Randomized Path Planning for Redundant Manipulators without Inverse Kinematics

Mike Vande Weghe
Institute for Complex Engineered Systems
Carnegie Mellon University
Pittsburgh, PA 15213
vandeweg@cmu.edu

Dave Ferguson, Siddhartha S. Srinivasa
Intel Research Pittsburgh
4720 Forbes Avenue
Pittsburgh, PA 15213
{dave.ferguson, siddhartha.srinivasa}@intel.com

***Abstract*— We present a sampling-based path planning algorithm capable of efficiently generating solutions for high-dimensional manipulation problems involving challenging inverse kinematics and complex obstacles. Our algorithm extends the Rapidly-exploring Random Tree (RRT) algorithm to cope with goals that are specified in a subspace of the manipulator configuration space through which the search tree is being grown. Underspecified goals occur naturally in arm planning, where the final end effector position is crucial but the configuration of the rest of the arm is not. To achieve this, the algorithm bootstraps an optimal local controller based on the transpose of the Jacobian to a global RRT search. The resulting approach, known as *Jacobian Transpose-directed Rapidly Exploring Random Trees* (JT-RRTs), is able to combine the configuration space exploration of RRTs with a workspace goal bias to produce direct paths through complex environments extremely efficiently, without the need for any inverse kinematics. We compare our algorithm to a recently-developed competing approach and provide results from both simulation and a 7 degree-of-freedom robotic arm.**

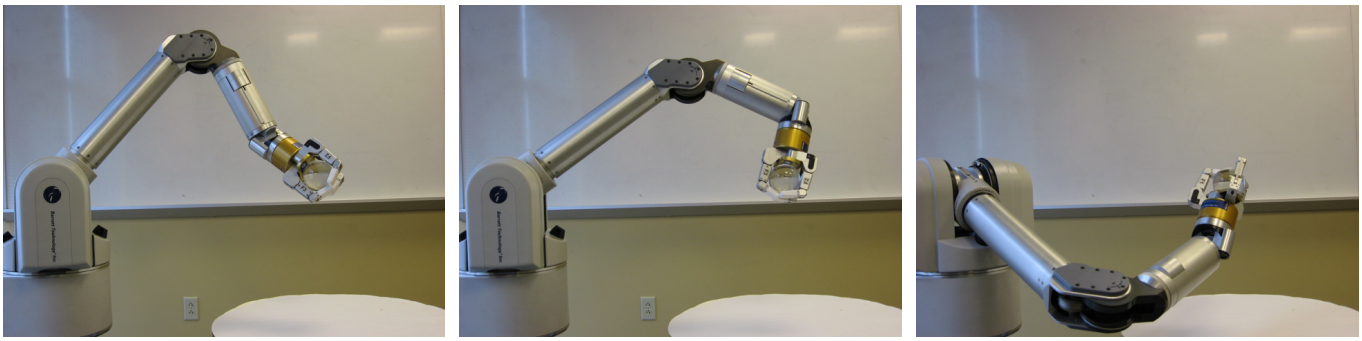
I. Introduction

Path planning for robotic systems operating in real environments is hard. Not only must such systems deal with the standard planning challenges of potentially high-dimensional and complex search spaces, but they must also cope with imperfect information regarding their surroundings and perhaps their tasks, dynamic environments, and limited deliberation time. As such, planning algorithms used by these systems must be extremely efficient to generate solutions that can be executed quickly while they are still applicable.

In response to these demands, researchers have developed sampling-based algorithms that rapidly generate solutions in very high-dimensional search spaces. One of the most widely-used of these algorithms is the Rapidly-exploring Random Tree (RRT) algorithm [1]. This algorithm grows a search tree out from an initial position in the search space (the initial configuration) and uses random sampling of the search space to bias the growth of this tree towards unexplored regions. Consequently, it explores the space extremely efficiently. Because randomization is used to grow the tree, the algorithm copes well with both very high-dimensional search spaces and very large branching factors. Further, the RRT algorithm can bias its search towards a particular goal configuration, which significantly improves its efficiency in generating a solution to a given planning problem.

Fig. 1. The Barrett Whole Arm Manipulator (WAM) used for manipulation in populated indoor environments.

