Path Planning in 1000+ Dimensions Using a Task-Space Voronoi Bias

Alexander Shkolnik and Russ Tedrake

***Abstract*— The reduction of the kinematics and/or dynamics of a high-DOF robotic manipulator to a low-dimension "task space" has proven to be an invaluable tool for designing feedback controllers. When obstacles or other kinodynamic constraints complicate the feedback design process, motion planning techniques can often still find feasible paths, but these techniques are typically implemented in the high-dimensional configuration (or state) space. Here we argue that providing a Voronoi bias in the task space can dramatically improve the performance of randomized motion planners, while still avoiding non-trivial constraints in the configuration (or state) space. We demonstrate the potential of task-space search by planning collision-free trajectories for a 1500 link arm through obstacles to reach a desired end-effector position.**

I. Introduction

Planning trajectories through space with obstacles is a computationally challenging, PSPACE complete problem [1]. The notion of planning a trajectory in a system with over a thousand dimensions seems remote. Yet, many systems in nature have hundreds of controlled Degrees of Freedom. The human body, for example has over 600 muscles, and over 200 bones, more than half of which are found in the hands and feet. Recent advances in randomized kinodynamic planners, including the Rapidly exploring Randomized Tree (RRT) [2], [3], have enabled fast average-time planning, including guarantees for probabilistic completeness. These planners can be very fast when planning trajectories within configuration space [1], [4] (C-Space). Despite the success of RRTs in certain classes of high-dimensional problems, the performance of the algorithm can be very sensitive to its implementation, particularly in how random samples are chosen, and the distance metric used. Additionally, as with most planners, the performance scales super-linearly with the number of dimensions.

In this paper, we argue that fast planning in high dimensional systems can sometimes be achieved by exploring actions in a low dimensional Task Space. This approach is commonly used in feedback control design, and is often called Task Space Control [5]. Consider a humanoid robot which needs to find its way from a start pose to cross a room full of obstacles, and pick up an object [6]. For such problems, it is natural to consider the task space. Rather than specifying a particular set of desired joint angles for the whole robot which accomplishes a grasping task, one may consider only the hand position. There are an infinite number of potential goal configurations which correspond to the desired hand position. Having a large set of goals can sometimes make planning easier, as there will be more trajectories that lead to a goal configuration. Furthermore, we argue that the defined task space can be used to guide the search, even when planning still occurs in the high-dimensional space in order to enforce all of the system constraints.

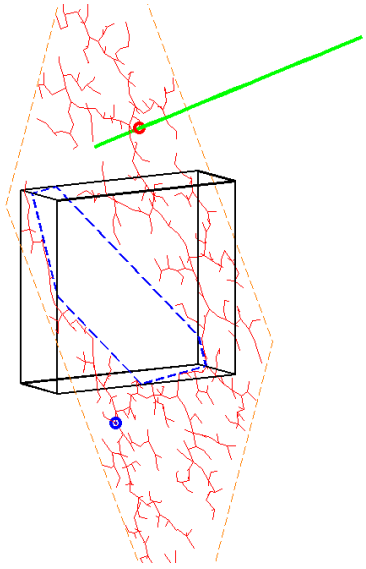

Fig. 1. Simple example, demonstrating an RRT planning in a 3D configuration space, while constrained to actions in a 2D task space. The task depicted is to find a path from the starting node (blue circle), to the goal in task space (green line), while avoiding an obstacle (Black Box). For simplicity, we resolve redundancy by restricting the search to lie in a plane (shown by red dash) perpendicular to the goal line, but other suitable choices for redundancy resolution would not be constrained to a manifold. The intersection of the obstacle on this plane is shown in blue dash line. For problems like this, planning in a lower dimensional projection can be much more efficient than planning in the full space.

To date, task space has been somewhat neglected in the planning literature, though there are a few recent examples that utilize it. In [7], for example, the sampling of points is biased by a heuristic function which includes distance to the goal, so that points closer to the goal in task space are more often randomly selected. The selected nodes are then extended with a randomly applied action. In a variation of this work by [8], a standard RRT is implemented which occasionally chooses to grow toward the goal in task space by using the Jacobian Transpose method. The goal of that work was to solve the problem of inverse kinematics at the

A. Shkolnik is a PhD candidate in EECS at the Computer Science and Artificial Intelligence Lab, MIT, 32 Vassar St., Cambridge, MA 02139, USA shkolnik@mit.edu

R. Tedrake is an Assistant Professor of EECS at the Computer Science and Artificial Intelligence Lab, MIT, 32 Vassar St., Cambridge, MA 02139, USA russt@mit.edu

goal region because infinite configurations may correspond to a single point in task space. Most recently, [9] proposed an approach that enables bidirectional RRT search [10], where a backward tree (grown from the goal) is grown in Task Space, and a forward tree (from the start pose) is grown in the full configuration space. Occasionally, the forward tree attempts to follow the backward tree using either random sampling of controls or the Jacobian transpose. These works utilize task space as an intuitive way of defining an infinite (continuous) set to describe a goal position, rather than using task space as a potential way to constrain the exploration of the full space.

