BiSpace Planning: Concurrent Multi-Space Exploration

Rosen Diankov and Nathan Ratliff
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA
{rdiankov, ndr}@cs.cmu.edu

Dave Ferguson and Siddhartha Srinivasa
Intel Research Pittsburgh
4720 Forbes Ave
Pittsburgh, PA
{dave.ferguson, siddhartha.srinivasa}@intel.com

James Kuffner
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA
kuffner@cs.cmu.edu

***Abstract*— We present a planning algorithm called *BiSpace* that produces fast plans to complex high-dimensional problems by simultaneously exploring multiple spaces. We specifically focus on finding robust solutions to manipulation and grasp planning problems by using BiSpace's special characteristics to explore the work and configuration spaces of the environment and robot. Furthermore, we present a number of techniques for constructing informed heuristics to intelligently search through these high-dimensional spaces. In general, the *BiSpace* planner is applicable to any problem involving multiple search spaces.**

I. Introduction

One of the long-term goals of robotics is to develop a general purpose physical agent that can co-exist with, and provide assistance to, human beings. Substantial progress has been made toward creating the *physical* components of such an agent, resulting in a wide variety of humanoid robots that possess amazing potential for dexterity and finesse. But progress toward controlling such agents in real-time in unstructured, inhabited environments has been slower.

Planning algorithms represent the state-of-the-art in control strategies for these agents. However, in many real-world scenarios the action possibilities for the agent can become very high dimensional and contorted, rendering many algorithms ineffective. Moreover, the environments in which these agents need to operate are often not known a priori and are often dynamic. A successful planning algorithm, therefore, must perform quickly so that the resulting solution can be executed before the environment changes substantially.

In response to these planning challenges, a number of sampling-based search algorithms, such as the Rapidly-exploring Random Trees (RRT) family of algorithms [1], have been developed which demonstrate encouraging empirical performance on high-dimensional planning problems. RRTs in particular are easy to implement and have shown fast convergence to feasible solutions on a wide variety of motion planning scenarios [1, 2, 3, 4, 5].

In this work, we focus on the problem of mobile robotic manipulation. Specifically, the robot must be able to maneuver to an object and grasp it without a human specifying a priori where and how the object should be picked up. In these manipulation scenarios, the robot must find a feasible motion trajectory from its initial configuration to a grasp-achieving configuration. While most sampling-based motion planning algorithms assume the goal is a point within the configuration space of the robot, the goal in manipulation problems can consist of a continuous subset of the configuration space — any configuration of the robot resulting in a feasible grasp can be considered a goal configuration.

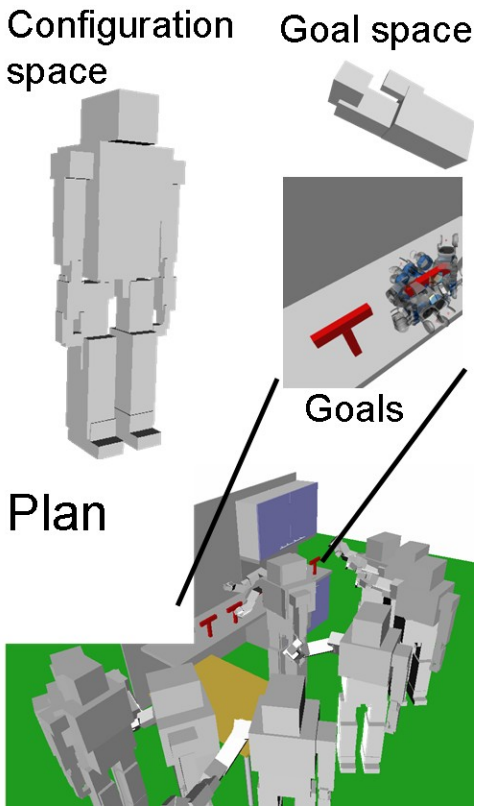


Fig. 1. Humanoid robot grasps an object by autonomously searching through the space of possible grasps and torso movements.

For example, consider a grasping problem where the robotic manipulator has to pick up a cup from the sink to put in a cupboard. The manipulator first has to move its end-effector close to the cup and then it has to achieve the correct contacts with the cup, constraints that are inherently defined in the workspace. In general, there is a fundamental

mismatch between the space used to describe the goals of the manipulation planning problem and the configuration space used to search for a solution.

Methods exist to overcome this mismatch, such as using inverse kinematics (IK) to transform the workspace constraints into configuration space goals [6, 7]. However, as the dimensionality of the problem increases the effective dimensionality of the goal set increases as well, often causing these methods to become prohibitively slow. Further, it has been shown that such approaches can fail to converge to a valid goal and are usually limited to finding only one of the potentially infinite number of goals [8]. When planning through obstacle-laden environments, having only one goal configuration is problematic, as this configuration is not guaranteed to be reachable from the initial configuration of the robot.

