is a Postdoctoral Researcher in Computer Science and Engineering, University of Michigan, Ann Arbor, MI 48109-2121, USA tamcm@umich.edu. This research was conducted while he was a Ph.D. candidate at Texas A&M University.

[2]Sandström, Thomas, and Amato are with the Parasol Laboratory, Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843-3112, USA {readamus,sthomas,amato}@cse.tamu.edu



(a) SCARA [7] (b) KUKA Youbot [8] (labels added)

Fig. 1. Examples of robots that employ revolute joints. (a) This SCARA (selective compliance assembly robot arm) has revolute joints at its first, second, and fourth axes. (b) A KUKA YouBot with a manipulator arm composed of five revolute joints.

Planning for rotational joints is in many ways more challenging than other joint types. Their range of motion is usually more constrained, limiting the amount the robot can deform or extend. In some cases this makes it difficult to fold the robot's appendages, limiting the robot's flexibility and ability to navigate narrow passages or tight turns. Most work is not specifically tailored to address these issues.

In this work, we develop *directed reachable volumes* (DRVs) which reparameterizes traditional configuration space ($\mathcal{C}_{space}$) into a new planning space called DRV-space. DRV-space encodes the oriented volume of workspace that individual joints can access in the context of how other joints are placed. DRVs extend the concept of reachable volumes (RVs) [11], [12], [13] to handle rotational joints in addition to spherical and prismatic joints. Because DRVs incorporate orientation as well as position of the reachable region, they are able to model constraints on both the position and orientation of a robot's joints and end-effectors instead of being limited to positional constraints only.

Rotational joints break many of the underlying assumptions in the previous RV work. In particular, they rely on Minkowski sums for their computation which do not apply to joints with orientation requirements. Thus, we define a geometric operation called a *rotational sum* and show how to use these to compute DRVs. We also provide DRV versions of key sampling-based motion planning primitives, namely sampling and local planning, that can be used in existing sampling-based strategies such as PRMs and RRTs.

We show that DRVs can be applied to a variety of real world grasping and manipulation tasks for robots containing rotational joints. We demonstrate that DRVs can solve problems more efficiently than existing methods and are often

capable of solving problems that other methods cannot.

II. Related Work

Sampling-based methods are the current state-of-the-art in motion planning. There are two main classes: graph-based methods (e.g., Probabilistic Roadmaps (PRMs) [14]) and tree-based methods (e.g., Rapidly-Exploring Random Trees (RRTs) [15]). They have been successfully applied to many different application domains. However, planning for manipulators poses additional challenges because each link of the robot is effectively constrained by the position and orientation of the other links.

A. Methods for Manipulators

Manipulation problems often involve one or more spatial constraints. Spatial constraints on the links or joints of a robot place a special kind of restriction on the free $\mathcal{C}_{space}$, which is often described as a sub-manifold. This is challenging for sampling-based planners because valid paths are restricted to this manifold. Due to the difficulty in producing valid samples that meet this criterion [6], several works investigate specialized methods for this class of problems.

The most common technique employed by such methods is a *gradient descent*, whereby randomly generated configurations are pushed onto the constraint manifold. Examples of this technique include CCD [16], ATACE [17], and CHOMP [18]. CCD (cyclic coordinate descent) first places the end-effector in a valid configuration and then iteratively pushes the remaining links until the entire configuration is valid. ATACE (alternate task-space and configuration-space exploration) uses a randomized gradient descent to find constraint-satisfying paths for the end-effector first, and then subsequently pushes the remaining DOFs to the constraint manifold. CHOMP (covariant hamiltonian optimization for motion planning) uses gradient descent to both satisfy hard constraints and optimize soft constraints.

Another method is to integrate a motion controller that incorporates the problem's constraints, as in [19]. In this method, the constraints are described as a kinematic map from the $\mathcal{C}_{space}$ to a task space. A feedback controller based on this map is then used to steer configurations through the constraint-satisfying sub-manifolds. This strategy supports virtually any robot/task combination for which an appropriate kinematic map can be implemented, although creating the maps is not trivial and each pairing requires its own map.

Some methods offload costly robot-specific computations to a preprocessing step. This is useful to avoid self-collision checks and inverse kinematics computatations when closing loops. For example, kinematics-based PRM (KBPRM) precomputes a local roadmap that uses inverse kinematics to close loops [20], [21]. To plan in a given environment, $\mathcal{C}_{space}$is populated with this roadmap with additional edges between copies that only require rigid body transformations. This idea is also used to build a tiling map that precomputes self-collision check results [22]. This tiling map is then used to construct the $\mathcal{C}_{space}$ roadmap where self-collisions no longer need to be checked.

B. Reachable Volumes

Reachable volumes (RVs) is a method for constrained systems based on Minkowski sums rather than gradient descent. Instead of pushing a configuration to a constraint-satisfying manifold, it directly samples it by computing the region of space that each joint (and end-effector) can occupy while satisfying a problem's constraints [11]. We have applied RVs to both PRM [12] and RRT-style planning [13] for a variety of unconstrained and constrained systems with as many as 262 degrees of freedom.