Another body of literature has focused on sample based planning algorithms for systems that are constrained to a manifold defined by a lower-dimensional workspace, for example for a robot that is writing on a wall, it must keep the end effector on the surface of the wall [11]. In that work samples are constructed in configuration space that meet the constraints imposed by the lower-dimensional manifold, but no attention is given to how the particular choice of sampling affects planner performance, which is important given that there may be infinite ways to construct such a sampling schema. In the ATACE framework [12], configurations are sampled uniformly in C-Space, while a workspace constraint is imposed by utilizing a Jacobian pseudoinverse method. A similar C-Space sampling based method with task space constraints is provided by [13]. Our method differs in that we propose to sample directly within Task Space. Doing so creates a Voronoi bias in Task Space, rather than in Configuration Space, which for many systems leads to more direct exploration. The other difference is that we do not require a low dimensional manifold constraint, but instead consider the projection into Task Space. Because of this, our method could theoretically directly search the full high dimensional C-Space, despite the attention to exploring in the lower dimensional Task Space.

Constraining a search to a lower dimensional space can be a powerful notion. For visualization, consider for example the task of planning in a 3D configuration space. The goal might be to find a path from a starting coordinate to some region of the space, while avoiding obstacles. Suppose the goal region is a line. Then one way to explore the space is to grow in the plane which is perpendicular to the line, and intersects the starting node (as shown in Figure 1). If there were no obstacles present, this type of search could produce the most direct (shortest) path in configuration space. Because this plane slices through the 3D configuration space, as long as the plane has a path from start to goal, then it would make sense to actually plan while constrained to this 2D plane, which drastically reduces the set of potential solutions that need to be explored. Now imagine if rather than a 3D configuration space, we have a large DOF system. If a plane can be defined which captures a large portion of the feasible set of trajectories from start to goal, then it would be much more efficient to plan within this plane rather than the full configuration space. A potential objection would be that if the planning problem can be solved in this low-dimensional space, then it is not a challenging planning problem. However, we would argue that in fact many of the success stories in whole-body motion planning have this property.

In this paper we propose a framework for task-space biased exploration for search algorithms. For simplicity, as an example, we extend the standard forward RRT [2] so that when trying to grow from a particular node on the tree toward a sample, this growth is done in a low-dimensional task space. The growth can be done efficiently by using the Jacobian Pseudoinverse ($\mathbf{J}^+$) method, commonly used in feedback task-space controllers, which eliminates the need for exact inverse kinematics solutions. This task space will have an associated nullspace, the use of which is left to the freedom of the algorithm implementation. The pseudoinverse method allows for redundancy resolution via a hierarchical control structure which takes advantage of the nullspace. One can think of the standard RRT as randomly exploring in the nullspace. However, when actions are constrained (which is usually the case), then as the dimensionality increases, it is increasingly likely that a random vector in configuration space is orthogonal to a direction in task space, producing a trade off between exploring directly in task space, or exploring the configuration space.

Extending nodes in task space by the ($\mathbf{J}^+$) method also opens the door to sampling directly in task space, without considering the full space. If using a uniform sampling strategy over the task space, and a euclidean distance metric, then the proposed algorithm has the property that the Voronoi bias associated with the RRT will be in task space instead of in configuration space. We will show that the two spaces are quite different, and for certain problems, the Voronoi bias in task space can significantly improve the search performance. The algorithm is demonstrated for path planning a collision free trajectory in an N-link arm, with the task of putting the end-effector at a desired coordinate. We are able to efficiently plan paths with up to 1500 degrees of freedom in under a minute.

The proposed method can be thought of as planning in task space, while checking and validating constraints in the full configuration (or state) space. The forward RRT was chosen for its simplicity, but this method can be extended to a variety of planning algorithms, including PRMs [14], bidirectional RRT's, e.g. [10] - either by implementing BiSpace [9], or directly if goal(s) are known in configuration space.

II. TS-RRT: Planning in Task Space with the RRT

A. RRT and TS-RRT Definitions

The standard RRT algorithm [3] is provided for reference in Algorithm 1 and 2. In our notation, we denote a vector in configuration space with $\mathbf{q} \in \Re^N$, and a vector in task space with $\mathbf{x} \in \Re^M$, with an associated task-space mapping $\mathbf{x} = f(\mathbf{q})$, and a Jacobian defined as $\mathbf{J} \in \Re^{M \times N} = \frac{\partial f(\mathbf{q})}{\mathbf{q}}$. A C-Space sample, $\mathbf{q}_{rand}$, is chosen at random, usually from a uniform distribution over some bounded region. The closest node on the tree, $\mathbf{q}_{near}$, is selected, usually using a Euclidean distance metric, in the NEAREST-NEIGHBOR() function.