Instead, Bertram et al. [8] developed a nice extension to the RRT algorithm that removes the need for an IK solution and instead accommodates a goal specified in the workspace of the manipulator's end effector. In their approach they use a workspace goal metric to select the configuration in the search tree that is closest to the workspace goal and then extend out from this configuration in a random direction. This goal extension occurs randomly throughout the growth of the tree.

A more recent approach by Vande Weghe et. al. [9] uses the Jacobian Transpose to do an even more focused search toward a workspace goal. Their approach operates similar to Bertram et al.'s except during the extension stage they use the Jacobian Transpose to move through configuration space in the direction of the workspace goal, resulting in a more efficient overall search. However, both of these algorithms are restricted to growing a single, forwards-directed search tree, and therefore do not capture the benefits of the more efficient bi-directional RRT approaches [10].

Thus, current approaches are limited to either approximating the desired goal configurations and concentrating the entire search in configuration space, or to using an accurate workspace goal representation and growing a single goal-biased search tree. Alternatively, in the following section we present an approach that allows us to search in multiple spaces simultaneously, with a forwards-directed search tree grown through one space, known as the *configuration space*, and a backwards-directed tree grown in another space, known as the *goal space*. This algorithm, known as *BiSpace*, relaxes the need for an explicit mapping from one space to another and in our present application is able to significantly reduce the amount of searching that occurs in the full configuration space.

After describing the algorithm in depth we provide comparative results involving a collection of complex robotic mechanisms and present an experimental results involving a physical 11 degree of freedom manipulator arm.

II. The Bi-space Algorithm

The core idea of the BiSpace algorithm is to grow two different search trees at the same time. One tree explores the full configuration space starting from the initial configuration and guarantees feasible, executable, and collision-free trajectories, while the other tree explores the backspace starting from the set of goal configurations and acts as an adaptive, well informed heuristic. The BiSpace algorithm proceeds by extending RRTs in both spaces. Once certain conditions are met, the forward tree attempts to *follow* the goal space tree path to the goal (Figure 2). The algorithm combines elements of both bidirectional RRTs and the RRT-JT algorithm [9].

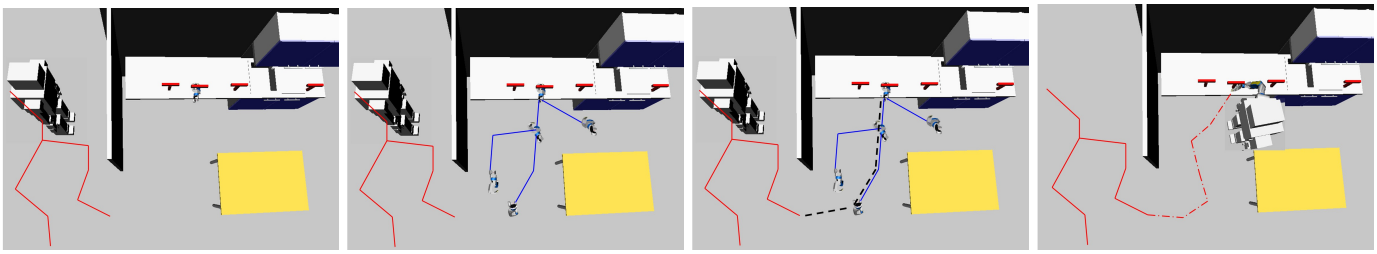
Algorithm 1: BiSpace(q_{init}, b_{goals})

```
/* ρ ∈ [0,1] - uniform random variable */
forward ← false
Init(T_f, q_init); Init(T_b, b_goals)
for iter = 1 to maxIter do
    if forward then
        for fiter = 1 to J do
            q ← Extend(T_f)
            if ρ < FollowProbability(q) then
                b_follow ← NearestNeighbor(T_f, q)
                {success, q'} ← FollowPath(q, b_follow)
                if success then
                    return success
        end
    else
        for biter = 1 to K do  Extend(T_b)
    forward ← not forward
end
return failure
```

For clarity, we denote a configuration with q and a goal space configuration with b. We assume that there exists a mapping $F(\cdot)$ from the configuration space to the goal space such that $F(q)$ maps to exactly one goal space configuration. Using this notation, given a goal space distance metric $\delta_b(F(q), b)$, the goal of planning is to find a path to a configuration q such that $\delta_b(F(q), b_{goals}) < \epsilon_{\text{goal}}$.

The flow of the BiSpace algorithm is summarized by Algorithm 1. The **forward** variable is used to keep track of which tree to grow. If **forward** is **true**, then the configuration space tree is extended **J** times, using the standard RRT extension algorithm Extend [1]. Alternatively, if **forward** is **false**, then the goal space tree is extended **K** times. After each iteration, the value of **forward** is flipped so that the opposite tree is extended during the subsequent iteration. After a new node q is added to the configuration space tree, a *follow* step is performed from q with probability FollowProbability(q). An example of such a distribution is described in detail in Section IV-C. If a *follow* step is performed, then q is extended toward b_{follow} and its parents.