While successful, RVs are limited in that they cannot be applied to rotational joints. The region of space a rotational joint can occupy depends its orientation as well as its position, whereas other joint types (i.e., spherical, prismatic) depend only on the position. RV computations are based on the observation that if a linkage L_1 can reach a point P_1 and a second linkage L_2 can reach a point P_2, then joining the linkages end to end will form a linkage that can reach the point P_1 + P_2 [11], [12], [13]. Consider the planar two-link linkage in Figure 2. Its reachable volume (Figure 2(a)) is based on the observation that if the second link can reach a point in black, then it can also reach all other points that are the same distance away along the gray circle (Figure 2(b)).

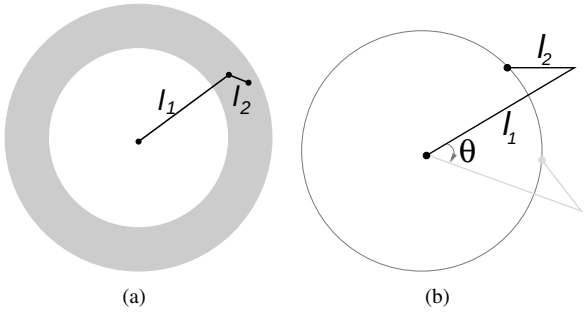


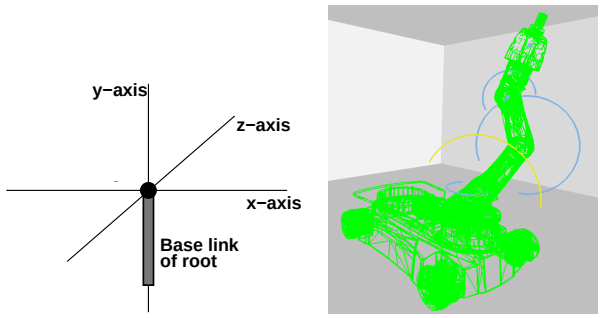
Fig. 2. (a) The reachable volume (gray) of a planar 2 link robot, l_1 and l_2 (black). l_1 rotates about the point in the center while l_2 rotates about the endpoint of l_1. (b) If the end effector (black) can reach a point, then it can reach all other points that are the same distance from the base (gray circle).

RVs leverage this observation to prove that the reachable volume of a chain is equivalent to the Minkowski sums of the reachable volumes of the links in that chain [11], [23]. Thus the reachable volume may be computed from a series of Minkowski sums. This observation does not hold if there is a dependence on the orientation of the links as would occur if they are connected by a non-planar rotational joint. Thus, Minkowski sums cannot be employed when there are orientation dependencies as Minkowski sums are orientation unaware. This precludes RVs from being applied to a large set of industrial robots that contain any rotational joints, the most popular joint manufactured.

III. Directed Reachable Volumes

The *directed reachable volume* (DRV) of a particular joint is the region of space it can reach given the context of the positions *and orientations* of other joints. These positions and orientations may either be single values (i.e., has already been sampled) or ranges (i.e., has not been sampled yet).

We define the directed reachable volume space (DRV-space) of a robot to be a space in which the root of the robot is located at the origin and the coordinate system coincides with the root's Denavit-Hartenberg (DH) [24] coordinates. The origin of DRV-space is located at a specified root joint, and the y-axis of DRV-space is the axis of rotation of the root joint (see Figure 3(a)). The key difference here from RV-space is the addition of the root's y-axis orientation.



(a) (b)

Fig. 3. (a) DRV-space definition. The root is positioned at the origin and oriented along the y-axis. (The x-axis and y-axis lie in the plane of the page, and the z-axis is perpendicular to the page.) (b) The DRVs (blue and yellow regions) for the Kuka Youbot's joints.

With this DRV-space notation, we can precisely define a joint's DRV to be the subset of DRV-space it can occupy given the position and orientation of the root joint. Thus the DRV of joint j is the set of points $p \in$ DRV-space such that there exists at least one configuration in which j is located at p. Figure 3(b) shows the DRVs of the Kuka Youbot's joints. Note that joints are not co-planar and have different limits.

DRVs and DRV-space are structured in a similar way to RVs and RV-space but with a few key differences:

- RVs are computed via Minkowski sums based on the observation that if a linkage L_1 can reach a point P_1 and a linkage L_2 can reach a point P_2, then joining the linkages end to end will be able to reach the point $P_1 + P_2$. However, for rotational and non-planar articulated joints, this is decidedly not true. Instead, computations must also consider orientation when determining the new set of points a joint can reach. Thus, we develop rotational sums that consider orientation to replace Minkowski sums in the underlying computations.
- DRVs can be applied to robots with rotational joints and non-planar articulated joints as well as spherical, planar and prismatic joints (and combinations thereof) while RVs cannot. Thus, RVs cannot be applied to many popular industrial robots.
- DRVs can enforce constraints on the position *and orientation* of joints and end effectors, whereas RVs can only handle positional constraints. Orientation constraints are useful when specifying motions for the end effector to approach an object from a particular direction, either to slide it off of a shelf or to achieve a certain grasp.