Algorithm 1 BUILD-RRT (q_0)

```
T.init(q_0)
for k = 1 to K do
    if with some probability P then
        q_rand ⇐ RANDOM-SAMPLE() ∈ ℜ^N
    else
        q_rand ⇐ q_goal
    end if
    EXTEND(T,q_rand)
end for
```

Algorithm 2 EXTEND (T, q)

```
q_near ⇐ NEAREST-NEIGHBOR(q,T)
u ⇐ CONTROL(q_near, q_rand)
q_new ⇐ NEW-STATE(q_near, u)
if COLLISION-FREE (q_near, q_new) then
    T.add-node(q_near, q_new, u)
end if
```

Algorithm 3 BUILD-TS-RRT (q_0)

```
T.init(q_0)
for k = 1 to K do
    if with some probability P then
        x_rand ⇐ RANDOM-SAMPLE() ∈ ℜ^M
    else
        x_rand ⇐ x_goal
    end if
    TS-EXTEND(T,x_rand)
end for
```

Algorithm 4 TS-EXTEND (T, x)

```
[q_near, x_near] ⇐ TS-NEAREST-NEIGHBOR(x,T)
u ⇐ TS-CONTROL(q_near, x_near, x_rand)
q_new ⇐ NEW-STATE(q_near, u)
x_new ⇐ f(q_new)
if COLLISION-FREE (q_near, q_new) then
    T.add-node(q_near, q_new, x_near, u)
end if
```

An action, $\mathbf{u}$ is then derived in the CONTROL() function. Finally a new state, $\mathbf{q}_{new}$, is integrated in the function NEW-STATE() from $\mathbf{q}_{near}$ when control $\mathbf{u}$ is applied for some Δt. Ideally, the control action is computed such that $\mathbf{q}_{new}$ is closer to $\mathbf{q}_{rand}$ than $\mathbf{q}_{near}$. The new state is then added to the tree, while keeping track of the action and parent node.

The RRT works well because the search has a Voronoi bias, meaning that nodes in the tree which are closest to the largest regions of unexplored space are most likely to be chosen for expansion into these regions. Thus, RRT's have a tendency to quickly branch into unexplored regions. When the unexplored regions become smaller and more uniformly distributed, then the search begins to fill in gaps with increasingly uniform coverage, ensuring that the algorithm is probabilistically complete, meaning that it will find a path if one exists as K goes to infinity.

B. TS-RRT

The proposed planner, given in Algorithm 3, is called Task-Space RRT (TS-RRT). It is similar to the standard RRT approach, except that a random sample, $\mathbf{x}_{rand}$ is generated directly in Task Space, typically using a uniform distribution over a bounded region. The TS-EXTEND() function, shown in Algorithm 4, is similar to the standard RRT, but works in Task Space rather than C-Space. The TS-NEAREST-NEIGHBOR() function selects the nearest point on the tree in Task Space. Each node on the tree contains both C-Space and Task Space information, so the computation of TS-NEAREST-NEIGHBOR() would be less than the comparable function in a standard RRT. A full configuration space action is generated in TS-CONTROL(). Ideally, this control attempts to grow the nearest node of the tree towards the random sample in Task Space. There are many ways of implementing this type of control, and the action can even be random, as done in [7]. A more efficient approach is to use the Jacobian pseudoinverse in a feedback type controller, as shown in Algorithm 5). A new state in C-Space, $\mathbf{q}_{new}$ is integrated using the same NEW-STATE() function as in the RRT. After checking for collisions in the full C-Space, $\mathbf{q}_{new}$, its Task Space mapping $\mathbf{x}_{new}$, and the control action are added to the tree.

Because the sampling occurs in Task Space, the Voronoi bias when growing the tree is now in this space, rather than in the configuration space. This encourages the planner to quickly explore empty regions within task space, which results in more efficient planning, assuming paths in this space exist and there are no narrow channels, using the selected Task Space mapping. Approaches for ensuring completeness are discussed in section IV.

The growth mechanism in Algorithm 5 allows for a secondary command, $\dot{\mathbf{q}}_{ref}$, to be defined in terms of the configuration (state) space. If left as zero, then this controller will simply take the shortest path within the configuration (state) space. However, this function can be used to encourage the system to go toward certain configurations as much as possible, and can be similar in spirit to the workspace centering (e.g. [15]), used in the more typical feedback based task-space control.