The differences between BiSpace and BiRRTs become clear in the *follow* step. In the BiRRT case, *following* consists of connecting the two trees along the straight line joining q and b_{follow}; this is possible since b_{follow} is also in the configuration space. Because each branch of the both the forward and backward trees in the BiRRT algorithm represent a valid



(1) (2) (3) (4)

Fig. 2. BiSpace Planning: A full configuration space tree is grown out from the robot's initial configuration (1). Simultaneously, a goal back tree is randomly grown out from a set of goal space nodes (2). When a new node is created, the configuration tree can choose to *follow* a goal space path leading to the goal (3). Following can directly lead to the goal (4); if it does not, then repeat starting at (1).

collision free path in the configuration space, connecting the two trees immediately implies a path can be found from the start configuration to the goal configuration. However, that is not true with the BiSpace algorithm. Since the goal space is different from the configuration space, the path suggested by the goal space tree must be validated in the configuration space.

Each unique path from a node in the goal space tree to a goal can be used by the forward tree as a heuristic to informatively bias extension toward the goal. Starting from b_{follow}, such a path can be extracted by recursively following its parents. The forward tree can use the goal space path generated by b_{follow} as a bias to greedily *follow* it. If the forward tree succeeds in reaching the goal, a solution is returned (Figure 2). Otherwise, the search continues as before.

Algorithm 2: $\{success, q\} \leftarrow$ FOLLOWPATH(q, b)

```
  /* ρ ∈ [0,1] - uniform random variable   */
1 success ← false
2 for iter = 1 to maxFollowIter do
3     best ← null
4     bestdist ← γ_inflation * δ_b(F(q), b)
5     for i = 1 to N do
6         q' ← SAMPLENEIGHBORHOOD(q)
7         if δ_b(F(q'), b) < bestdist then
8             bestdist ← δ_b(F(q'), b)
9             best ← q'
10    end
11    if best is null then
12        if b.parent is null then
13            break
14        b ← b.parent
15    else
16        q ← T_f.add(q, best)
17 end
18 success ← δ_b(F(q), b.root) < ε_goal
   /* Optional IK test                    */
19 if not success and (q' ← IKSOLUTION(q, T_b.goals))
   then
20    {success, q} ← BiRRT(T_f, q, q')
```

Path following is an integral part of the BiSpace algorithm. It generates a very powerful bias as to where the configuration tree should grow by using the nodes in the goal space tree. Each goal space node has already validated a subset of the conditions necessary for the configuration tree to follow it. Although the FOLLOWPATH function is general, we present a simple, but effective, implementation of a stochastic gradient approach for it (Algorithm 2). The forward tree slowly makes progress by randomly sampling configurations that get close to the target goal space node b. Whenever the forward tree stops making progress, it checks if b has any parents. If it does, b is set to its parent and the loop repeats. If there are no more parents, the goal space distance from q to the final parent $b.root$ is checked: if this distance is within the goal threshold, the function returns success; otherwise it returns false.

FOLLOWPATH can require a lot of samples if SAMPLENEIGHBORHOOD uniformly samples the neighborhood of q. This is especially a problem for the high-dimensional configuration spaces used in manipulation planning. Instead, we sample each of the dimensions one at a time while leaving the rest fixed. This type of coordinate descent method has been shown to perform better than regular uniform sampling in optimization and machine learning algorithms [11]. Furthermore, SAMPLENEIGHBORHOOD can incorporate the Jacobian transpose idea from [9] to further bias samples in the correct direction. Because it is not always beneficial to be greedy due to many local minima, we introduce $\gamma_{inflation}$ to relax the distance metric we are minimizing[1].

As an optional addition to the FOLLOWPATH algorithm, we propose using IK solutions, if they exist, to speed up planning. After the forward tree terminates at a configuration q, an IK solution can be checked for a subset of the DOFs of the configuration space. If there exists a solution, we can run a bidirectional RRT using the subset of DOFs used for IK to find a path from q to the new goal configuration. For example, if a 7 DOF arm is mounted on a mobile platform, its full configuration space becomes 10 dimensional, however, the arm's IK equations will still remain 7 dimensional. In this case, IKSOLUTION$(q, goals)$ will use the arm's position from the last configuration q and check the standard arm IK. Having such a check greatly reduces planning times and is

[1]This has a similar effect to inflating the goal heuristic in A*. We use $\gamma_{inflation} = 1.4$ for all results.

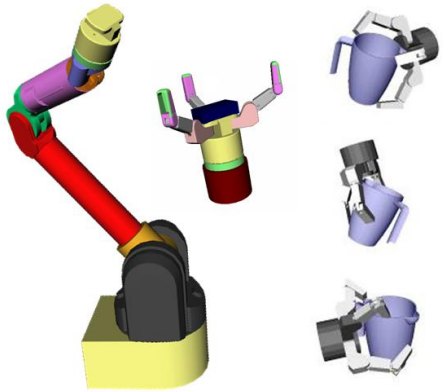


Fig. 3. The hand and arm area treated separately in the grasp planning framework. Only the hand is allowed contact with the environment. The arm is only used for planning.

not prohibitively expensive if the IK equations are in closed form. While some algorithms ignore IK solutions, BiSpace can naturally use inverse kinematics to its advantage. Empirical results suggest that BiSpace can experience a 40% decrease in planning time when exploiting available IK solutions. However, it should be noted that having inverse kinematics equations for a robot does not guarantee feasible solutions can be efficiently found.