A. Rotational Sum

Rotational sums are key to considering orientation in DRV computations. We define the rotational sum about an axis a, $RotSum_a(S)$, to be the sum of all possible rotations of the set of points S about a:

$$RotSum_a(S) = \sum_{\varphi=0}^{2\pi} \text{rotate } S \text{ about axis } a \text{ by } \varphi.$$

We define the rotational sum about a point p, $RotSum_p(S)$, to be the sum of all possible rotations of the set of points S about p. This is equivalent to the set of spherical shells centered at p that intersect one or more points in S:

$$RotSum_p(S) = \sum_{s \in S} \{q \in \mathbb{R}^3 | \delta(q,p) = \delta(s,p)\},$$

where δ is a distance function.

Rotational sums will be used to compose directed reachable volumes in the same manner that Minkowski sums were used to compose reachable volume in [11]. The rotational sum is similar to the Minkowski sum except that it is based on rotations instead of translations. Recall that the Minkowski sum of two sets is computed by adding all possible pairs of vectors between the two sets. Thus it applies all possible translational offsets of one set to the other set. Rotational sums instead apply rotational offsets from one set to another.

B. Computing Directed Reachable Volumes

The DRV of a joint can be computed inductively for serial robots by the following. We define $DRV_{j,j'}$ to be the directed reachable volume of j in the DRV-space rooted at j'. Similarly, $DRV_{j,root}$ is the directed reachable volume of j in the global DRV-space of the robot. We also define the minus operator to denote a joint's parent on the path from the root to j (see Figure 4). Note that even complex joint structures may be partitioned into serial pieces that can be handled this way.

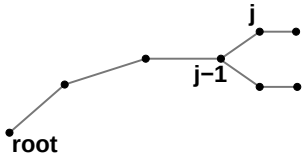


Fig. 4. Joint $j-1$ is the parent of joint j along the path to the root.

We compute $DRV_{j,root}$ by initializing $DRV_{j,j}$ to the origin and iteratively computing $DRV_{j,j-1}$ (corresponding to the DRV-space rooted at j's parent) until we obtain $DRV_{j,root}$. The procedure for obtaining $DRV_{j,j'-1}$ from $DRV_{j,j'}$ depends on what type of joint $j'-1$ is:

- $j'-1$ **is a rotational joint** — We transform $DRV_{j,j'}$ into the frame of $j'-1$ by shifting $DRV_{j,j'}$ by the length of the link connecting j' to $j'-1$ and then rotating the coordinate frame by the offset angle of j'. We then apply a rotational sum about the y-axis to obtain $DRV_{j,j'-1}$. Thus,

$$DRV_{j,j'-1} = RotSum_y(\text{rotate}(DRV_{j,j'} + l_{j',j'-1}, \theta_{j'-1})),$$

where $l_{j',j'-1}$ is the length of the link connecting j' to $j'-1$ and $\theta_{j'-1}$ is the offset angle of joint $j'-1$.

- $j'-1$ **is a planar articulated joint** — $DRV_{j',j'-1}$ is obtained by shifting $DRV_{j,j'}$ by the length of the link connecting j' to $j'-1$ and then computing the rotational sum about the joint's axis of rotation:

$$DRV_{j',j'-1} = RotSum_{\alpha}(DRV_{j,j'} + l_{j',j'-1}),$$

where $l_{j'-1,j'}$ is the length of the link connecting j' to $j'-1$ and α is the axis perpendicular to the plane of rotation of $j'-1$.

- $j'-1$ **is a spherical joint** — $DRV_{j',j'-1}$ is obtained by shifting $DRV_{j,j'}$ by the length of the link connecting j' to $j'-1$ and then computing the rotational sum about the origin O:

$$DRV_{j',j'-1} = RotSum_{O}(DRV_{j,j'} + l_{j',j'-1}),$$

which equates to

$$DRV_{j',j'-1} = \{(x,y,z) | \exists (x',y',z) \in DRV_{j,j'}$$

such that

$$\sqrt{x^2+y^2+z^2} = \sqrt{x'^2+y'^2+z'^2} + l_{j',j'-1}\}.$$

- $j'-1$ **is a prismatic joint** — For a joint that can extend from d_{min} to d_{max}, $DRV_{j',j'-1}$ is the line segment $L = \{(1-t)d_{min} + t(d_{max}) | t \in [0,1]\}$. $DRV_{j,j-1'}$ is obtained by computing the Minkowski sum of $DRV_{j,j'}$ and L and then rotating the coordinate frame by the offset angle of j'.

IV. Sampling-Based Motion Planning with Directed Reachable Volumes

Sampling-based motion planning methods rely on primitive operations including sampling, local planning, and, in the case of RRTs, expansion from an existing sample. We present DRV versions of these primitives so existing sampling-based motion planners can be used to solve problems where the robot includes combinations of planar, spherical, prismatic and rotational joints.

A. Directed Reachable Volume Sampling

To generate DRV samples, we first compute the DRVs of all robot joints. Algorithm 1 outlines this process. These DRVs only need to be computed once and can be reused to generate as many samples as needed.