III. Scalability of TS-RRT compared to RRT

A. Path Planning for the N-Link Arm

In this section, we examine a kinematic search problem, in which an N-link arm needs to find a suitable trajectory from a given start pose, with the task of putting its end effector at a particular coordinate. Figure 2 demonstrates a 5-link version of this problem. The task space utilized is the Cartesian coordinate of the end effector. This path planning problem does not have dynamics, and therefore does not have any second-order effects including gravity and torque limits. However, the joints are limited to move less than .05 radians per each Δt, and obstacles are imposed within the Cartesian plane, which are checked for collisions against all links.

Algorithm 5 Example TS-CONTROL($q_{near}, x_{near}, x_{rand}$)

1: $\mathbf{u}_{ts} \Leftarrow (\mathbf{x}_{rand} - \mathbf{x}_{near})/\Delta t$
2: $\mathbf{u}_{ts} \Leftarrow$ CROP-WITHIN-LIMITS($\mathbf{u}_{ts}$)
3: $\mathbf{J} \Leftarrow$ COMPUTE-JACOBIAN($\mathbf{q}_{near}$)
4: $\dot{\mathbf{q}}_{ref} =$ SECONDARY-CONTROL($\mathbf{q}_{near}, \mathbf{x}_{rand}$)
5: $\dot{\mathbf{q}} \Leftarrow J^+ \cdot \mathbf{u}_{ts} + \alpha(\mathbf{I} - \mathbf{J}^+ \cdot \mathbf{J}) \cdot \dot{\mathbf{q}}_{ref}$
6: **return** $\Delta\mathbf{q} \Leftarrow \dot{\mathbf{q}} \cdot \Delta t$

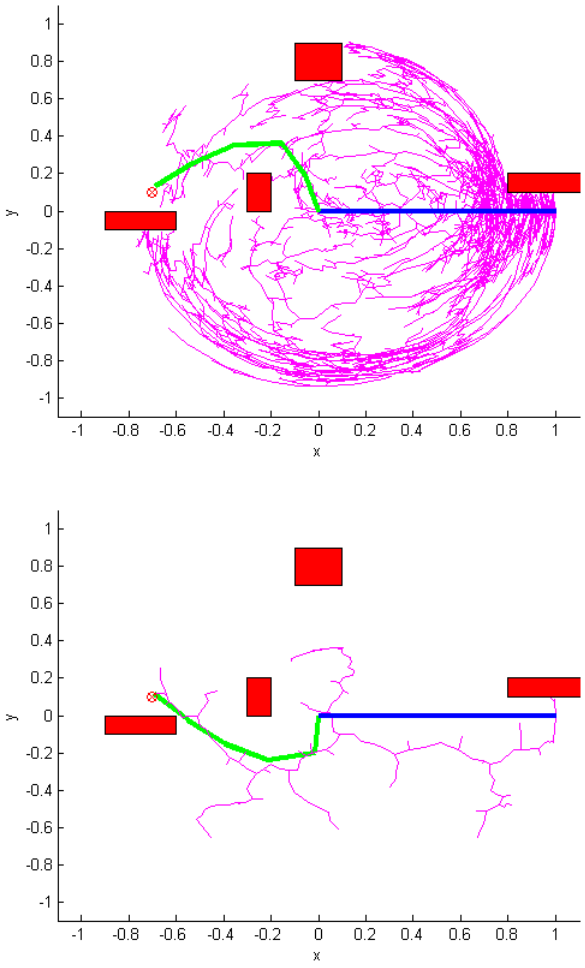

Fig. 2. Projection of RRT into End Effector Cartesian Coordinates for a 5-link arm (shown in Magenta). The starting configuration ($[0, 0, 0, 0, 0]^T$) is shown in Blue, and the achieved goal configuration is shown in green. **TOP**: Standard RRT search was performed in the 5-D configuration space. This tree has 2000 nodes. Obstacles in the workspace are shown in red, with collision checking performed for 10 evenly spaced points along each link. **BOTTOM**: the resulting RRT using TS-RRT is shown, where exploration is constrained to the 2D task space. This tree has 150 nodes.

Furthermore, joint limits of +/- 2.5 radians were imposed on all links. This problem was explored with a constant set of obstacles, shown in red in Figure 2. A constant start pose (all angles = 0), and a constant desired goal pose were used. The number of links, N, is variable, with the link length set to 1/N so that the total arm length is always 1.