III. Applications to Manipulation and Grasp Planning

In grasp planning, the task is to plan for an arbitrarily complex robot to move to pick up any object in the environment. Ideally, the planner decides how to best grasp the object and how to manipulate the robot to achieve the desired grasp. Recently, [6] proposed a method to solve this problem when the robot is stationary and there exist IK equations that provide an efficient mapping from workspace to configuration space. They use a two-tiered approach: they first find the workspace positions of any feasible grasps by sampling from a precomputed table and testing in the real environment, then seed a BiRRT planner with the IK solution of each of those workspace positions. Applying their method to a mobile robot poses several challenges because it relies on producing fast configuration space goals from workspace positions of the end-effector of the robot. As we show, this two-tiered approach is much slower than using BiSpace. Nevertheless, our systems level approach to solving manipulation and grasp planning for mobile robots is inspired from their method (Figure 4).

We divide each robot into two semantic pieces: the *hand* and the *arm* (Figure 3). Only the hand can make contact with the target object. Following [6], grasp tables can be precomputed for every hand-object pair. These grasp tables are computed with the detached *hand* approaching the object from all possible directions with all possible preshapes; usually the final tables are on the order of 300-800 good grasps per object (Figure 1). In the real environment, each grasp in the table is tested against the object and environment for collisions. Note that collisions are only checked with the detached *hand*

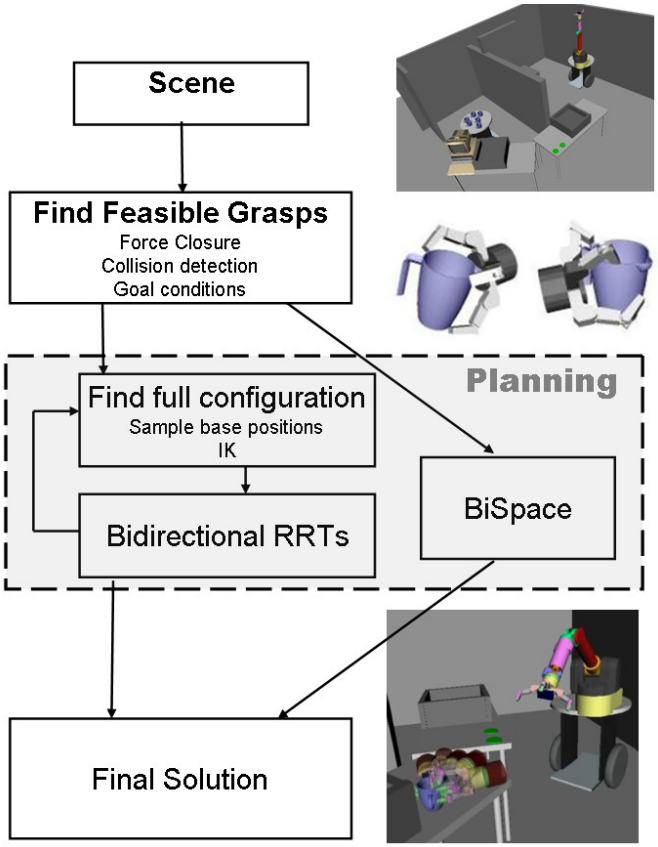


Fig. 4. A framework for grasp planning. BiSpace allows exploration of the space without locking down on a particular IK solution.

since the full configuration of the robot is unknown. Once a collision-free grasp is found, we treat the final hand pose as the goal. The goal of the BiSpace planner is now to move the robot so that the hand attached to it achieves the desired grasp. For simplicity we assume that the planner only moves the *arm* and the mobile base it is mounted on.

Since BiSpace is a randomized algorithm, in general it cannot detect in a finite amount of time that a given collision-free grasp is impossible to reach. Therefore, seeding BiSpace with only one grasp at a time is dangerous as the planner might never find a solution. Instead, it is favorable to seed the BiSpace planner from the beginning with as many feasible grasps as possible using the precomputed grasp tables, increasing the likelihood that at least one of the grasps can be reached. Since the EXTEND operation is not affected by the number of trees being grown, incorporating multiple goals in the goal space does not affect efficiency [12].

IV. Manipulation Planning Heuristics

We introduce several heuristics to the BiSpace planner in order to improve planning efficiency. Each of these heuristics assumes that every robot is composed of a mobile base and at least one arm whose end-effector is used to make contact with the target objects. Note that no assumptions are made about the kinematics of any of the robot's parts, and both humanoid

and wheeled robots are applicable in this framework (as we demonstrate in Section V). Furthermore, each heuristic can be automatically derived for any robot platform, making them ideal for general use.

A. Base Reachability

Many researchers have shown that using some form of goal biasing by modifying the configuration space sampling distribution greatly reduces planning times. This section tackles the problem of intelligently biasing configuration space samples when the only goals given are the final grasps.