Algorithm 1 Preprocessing Step

Input: A robot $R = (J, L)$, $root \in J$
Output: $DRV_{j',j}$s needed for sampling

```
1: J_visited = {root}
2: for all j ∈ (J \ root) do
3:     for all j' ∈ J_visited do
4:         DRVS_j[j'] = DRV_{j',j}
5:     J_visited ← j
6: return DRVS
```

At run time, samples are generated by sampling DOF values for at least one joint and iteratively placing adjacent joints in their DRVs given the positions and orientations of the joints that have already been placed (Algorithm 2). For each sampled joint j', we translate and rotate $DRV_{j,j'}$ to match the sampled position of joint j'. The intersection of the translated and rotated $DRV_{j,j'}$s is the DRV of j given the positions and orientations of the joints that have already been placed. We place j randomly in this intersection. Placing all joints results in a DRV-space sample that can be converted into a $\mathcal{C}_{space}$ sample in $O(|J|)$ time (follows similar proof in [11]).

Algorithm 2 Basic DRV Sampling

Input: A robot $R = (J, L)$, $root \in J$
Output: A DRV sample S

```
1: S[root] = (0, 0, 0)
2: for all j ∈ (J \ root) do
3:     for all DRV_j as j' → DRV_{j,j'} do
4:         DRV_j = DRV_j ∩ rotate(DRV_{j,j'} + S[j'], angle_{j'})
5:     S[j] = random point from DRV_j
6: return S
```

This method is analogous to the RV sampling method presented in [11]. The principle differences are that sampling is done in DRV-space instead of RV-space and the $DRV_{j,j'}$s must be rotated to align with the rotation axis of j' which is not necessary in RV-space due to RV symmetries.

Optimized Sampling. The general method presented in Algorithm 2 requires one to compute the DRV between the joint being sampled and every previously sampled neighbor (see line 4). Instead of selecting joints to sample at random, we can select joints whose neighbors have already been sampled. This reduces the depth of the DRV computation sequence required to place the joint. Thus, the DRV of a joint j given the position and orientations of all previously sampled joints only depends on $DRV_{j,root}$ and $DRV_{j,j' \in Neighbors(j)}$ (assuming any cycles in the robot have been broken), see Algorithm 3.

Algorithm 3 Optimized DRV Sampling

Input: A robot $R = (J, L)$, $root \in J$, an arbitrary starting joint s
Output: A DRV sample S

```
1: S[root] = (0, 0, 0)
2: J_sampled = {root}
3: queue.push(s)
4: while j = queue.pop() do
5:     DRV_j = DRV_{j,root}
6:     for all j' ∈ neighbors(j) \ J_sampled do
7:         DRV_j = DRV_j ∩ DRV_{j,j'}
8:         if j' ∉ queue then
9:             queue.push(j')
10:    S[j] = random point from DRV_j
11:    J_sampled ← j
12: return S
```

This ordering is advantageous because $DRV_{j,j'}$ is always the DRV of a single link (see line 7), which can be computed in constant time as follows:

- $j_{neighbor}$ **is an end effector/spherical joint** — $DRV_{j,j_{neighbor}}$ is a shell that is centered at $S[j_{neighbor}]$ with radius equal to the link connecting $j_{neighbor}$ to j.
- $j_{neighbor}$ **is a planar articulated joint** — $DRV_{j,j_{neighbor}}$ is a circle in the plane of motion of $j_{neighbor}$. This circle is centered at $S[j_{neighbor}]$ with radius equal to the link connecting $j_{neighbor}$ to j.
- $j_{neighbor}$ **is a prismatic joint** — $DRV_{j,j_{neighbor}}$ is the line segment defined by the position and orientation of $j_{neighbor}$, the minimum and maximum extension of $j_{neighbor}$, and the position and orientation of $S[j_{neighbor}]$.
- $j_{neighbor}$ **is a rotational joint** — $DRV_{j,j_{neighbor}}$ is the circle of rotation of the joint, which is defined by the position and orientation of $j_{neighbor}$, the offset angle of $j_{neighbor}$, and the position of $S[j_{neighbor}]$ (see Figure 5).

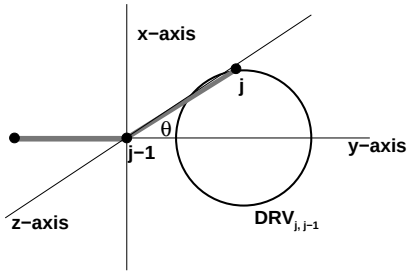


Fig. 5. $DRV_{j,j-1}$ when $j-1$ is a rotational joint.

Non-serial Robots. We generate samples for non-serial robots such as closed chains and graspers by decomposing them into chains and sampling the chains independently (using Algorithm 3). We then merge the sampled chains to form a DRV sample. RVs use a similar technique [11], however with DRVs we must ensure that the orientation of the adjoining joints matches up when merging samples.

B. Directed Reachable Volume Local Planning

For unconstrained problems, DRV samples can be connected with linear interpolation as in standard $\mathcal{C}_{space}$. However, when local planning in DRV-space for problems with constraints (such as a closed chain), the intermediate samples cannot be simply linearly interpolated as this would invariably break constraints. Thus, we present a DRV method for stepping one configuration towards another through DRV-space while maintaining constraint satisfaction. We then use this stepping function for two sampling-based motion planning primitives: local planning and RRT extension.