Two algorithms were explored on this problem. First, a standard forward RRT was implemented. Because there are

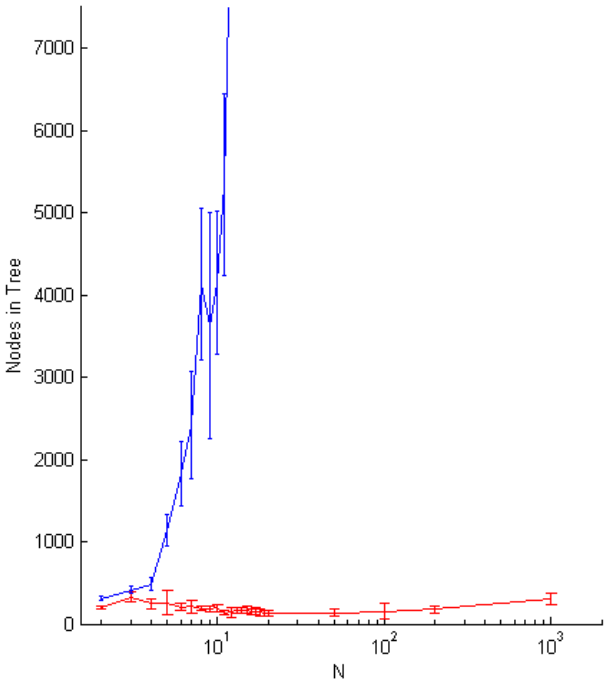

Fig. 3. The number of nodes that the RRT algorithm searches before finding a feasible trajectory for the problem shown in figure 3 is shown as a function of N, the number of dimensions. The obstacles and goal task are held constant, and the link lengths are set to 1/N so that the total arm length remains constant. Note that the **x axis is on a log scale**, starting with N=2, going up to N=1000. **RED**: TS-RRT, which remains approximately constant. **BLUE**: full configuration space RRT, which grows super-linearly.

infinite potential goal poses which correspond to the goal task in this problem, the configuration space was randomly sampled until 20 poses were found that were collision free and whose end effector positions were close to the desired position. The RRT Algorithm 1 was then run, with samples drawn uniformly in configuration space ($\in \Re^N$ with bounds of +/- pi). With probability P=.1, one of the 20 goal configurations was chosen at random to be used as the sample to grow the tree toward. A Euclidean metric (all joints weighted equally) was used in NEAREST-NEIGHBOR. Node growth occurred by computing the vector between a sample and the closest node on the tree, and cropping the components of this vector to be within +/- .05 radians.

In addition to the standard RRT, a forward TS-RRT was run on the same problem. The Jacobian pseudoinverse was used to calculate a desirable change in q to grow a selected node toward a sample in task space. The nullspace reference command utilized was set to $\Delta\mathbf{q}_{ref} = -q$, which encourages the arm to straighten when possible. $\Delta\mathbf{q}$ was normalized so that the max norm was .05. Random samples were taken uniformly in task space ($\in \Re^2$ with bounds of +/- 1.1), and a Euclidean distance metric was used in task space to select the closest nodes on the tree. With probability .1 the goal task was chosen as the sample. Most of the code between the two implementations was shared, with an attempt to minimize

tuning toward each problem / implementation.

B. Results

Each RRT was run 20 times, while varying the number of links (N) in the arm from 2 to 1000 (TS-RRT), and from 2 to 15 (RRT). The median number of nodes explored by each tree is shown in Figure 3, as a function of N. The standard RRT quickly started exploring thousands of nodes as N increased above 4, while the TS-RRT exploration remained approximately constant. Of course, the Jacobian pseudoinverse and nullspace calculations are expensive to compute, but in our implementation, using standard Matlab functions, they added an overhead of 50 percent to the time needed to grow a node, including collision checking. As an example of the actual time to compute paths, the TS-RRT algorithm took 4-8 seconds with N = 200 (approximately 200 nodes), which was comparable to the standard RRT when using N = 6 (approximately 1500 nodes). Note, the time difference per node is due mainly to increased computation for collision checking in higher dimensions, rather than to the Pseudoinverse computation.

A comparison of an example of trees produced by each algorithm, with N=5, projected into task space, is shown in Figure 2. It is clear that the TS-RRT searches more directly and evenly within task space. A comparison of the trees produced, shown in the full configuration space (for N=3), is shown in Figure 4. The figure shows two different views of the same two trees, the blue corresponding to TS-RRT, and the magenta being the standard RRT. The rectangular obstacles in task space correspond to large cylindrical (pancake-like) obstacles in configuration space, shown by red dots in the figure. The goal task point is denoted by green dots in the figure. The standard RRT produces a tree which seems to evenly spread out in the 3D configuration space, as expected. The TS-RRT, on the other hand, is more surface-like, but is able to navigate the obstacles.