Given a target grasp g, we would like to quickly compute a distribution $P_g(p, \theta)$ over the 2D placement (p, θ) of the base of the robot for which g will be successful.

We first perform a kinematics workspace analysis for the arm similar to [13]. Figure 5 shows the hand reachability volume generated for the HRP2 humanoid robot. This was computed by randomly sampling a 6D end-effector position around the space of the humanoid's shoulder and querying for an IK solution. It can similarly be computed by randomly sampling arm configurations and storing their resultant 6D end-effector. We store all the valid 6D end-effector positions in $\mathcal{X} = \{(x_q, x_t)\}$ where x_q and x_t are the rotation and translation associated with the transformation of the end-effector. Clearly, to succeed in the planning for a specific grasp, the robot should move its body so that its reachability volume coincides with the particular grasp.

Each grasp g represents an affine transformation where g_t is the translation and g_q is the rotation. Our goal is to find similar grasps to g in $\mathcal{X}$ and perform simple counting to extract a probability of existence of an IK solution.

We begin by defining a rotation on the plane as $R_n(\theta)$ where n is the normal vector to the plane the robot rotates on. The group of all rotations on the plane is denoted by

$$\mathcal{Q}_n = \{R_n(\theta) \mid \theta \in \mathbb{S}^1\} \tag{1}$$

We can now define an equivalence class of rotations that differ only by a rotation about the plane as

$$g_q \, \mathcal{Q}_n = \{r * g_q \mid r \in \mathcal{Q}_n\}. \tag{2}$$

where $*$ denotes the action of applying one rotation after another. We then define the equivalence class of all similar grasps up to a rotation on the plane as

$$\mathcal{X}_g = \{(g_q, (g_q * x_q^{-1})(x_t)) \mid x \in \mathcal{X},\ x_q \in g_q \, \mathcal{Q}_n\} \tag{3}$$

where $(g_q * x_q^{-1})(x_t)$ transforms the position of the end effector from the frame of the robot (in the reachability map) to the frame of the grasp.

We compute equivalence classes for the entire set $\mathcal{X}$ by sampling x_i, storing its equivalence class $\mathcal{X}_i$ and continuing sampling from $\mathcal{X} - \mathcal{X}_i$. Doing this greatly reduces the number of grasps from over $100,000$ in $\mathcal{X}$ to about 100 equivalence classes. In practice, we accept grasps if they are within a threshold of the rotation x_q.

We now compute the inverse reachability volume $\mathcal{D}_g(p, \theta)$ for each equivalence class g. Note that each end-effector position x_t where $x \in \mathcal{X}_g$ has been aligned to the frame of the grasp g. p and θ are still in world coordinates and need to be converted to the grasp coordinate system induced by g. This is achieved by

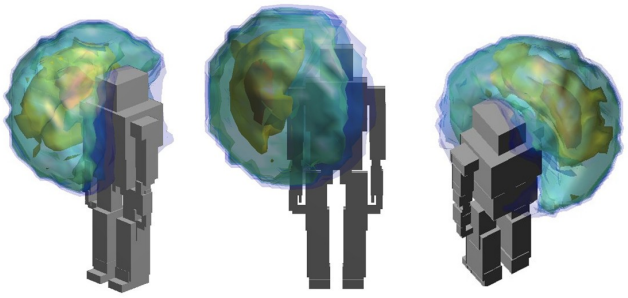


Fig. 5. 6D hand reachability from a given base placement projected in 3D. Shown are three different views of the reachability volume. Dark opaque areas contain more reachable end-effector positions.

$$\mathcal{D}_g(p, \theta) = \{ \, \|x_t - (R_n(\theta) * g_q)^{-1}(g_t - p)\| < \epsilon \mid x_t \in X_g \} \tag{4}$$

Finally the inverse reachability map converts $\mathcal{D}_g(p, \theta)$ into a probability distribution as

$$P_g(p, \theta) = \exp\left\{ -\omega \left(\frac{\sum_{d \in \mathcal{D}_g(p,\theta)} d}{|\mathcal{D}_g(p, \theta)|} \right)^2 \right\} \tag{5}$$

Views of the map for various equivalence classes g in a sample scene for the HRP2 humanoid are shown in Figure 6.

To test the effectiveness of the inverse reachability map, we randomly sampled base positions using $P_g(p, \theta)$ to see if they would contain feasible IK solutions. Empirical results showed that $P_g(p, \theta)$ was able to generate a feasible base placement 2.5 times faster than uniform sampling around the grasp.