Stepping. To step a DRV configuration towards another, we iteratively select a joint j, move it toward the direction of that joint in the target configuration, and reposition all of j's children to be in their DRVs. Algorithm 4 gives the overall stepping procedure and Algorithm 5 details how to reposition a joint.

Algorithm 4 DRV Stepping

Input: A DRV configuration q, a joint j, a target position p_{target}, a stepping parameter δ, and a root joint $root$.
Output: A DRV configuration q_{new} in which the joint j has been perturbed by δ in the direction of p_{target}

1: Let p_{init} be the position of j in q
2: $p_{new} = p_{init} + \delta * (p_{target} - p)$
3: **if** $p_{new} \in DRV_{j,j-1}$ **then**
4: $\quad q_{new} = q$
5: $\quad$ Set the position of j to be p_{new} in q_{new}
6: $\quad$ **for all** j' such that $j' - 1 = j$ **do**
7: $\quad\quad$ Reposition(q_{new}, j')
8: $\quad$ **return** q_{new}
9: **else**
10: $\quad$ **return** $\emptyset$

Algorithm 5 Reposition

Input: A configuration q and a joint j
Output: q such that $j' \in j \cap$ descendants(j) in their DRVs

1: **if** $j \in DRV_{j,j-1}$ AND ori(j) = ori(link($j-1, j$)) **then**
2: $\quad$ **return** q
3: **else**
4: $\quad$ Adjust position of j in q to be within $DRV_{j,j-1}$
5: $\quad$ **for all** j' such that $j' - 1 = j$ **do**
6: $\quad\quad$ Reposition(q, j')
7: $\quad$ **return** q

DRV stepping differs from RV stepping in [13] in three ways. Firstly, the DRV of a repositioned joint must be rotated as well as translated in order to reflect the new position and orientation of the joint. Secondly, the DRV of the stepped joint needs to ensure that the rotation of each joint j matches the rotation of the link connecting it to its parent, link($j-1, j$). Thirdly, the recursive repositioning reflects a root-first ordering [13], a specific case of the optimal ordering presented in Section IV-A.

Local Planning. DRV local planning finds a path between two samples S_1 and S_2 by stepping each joint from its position in S_1 to its position in S_2. The resulting path may be tested for collisions by checking each intermediate configuration. This is analogous to the RV local planner [13].

RRT Extension. DRV stepping can also be used for RRT construction in the same way as local planning by simply setting the target to step toward as the random sample to extend towards in traditional RRTs.

V. Experiments

We showcase DRVs in a set of simulated pick-and-place tasks using a Kuka Youbot [8]. This models the increasingly common application of robotic loading/unloading items from a container. We present a full pick-and-place problem along with experiments that isolate specific subproblems. Our results show that DRVs improve the robustness and solution time for PRM planners in these scenarios.

All experiments compare the performance of a PRM that samples either uniformly at random from $\mathcal{C}_{space}$or from

DRV-space. A scaled euclidean distance metric, straight-line local planner, and eight-nearest-neighbors connection strategy are used. All experiments are run on an Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz processor. Our algorithms are implemented in C++ and compiled with GCC v4.8. All times are averaged over ten runs. Runs were terminated after a maximum time of 4 hours.

A. Kuka Pick-and-Place

Our pick-and-place problem consists of a Kuka Youbot and a cupboard with many long, narrow cubby holes (Figure 6). The robot must reach into the cubby holes to reach a pre-defined grasping pose before moving to the drop-off bin. This emulates the planning process needed to pick an object from the shelves and place it in the bin.

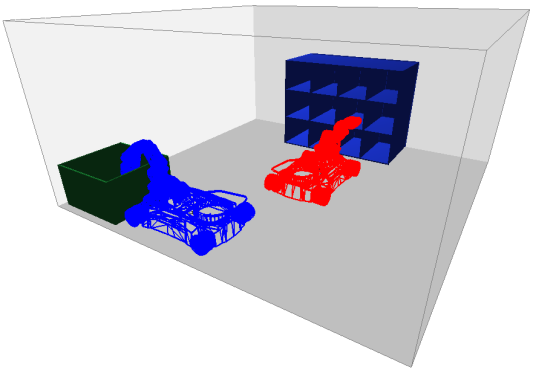


Fig. 6. The Kuka shelves environment, using an 8 or 10 DOF Kuka Youbot (K8 and K10, respectively). This pick-and-place problem requires the Youbot (K8 shown) to position its arm inside a cupboard for a grasping maneuver and then move to a drop-off container.

We use two variations of the Kuka Youbot: the standard Youbot with a 5 DOF arm (referred to as K8) and a fictitious variant that adds two extra links to increase the arm complexity to 7 DOF (referred to as K10). The movable base for both variants adds 3 DOF for totals of 8 and 10 DOF, respectively. All of the joints in both robots are revolute, making DRVs applicable to them while RVs are not.

We evaluated three variations for each robot where the grasping pose inside the cupboard is positioned at the front, middle, or back of the compartment. This is shown for K8 in Figures 7(a) - 7(c), and for K10 in Figures 7(d) - 7(e). Moving the grasping pose further into the compartment forces the robot to reach deep inside and makes the problem considerably more difficult. A spectrum of three difficulties is provided to illustrate a range of performance characteristics across realistic variants of the problem.