The standard RRT could only run up to N=15 in a reasonable time (approx. 20 minutes to solve). However, we were able to find trajectories with N = 1500 in less than a minute with TS-RRT. Two example trajectories (and trees) are shown in Figure 5. In one trajectory, the arm bends, and then changes the direction of the bend part way through. In the other trajectory (bottom), the arm actually tied a knot through itself in order to reach the goal.

We should note that in order to handle singularities, we simply restricted the maximum magnitude of angular displacement, $\Delta\mathbf{q}$ for any child node. One can see that the starting pose is singular, but because the planner is sample based it tends to immediately break out of any singular regions. More acute examples of the influence of singular regions can be seen at the extremes of the workspace in Figure 5. Towards the edges of the reachable workspace, the TS-RRT becomes slightly irregular, with some nodes being selected repeatedly and expansion restricted. More clever handling of singularities is possible as is done in standard feeback control, but was not required for the task presented here as the system performance was still quite good.

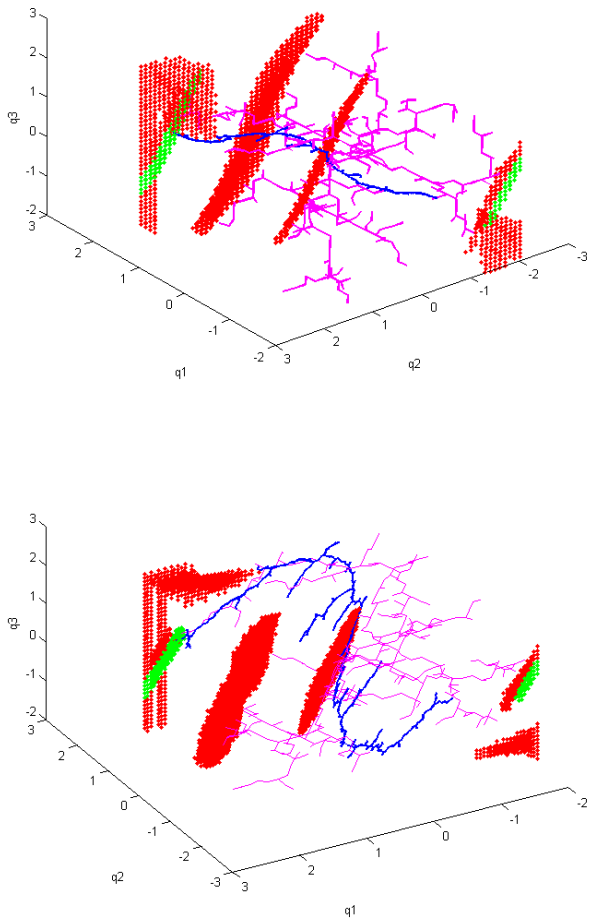

Fig. 4. **TOP/BOTTOM** are two rotated views of the same 3D plot. This figure shows the configuration space view of an RRT tree (magenta), and a TS-RRT (blue), both for a 3-link arm. Obstacles (same as those shown in Figure 2) are shown by red dots, and the goal task point is shown by green dots in the configuration space. The standard RRT has good coverage in the space, as expected. Both trees converge to a collision free trajectory, but it clear that the TS-RRT is much more direct.

IV. Extensions and Possible Modifications

The algorithm is amenable to various modifications, and problem specific implementations.

A. Dynamic and Underactuated Systems

It is possible to extend the TS-RRT algorithm to dynamic systems where new states are generated by integrating the dynamics forward. The torque (or action) used can be computed once per step, or incrementally, where $\mathbf{u}_{ts}$ is recomputed inside the integration loop as $\mathbf{x}_{near}$ changes. Algorithm 5 utilized a pseudoinverse Jacobian based velocity controller, but any preferred method for computing velocity or acceleration based inverse kinematics ([16] for review, [17] for an example applied to a 12-DOF quadruped) can be utilized and integrated forward to generate a new state given a desired point in task space.