B. Workspace Exploration

As humans, we employ different navigation strategies based on our distance to a goal object. When a person is far away from an object of interest, they care primarily about moving their body in a direction that will get them close to the object. When they are close, they usually plant their feet and use their arms to make contact with the object. We can achieve the same behavior in BiSpace by modifying the configuration space distance metric such that

$$\delta(q) = |\omega(q) q_{arm}| + |q_{base}| \tag{6}$$

where q_{arm} is the degrees of freedom associated with the arm. When the robot base is far away from the goal, the weight ω should be small so that the robot takes bigger steps on average. This suggests a simple monotonic function for ω:

$$\omega(q) \propto \exp\left\{ -\frac{\min_i |goal_i - BasePosition(q)|^2}{2\sigma^2} \right\} \tag{7}$$

where σ is proportional to the length of the arm. Figure 7 demonstrates the behavior of BiSpace when using the modified

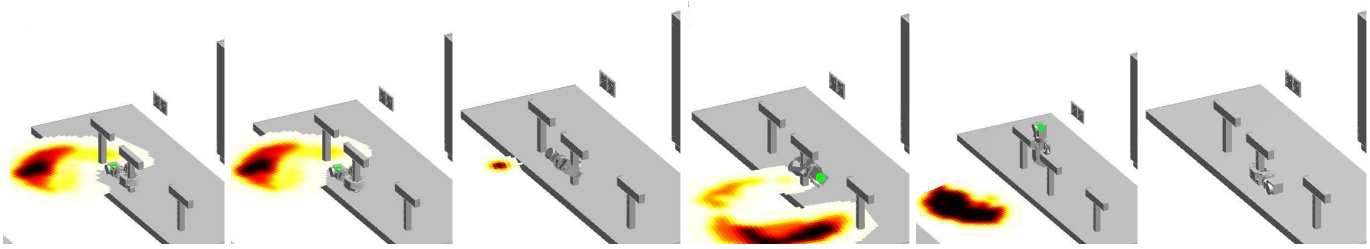


Fig. 6. Base placements derived from the goal space goals. As described in the text, these placements can be efficiently generated using the pre-computed hand reachability volume.

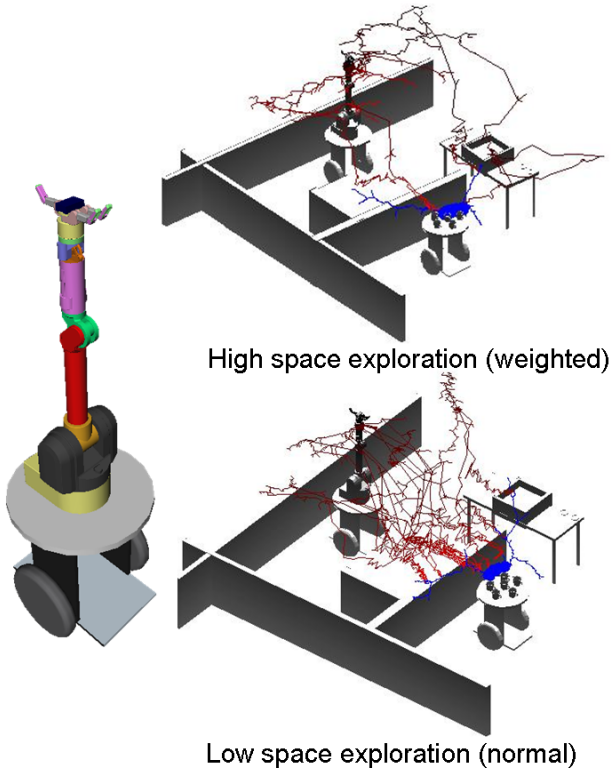


Fig. 7. Comparison of how the distance metric can affect the exploration of the arm. The top image shows the search trees (red/black) generated when the distance metric and follow probability is weighted according to Equation 6. The bottom image shows the trees when the distance metric stays uniform across the space; note how it repeatedly explores areas. The goal space trees are colored in blue.

distance metric, and empirical results show that planning times reduce by 20% when this metric is used.

C. Follow Probability

The farther the robot is away from the goal, the less chance it will have of reaching it through FOLLOWPATH. The reason is because FOLLOWPATH itself is not exploration-centric like RRTs; it is meant for greedily approaching the goal when the body and hand of the robot are relatively unobstructed by complex environment obstacles. We propose two metrics to compute the follow probability: the hand reachability volume (Figure 5) or the distance falloff $\omega(q)$ (Equation 7). Both metrics monotonically decrease as the robot gets farther from the goal. The hand reachability is more informed since it is a 6D table reflecting the real arm kinematics while $\omega(q)$ is much easier to compute and often very effective (Figure 7). The correct follow probability can have a dramatic effect on planning times, sometimes reducing it by 60-70%.

V. RESULTS

For all planners, simulations, and real-robot experiments, we used an open-source planning test-bed called OpenRAVE [14]. There are few existing algorithms that work in high-dimensional spaces and cope with goals that are not specified explicitly in the configuration space, making direct comparison of BiSpace a little challenging. We chose to compare BiSpace with RRT-JT [9] and the two-tiered BiRRT approach described in Section III. Whenever the robot is mobile in a test scene, it adds 3 degrees of freedom to its configuration since its base can translate and rotate freely on the floor. Because randomized algorithms are known to have a long convergence tail, we terminate the search after 10-20 seconds and restart. This termination strategy produces much faster average times for all algorithms. Note however that every termination counts against the final planning time for that particular algorithm. Termination times were uniquely set for each algorithm in order to give it the fastest possible average time. Each algorithm is run on each scene 16-30 times, and the average planning time is recorded in Table I. Other parameters like RRT step size and goal thresholds were kept the same for all algorithms. To demonstrate the generality of the proposed algorithms, we produced results using both the HRP2 humanoid and a WAM arm loaded on a segway (Figure 8).