The results (Figure 8) show that DRVs are able to solve all variants of this problem more efficiently than uniform PRM. The sole exception is k10-f where DRV performs slightly worse than uniform. The time spent computing the initial DRV structure for the longer arm (K10) was between 90 and 100ms, and is included in all the results.

B. Pick

In this experiment, we isolate the 'pick' portion of the pick-and-place problem. We study the process of generating

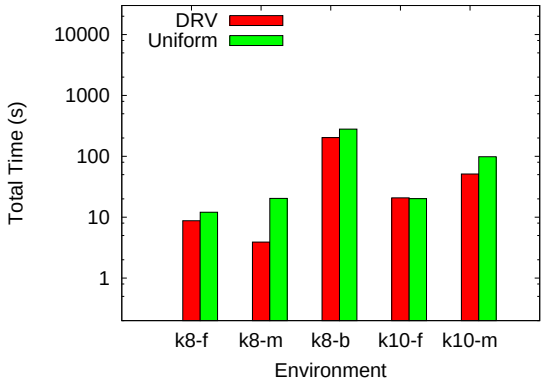


Fig. 8. Average time (log scale) to solve the pick-and-place problem. Stars indicate methods unable to generate samples in the allotted time.

valid grasping poses by generating samples where the robot is grasping objects in one of the cubbies.

For each robot we study three variations of this problem, a 'easy' variation where the robot's end effector can be located anywhere in the cubby hole, a 'medium' difficulty problem where the must be in the back 3 quarters of the cubby, and a 'hard' variation in which the end effector must be in the back half of the cubby. Example poses are shown in Figures 7(a) through 7(f). For each variation we generated 25 valid grasping samples and recorded the total sampling time (Figure 9).

Our results confirm that DRVs are better able to produce samples in the tight confines of the cubby holes, allowing them to solve the easier queries in less time (Figure 9). They also confirm that DRVs are able to find paths to the more difficult positions where the robot is grasping objects at the back of the cupboard.

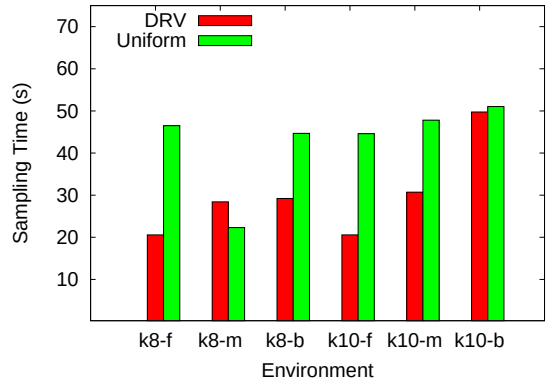


Fig. 9. Average time required to generate 25 valid pick samples.

C. Place

This experiment isolates the 'place' action, which consists of the robot reaching into a drop-off bin to emulate placing a picked object within. Similar to the pick problem, we use a fixed base here to highlight the manipulator portion of the problem. We again examined three variations (with short, medium, and tall drop-off bins, Figure 10) for each robot to