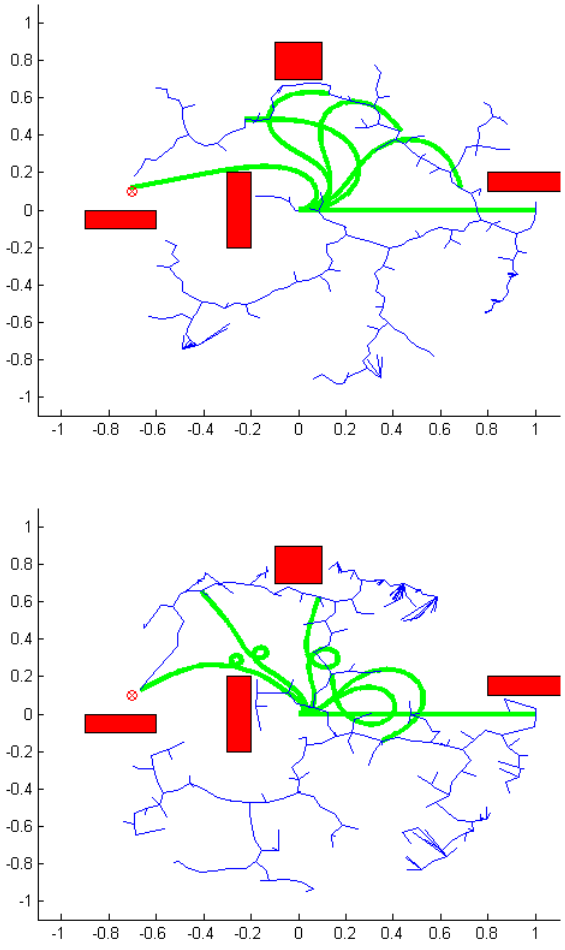

Fig. 5. Two example trajectories and RRT plans, found with N = 1500. Total computation time was less than one minute.

This method may also be extended to control underactuated systems, if a carefully chosen task space is used, as in [18], which demonstrated swing-up of a 5-link pendulum with up to 3 passive joints, by controlling the center of mass of the system. Potential applications for planning in high dimensional dynamic systems are far reaching, and because the task-space control allows for hierarchical control (e.g. the secondary control $\dot{\mathbf{q}}_{ref}$ is only applied in the nullspace of $\dot{\mathbf{x}}_{ref}$, but $\dot{\mathbf{q}}_{ref}$ may have been computed from another task space with a nullspace, allowing for a tertiary controller, and so on). An example of this might be planning humanoid grasping movement, by using the Cartesian coordinate of the hand as a primary task, the orientation as a secondary task, and a desirable grip posture (e.g. to leave fingers open) as a a tertiary task. Or, for walking problems, [17], [19] have proposed to use Center of Mass control as a primary task, with swing-foot position as a secondary task.

B. On Completeness

The obvious drawback to exploring directly in task space is that the planner may lose completeness, in the sense that

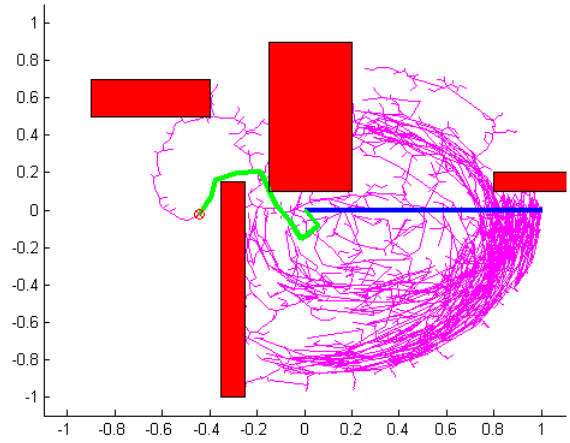

Fig. 6. With more challenging obstacles, a hybrid approach can be used to search efficiently by exploring directly in task space part of the time, and directly in configuration space the remaining time.

it may no longer be possible to guarantee that a path will be found, given that the set of trajectories which can be explored has been severely restricted. Thus the choice of task space is critical, as is the choice of redundancy resolution (e.g. selection of $\dot{\mathbf{q}}_{ref}$). If a task space can encode actions with enough breadth such that feasible solutions exist, it remains unnecessary to explore in the full state space. If a solution exists within the task space, given the redundancy resolution mechanism employed, then this planner will still remain probabilistically complete. Additionally, depending on the redundancy resolution mechanism, taking an action (in task space) in one direction and then taking the reverse action can have the effect that the system returns to the original point in task space, but to a different point in state space. As such, it may be possible that all points in state space are reachable with some action sequence, implying that the proposed algorithm is still probabilistically complete.

The TS-RRT algorithm presented can also be modified so that samples are chosen in configuration space, as in the standard RRT. The standard NEAREST-NEIGHBOR function would be used, so that a configuration space metric is used to evaluate and find the closest node on the tree. The remainder of the algorithm is the same, where a node chosen for expansion grows toward the sample in task space. Note, it is possible to augment or define the $\dot{\mathbf{q}}_{ref}$ term in Algorithm 5, the secondary task in the control function, with: $(\mathbf{q}_{rand} - \mathbf{q}_{near})/\Delta t$. In such a case the system will try to move toward the sample in configuration space, making the algorithm more similar to the standard RRT. By tuning α, growth can be biased to explore more in task space, or to explore more in configuration space, like the standard RRT. However, it is important to realize that a drawback to such an approach is that the Voronoi bias within task space will be disrupted by the sampling within configuration space, because uniform sampling in one space usually does not map to a uniform sampling in the other space.