Since BiRRTs operate only in the full configuration space it would be unfair if they were seeded with the final solutions without any penalties. In order to make comparison fair, we randomly sample full configuration solutions for a given target grasp until a collision-free, feasible configuration is generated. The recorded time is added to the final planning time. The sampling takes somewhere from 2-9 seconds for HRP2 and less than 1 second for the WAM on segway[2].

[2]The large difference in sampling times is because the reachability area for the WAM is much larger

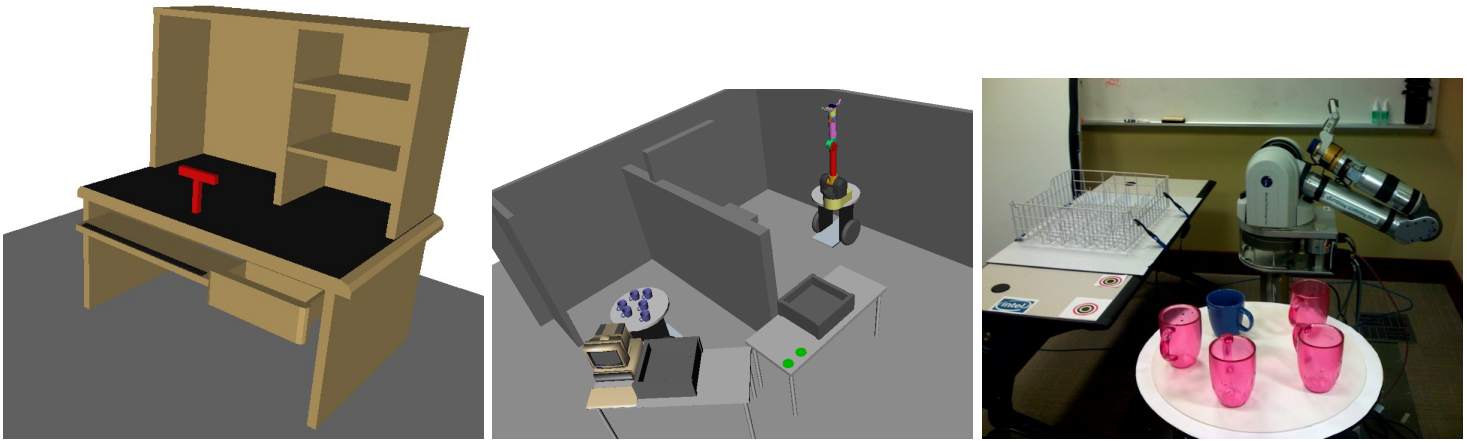


Fig. 8. Scenes used to compare BiSpace, RRT-JT, and BiRRTs.

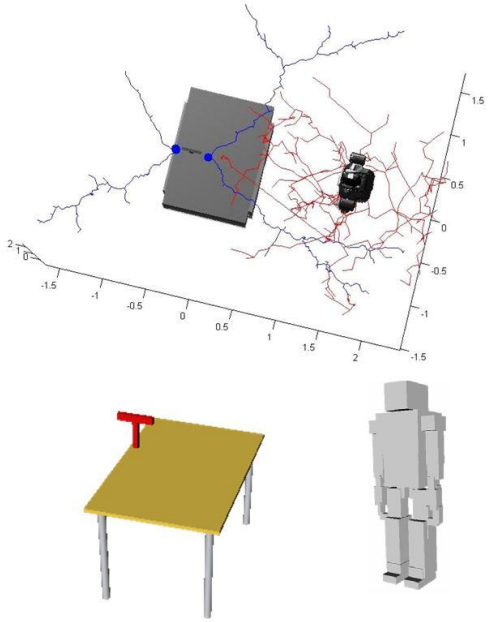


Fig. 9. Hard scene for BiSpace. The forward space tree (red) does not explore the space since it is falsely led over the table by the goal space tree (blue).

	BiSpace	RRT-JT	BiRRTs
HRP2 table (11 DOF, easy)	**33**	53	68
HRP2 table (11 DOF, harder)	**45**	528	78
HRP2 random (11 DOF)	**37**	170	40
WAM/segway (10 DOF)	**17.25**	25.2	22.93
WAM (7 DOF)	0.44	11	**0.37**

TABLE I

AVERAGE PLANNING TIME IN SECONDS FOR EACH SCENE.

A. HRP2

When planning for the HRP2 robot, we make the assumption that its base can freely travel on the floor and the legs do not need to move. Once BiSpace has planned a global trajectory, later footstep planners can add the necessary leg movements and dynamics to make the HRP2 move. In order to allow for leg space, an invisible cylinder is super-imposed over the lower body. Thus the planning space for HRP2 is reduced to 11 degrees-of-freedom: 3 for the base, 1 for the waist, and 7 for the arm. As can be seen from Figure 5, most of the hand reachability lies shoulder height to the side of the robot. This makes it hard for the robot to manipulate objects in front of it at waist height, which is why all the planners require significant planning time.