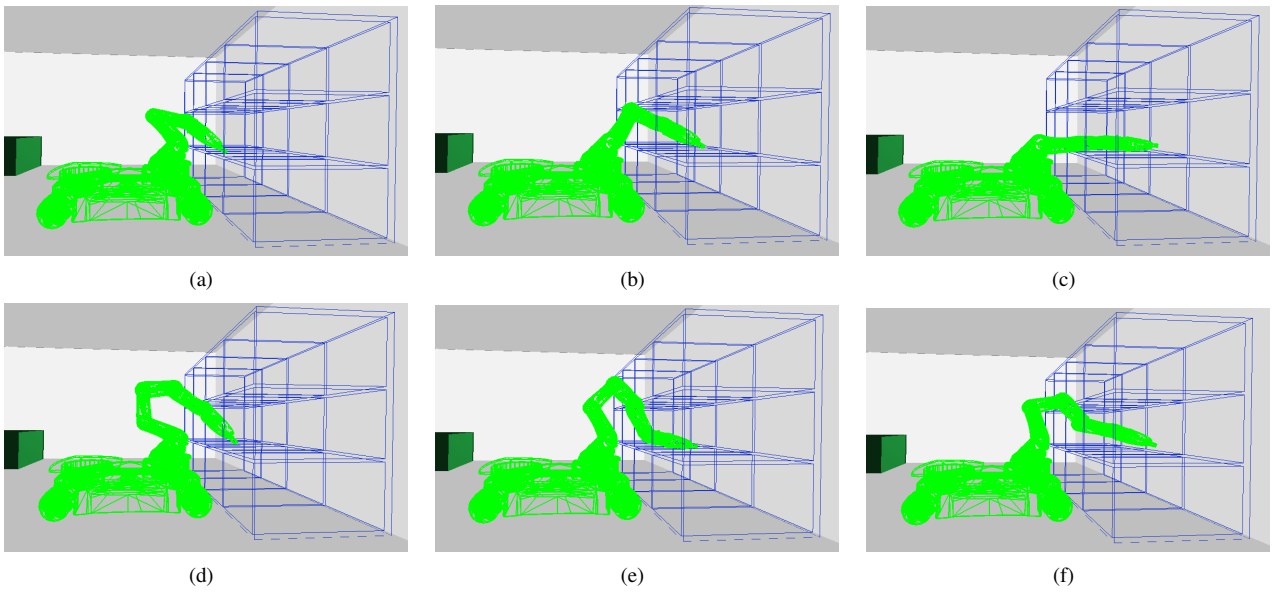


Fig. 7. The 'pick' portion of our pick and place problem. Target configurations of increasing difficulty for K8: (a) shows an easy pose at the front of the shelf, (b) shows an intermediate pose in the middle of the shelf, and (c) shows a difficult pose at the back of the shelf. (d-e) Equivalent problems for the longer K10 robot.

show how performance changes with problem difficulty. As the drop-off bin gets taller, the robot needs to reach further to get over the wall, and the gap between the box and the ceiling gets tighter. This results in fewer valid paths that lead to a successful drop maneuver.

Our results showed that in all cases, uniform sampling and DRVs achieved comparable performance (Figure 11). This demonstrates that even when the problem is sufficiently easy for uniform sampling, the performance benefits of DRVs are at least sufficient to compensate for their overhead.

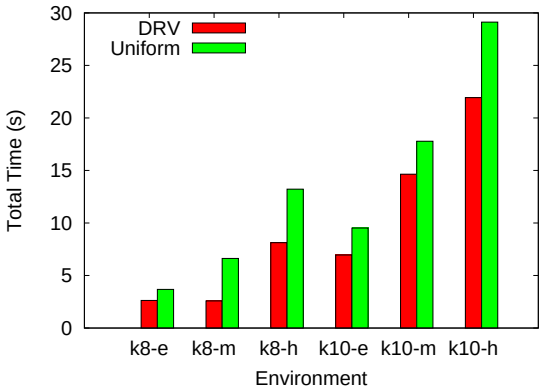


Fig. 11. Average time for the 'place' portion.

VI. Analysis

The improved performance with DRVs stems from iterative refinement of self-invalid configurations, resulting in a more even coverage over the distribution of valid configurations. By comparison, uniform sampling has greater difficulty in avoiding these self-collisions and must therefore spend significantly more iterations to find valid configurations.

More even coverage also results in more options for local planning. Configurations that are difficult to reach are, by definition, only reachable via a certain subset of the possible paths through $\mathcal{C}_{free}$. Using a straight-line local planner implies that those paths will only be generated from a certain subset of starting configurations. This suggests a significant advantage for DRVs compared with uniform sampling when a sufficient number of nearest neighbors are considered as the starting point, as the neighbors are more likely to be diverse enough to contain at least one acceptable starting point.

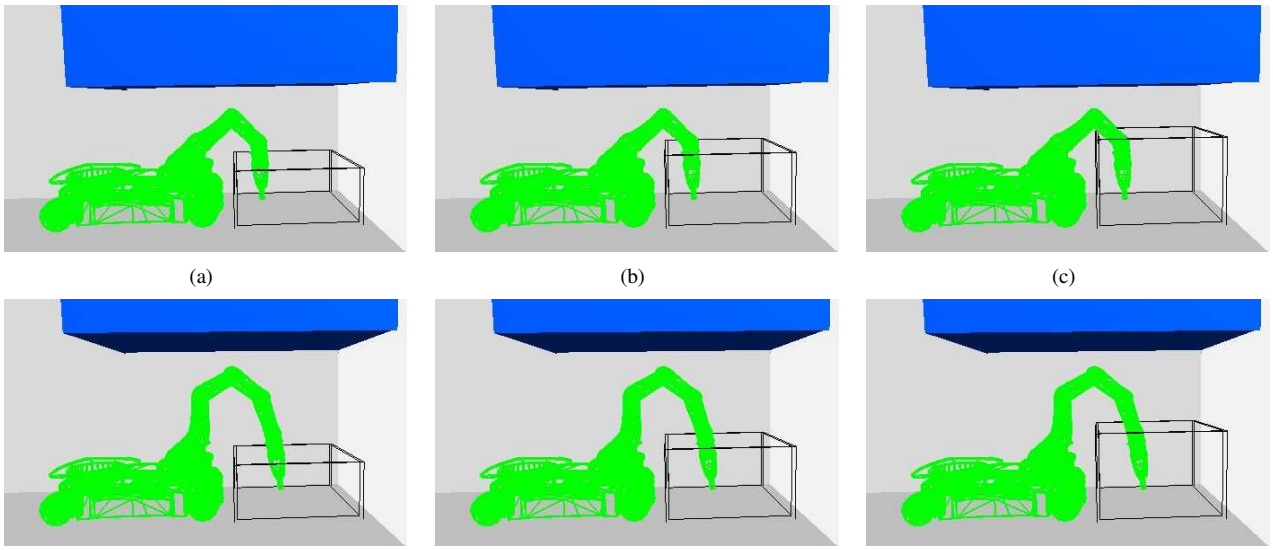
VII. Conclusion

In this work, we present the concept of directed reachable volumes (DRVs), which reparameterizes the planning space to give an explicit representation of the workspace regions that the joints and end effectors of a robot can reach. The resulting DRV-space (and supporting DRV planning primitives) makes finding paths for manipulators, including industrial robots with rotational joints, that avoid self-collision trivial. DRVs generalize prior RV work that cannot properly handle rotational joints and non-planar articulated joints that depend on both the position *and orientation* of other joints in the robot. Our experiments show that DRV-space offers benefits in planning time and robustness for manipulation problems compared with uniform sampling.