C. Hybrid Approach

Because book-keeping is virtually the same for both the RRT and TS-RRT algorithms, they are quite interchangeable, and in fact, we can combine both approaches by searching in configuration space with some probability, while searching in task space the remaining time. A variation of may be to search in task space until it is detected that the tree is not expanding, at which point the algorithm can start to sample in configuration space. Any sampling in C-Space ensures probabilistic completeness, while sampling in T-Space allows for faster, more direct exploration when possible. Figure 6 demonstrates the resulting RRT which used such a hybrid approach. Nodes were expanded using either RRT or with TS-RRT with a probability P = .5, to solve a difficult trajectory problem for a 10 link arm with obstacles that define very narrow passable channels. The solution in this configuration is typically found after exploring fewer than 5000 nodes. Interestingly, we were unable to find a trajectory for this problem setup if using exclusively either the standard-RRT or the non-augmented TS-RRT, even after 200,000 nodes of exploration.

In the worst case, if all extend operations in C-space are inefficient (approximately perpendicular to all solution trajectories), then the maximum slowdown of this approach is a factor of $< 1/P$ compared to searching in T-space alone. However, this may be a small price to pay in order to achieve probabilistic completeness. Generally, the more challenging the problem, defined by the degree of complexity of obstacles (e.g. narrow obstacles), the more it will benefit to explore more often in C-space. We leave a more thorough analysis of such a hybrid approach to future work.

V. Concluding Remarks

Task Space Control [5] is widely used in feedback control, but has been widely neglected in the planning community. We have shown that for some classes of problems, a carefully chosen task space may offer a tremendous improvement in planning performance because a Voronoi bias encourages exploration of empty regions in task space, rather than in the full configuration space, which allows for solutions to be found more directly. It is not unreasonable to assume that problems that can be solved this way in low dimensions can be scaled to higher dimensions in approximately linear time, as the number of nodes needed to find a path stays approximately constant with N, and only the time to check collisions and to compute the Jacobian pseudoinverse increases. Although this paper was specifically targeted toward modifying the RRT algorithm in particular, the concept of task-space exploration may be a powerful tool to enable a variety of planning algorithms to work in higher dimensions. Furthermore, the applications are numerous, especially given that 1) many real world systems are high dimensional and 2) It is often natural to specify low dimensional tasks for many problems.

Acknowledgment

This work was supported by the DARPA Learning Locomotion program (AFRL contract # FA8650-05-C-7262). The authors would like to thank Thomas Kollar for his suggestion to explore the N-Link arm.

References

[1] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[2] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Iowa State University, Dept. of Computer Science, Tech. Rep. 98–11, 1998.

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

[4] T. Lozano-Perez, M. T. Mason, and R. H. Taylor, "Automatic synthesis of fine-motion strategies for robots," *International Journal of Robotics Research*, vol. 3, no. 1, pp. 1–34, December 1983.

[5] A. Liegeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-7, no. 12, pp. 868 – 871, December 1977.

[6] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, "Motion planning for humanoid robots under obstacle and dynamic balance constraints," *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 1, pp. 692–698 vol.1, 2001.

[7] Bertram, D., Kuffner, J., Dillmann, R., Asfour, and T., "An integrated approach to inverse kinematics and path planning for redundant manipulators," *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 1874–1879, 15-19, 2006.

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

[9] R. Diankov, N. Ratliff, D. Ferguson, S. Srinivasa, and J. Kuffner, "Bispace planning: Concurrent multi-space exploration," in *Proceedings of Robotics: Science and Systems IV*, June 2008.

[10] J. J. Kuffner, J. Steven, and M. Lavalle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000, pp. 995–1001.

[11] X. Tang, S. Thomas, and N. M. Amato, "Efficient planning of spatially constrained robot using reachable distances," Texas A&M University, Tech. Rep. TR07-001, September 2006.

[12] Z. Yao, Gupta, and K., "Path planning with general end-effector constraints: using task space to guide configuration space search," *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 1875–1880, Aug. 2005.

[13] M. Stilman, "Task constrained motion planning in robot joint space," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 3074–3081.

[14] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, August 1996.

[15] L. Sentis and O. Khatib, "Control of free-floating humanoid robots through task prioritization," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

[16] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Comparative experimental evaluations of task space control with redundancy resolution," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005.

[17] A. Shkolnik and R. Tedrake, "Inverse kinematics for a point-foot quadruped robot with dynamic redundancy resolution," in *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, April 2007.

[18] A. Shkolnik and R. Tedrake, "High-dimensional underactuated motion planning via task space control," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2008.

[19] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiware, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *ICRA IEEE International Conference on Robotics and Automation*. IEEE, Sep 2003, pp. 1620–1626.