One of the hardest scenes for BiSpace is when the target object is on a table and HRP2 has to circle the table to get to it (Figure 9). Here, the goal space tree produces many false paths directly over the table, which the HRP2 cannot follow to the end. This process goes on until the rest of the configuration space tree finally explores the space on the other side of the table. This limitation is characteristic of bi-directional RRTs also and provides a good example of why exploration is always a crucial ingredient in sampling-based planners.

B. WAM

We tested two main scenes for the WAM: a living room scenario where the WAM is mobile, and a scenario where the WAM has to put cups in a dishwasher. The WAM arm has 7 degrees of freedom and very high reachability making planning very fast. BiSpace compares relatively well with BiRRTs, however it is a little slower due to the extra overhead in the FOLLOWPATH function. We also tested the entire grasp planning framework using the BiSpace planner on a real WAM arm setup (Figure 8). The WAM arm runs in real-time and can compensate immediately for changes in the environment.

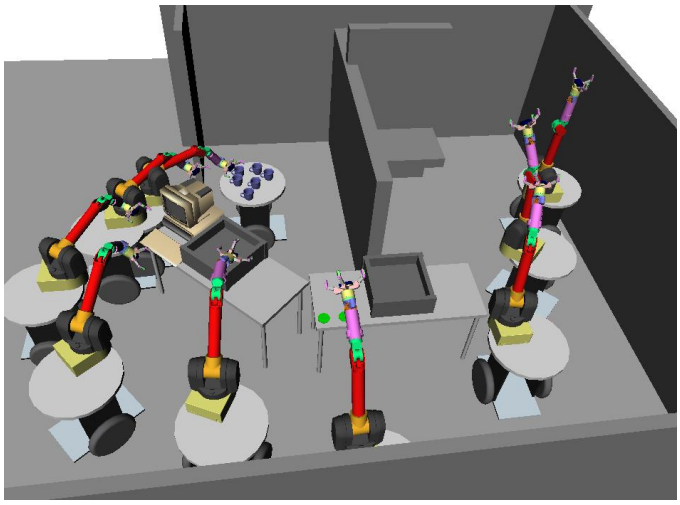


Fig. 10. Manipulation and Grasp Planning solutions for one of the test scenes.

Fig. 11. The real WAM arm grasping mugs.

VI. Conclusions

We presented the BiSpace algorithm for efficiently producing solutions to complex path planning problems involving a goal space that is different from the configuration space. We used this algorithm to plan for several mobile robots to perform manipulation tasks. One key feature of the grasp framework we employed is that it makes very few assumptions about how the robot should move to manipulate its target object, which makes it ideal for autonomous robot scenarios. Furthermore, we showed several heuristics that exploit various information about the kinematic structure of the mechanism to speed up planning. Finally we presented results for a real WAM arm loading cups into a dishwasher rack.

VII. Acknowledgements

The authors would like to thank the Digital Human Research Center for the allowing the use of the humanoid HRP2 model in these experiments. This material is based upon work supported in part by the Quality of Life Technology Research Center as part of the National Science Foundation under EEC-0540865.

References

[1] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[2] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, "Motion planning for humanoid robots," in *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2003.

[3] J. Kim and J. Ostrowski, "Motion planning of aerial robots using Rapidly-exploring Random Trees with dynamic constraints," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[4] G. Oriolo, M. Vendittelli, L. Freda, and G. Troso, "The SRT Method: Randomized strategies for exploration," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[5] D. Ferguson and A. Stentz, "Anytime RRTs," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[6] D. Berenson, R. Diankov, K. Nishiwaki, S. Kagami, and J. Kuffner, "Grasp planning in complex scenes," in *Proceedings of IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2007.

[7] C. Klein and C. Huang, "Review on pseudoinverse control for use with kinematically redundant manipulators," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 13, no. 3, pp. 245–250, 1983.

[8] D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour, "An integrated approach to inverse kinematics and path planning for redundant manipulators," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

[9] M. V. Weghe, D. Ferguson, and S. Srinivasa, "Randomized path planning for redundant manipulators without inverse kinematics," in *Proceedings of IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2007.

[10] J. Kuffner and S. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[11] Z. Luo and P. Tseng, "On the convergence of coordinate descent method for convex differentiable minimization," *Journal of Optimization Theory and Applications*, vol. 72, no. 1, pp. 7–35, 1992.

[12] K. Okada, T. Ogura, A. Haneda, J. Fujimoto, F. Gravot, and M. Inaba, "Humanoid motion generation system on hrp2-jsk for daily life environment," in *International Conference on Machantronics and Automation (ICMA)*, 2004.

[13] F. Zacharias, C. Borst, and G. Hirzinger, "Capturing robot workspace structure: Representing robot capabilities," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[14] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34, July 2008.