References

[1] K. Kotay, D. Rus, M. Vona, and C. McGray, "The self-reconfiguring robotic molecule: Design and control algorithms," in *Robotics: The Algorithmic Perspective*, P. K. Agarwal, L. E. Kavraki, and M. T. Mason, Eds. Boston, MA: A. K. Peters, 1998, pp. 375–386, book contains the proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR), Houston, TX, 1998.

[2] A. Nguyen, L. J. Guibas, and M. Yim, "Controlled module density helps reconfiguration planning," in *New Directions in Algorithmic and Computational Robotics*. Boston, MA: A. K. Peters, 2001, pp. 23–36, book contains the proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR), Hanover, NH, 2000.



(a) (b) (c)

(d) (e) (f)

Fig. 10. An isolated 'place' problem where the robot needs to reach a drop-off pose without moving its base. Low ceilings are added to increase the problem difficulty. (a-c) Target configurations of increasing difficulty for K8: (a) shows a short bin, (b) shows a medium bin, and (c) shows a tall bin. (d-f) Show the same problems with the K10 robot.

[3] 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)*, 1999, pp. 252–261.

[4] M. Kallmann, A. Aubel, T. Abaci, and D. Thalmann, "Planning collision-free reaching motion for interactive object manipulation and grasping," *Computer Graphics Forum*, vol. 22, no. 3, pp. 313–322, Sept. 2003.

[5] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1999, pp. 473–479.

[6] J. H. Yakey, S. M. LaValle, and L. E. Kavraki, "Randomized path planning for linkages with closed kinematic chains," *IEEE Trans. Robot. Automat.*, vol. 17, no. 6, pp. 951–958, 2001.

[7] Zhuohe Instrumentation Co., Ltd., "Using small assembly robots is a trend of automation packaging," http://www.meterforall.com/technical/automation-packaging-robots.html, 2009, accessed: Sept. 15, 2016.

[8] K. R. Corporation, "Kuka youbot," http://www.youbot-store.com, accessed: June 1, 2013.

[9] WillowGarage, "PR2 Robot," Palo Alto, CA, 2016, www.willowgarage.com.

[10] FetchRobotics, "Fetch robot," fetchrobotics.com, accessed: Sept. 15, 2016.

[11] T. McMahon, S. Thomas, and N. M. Amato, "Sampling based motion planning with reachable volumes: Theoretical foundations," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Hong Kong, China, May 2014, pp. 6514–6521.

[12] ——, "Sampling based motion planning with reachable volumes: Application to manipulators and closed chain systems," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, Chicago, Il., Sept. 2014, pp. 3705–3712.

[13] T. McMahon, S. L. Thomas, and N. M. Amato, "Reachable volume RRT," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Seattle, Wa., May 2015, pp. 2977–2984.

[14] 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.*, vol. 12, no. 4, pp. 566–580, August 1996.

[15] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *New Directions in Algorithmic and Computational Robotics*. A. K. Peters, 2001, pp. 293–308, book contains the proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR), Hanover, NH, 2000.

[16] A. A. Canutescu and R. L. Dunbrack, Jr., "Cyclic coordinate descent: A robotics algorithm for protein loop closure," *Protein Sci.*, vol. 12, no. 5, pp. 963–972, 2003.

[17] 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)*, Edmonton, Alberta, Canada, 2005, pp. 1875–1880.

[18] M. Zucker, N. Ratli, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant hamiltonian optimization for motion planning," *Int. J. Robot. Res.*, vol. 32, no. 9–10, pp. 1164–1193, 2013.

[19] G. Oriolo and M. Vendittelli, "A control-based approach to task-constrained motion planning," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 2009, pp. 297–302.

[20] L. Han and N. M. Amato, "A kinematics-based probabilistic roadmap method for closed chain systems," in *New Directions in Algorithmic and Computational Robotics*. Boston, MA: A. K. Peters, 2001, pp. 233–246, book contains the proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR), Hanover, NH, 2000.

[21] 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)*, New Orleans, LA, 2004, pp. 473–478.

[22] O. Salzman, K. Solovey, and D. Halperin, "Motion planning for multilink robots by implicit configuration-space tiling," *IEEE Robot. and Automat. Letters (RA-L)*, vol. 1, no. 2, pp. 760–767, July 2016.

[23] T. M. McMahon, "Sampling based motion planning with reachable volumes," Ph.D. Dissertation, Dept. of Computer Science and Engineering, Texas A&M University, August 2016.

[24] J. Denavit and R. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Trans ASME J. Appl. Mech.*, vol. 23, pp. 215–221, 1955.