Because of their ability to solve very complex, high-dimensional planning problems and their relative ease of implementation, RRTs have been used in a huge range of motion planning scenarios [1], [2], [3], [4], [5]. In particular, they have been used for high-dimensional manipulation planning [6], which is our current focus. In this scenario, the aim is to generate a trajectory for a manipulator to take it from some initial configuration to a desired goal. However, in contrast to many of the other domains in which RRTs have been used, in manipulator planning the goal is usually not specified as a single desired joint configuration, but rather a desired location in workspace of the end effector of the arm. For example, if we are trying to solve a grasping problem in which we want our robotic arm to pick up a cup for us from the sink, we are trying to figure out how it can plan to have its hand (end effector)



(a) (b) (c)

Fig. 2. Redundant Manipulators can have an infinite number of configurations corresponding to the same end-effector pose in space. Figures (a) through (c) show three different configurations of our WAM that result in the same hand pose.

make particular contact with the cup. Unfortunately, if we are dealing with a redundant manipulator, then this end effector workspace goal corresponds to a potentially infinite number of configuration space goals for the manipulator. Further, no closed-form solution exists for solving the mapping from workspace goal to configuration space goal(s) for complex manipulators.

As a result, the standard RRT algorithm does not perform exceptionally well in this problem domain. Instead, other approaches have been developed that attempt to directly address this inverse kinematics challenge. However, all of these approaches have limitations, as we discuss in the following section.

In this paper, we present an extension to the RRT algorithm that is able to overcome the problem of inverse kinematics by exploiting the nature of the Jacobian as a transformation from configuration space to workspace. The resulting algorithm is able to harness the power of the RRT algorithm for exploring very high-dimensional spaces while also being able to focus its search towards a desired goal in workspace.

We begin by discussing the problem of inverse kinematics in redundant manipulator planning and describe existing planning algorithms that attempt to deal with this problem. In particular, we discuss a recent algorithm by Bertram et al. [6] that does a nice job of extending the RRT algorithm to operate without requiring a solution to the inverse kinematics of the manipulator. We then describe an analytical method for producing a configuration-space trajectory that follows a workspace path. We use this method to develop an RRT-based algorithm that is able to efficiently focus its search towards a goal specified in workspace, while still exploring configuration space. We provide a number of results comparing our algorithm to the current state of the art in manipulation planning and illustrate its effectiveness in both simulation and on a physical 7 DOF manipulator arm.

II. Redundant Manipulator Planning

The standard path planning problem can be formulated as a search for a path from some initial configuration q_{start} of a system to some desired goal configuration q_{goal}. The *configuration space* through which this path is searched for represents the set of all possible states or permutations of the system. In robotic manipulation, where we have a robotic system comprised of several links connected to each other through various joints (see Figure 1 for an example such robot), the configuration space corresponds to all the different shapes the arm can make in space. Each of these shapes is formed by a unique set of joint angles (and/or joint offsets for robots with prismatic joints), and the configuration space is thus exponential in the number of joints contained in the manipulator. In such cases, a configuration of the system is the set of angles for the joints for a particular shape.

Because the configuration space for a manipulator with several joints can be extremely vast and high-dimensional, classical planning techniques based on discretizing the configuration space and then deterministically searching through this space (e.g. Dijkstra's search [7] or A* [8]) are usually far too memory and computation intensive to generate solutions to manipulator path planning problems. Instead, sampling-based planning techniques have shown themselves to be very well suited to this class of planning problem. Perhaps the most popular sampling-based algorithm in robotics is the Rapidly-exploring Random Tree (RRT) algorithm [1].

RRTs search for paths through configuration space by growing a search tree from the initial configuration q_{start} and trying to connect this tree to the goal configuration q_{goal}. To do this, they randomly sample points in the full configuration space and then attempt to extend the search tree out towards these points. As a result, the growth of the tree is biased towards previously unvisited regions of the configuration space and exploration occurs very quickly.

As mentioned earlier, they can also be made far more efficient by focusing their growth more directly towards the goal. To do this, rather than randomly sampling points to extend towards at every iteration of the algorithm, they occasionally (with some probability p_g) select the goal configuration as the point to extend the tree towards. This has the effect of pulling the tree in the direction of the goal and usually produces solutions for significantly less computation. This goal bias is a very important feature of the RRT algorithm as it improves the chances of reaching within a desired tolerance of the goal without needing to explore the entire configuration space to

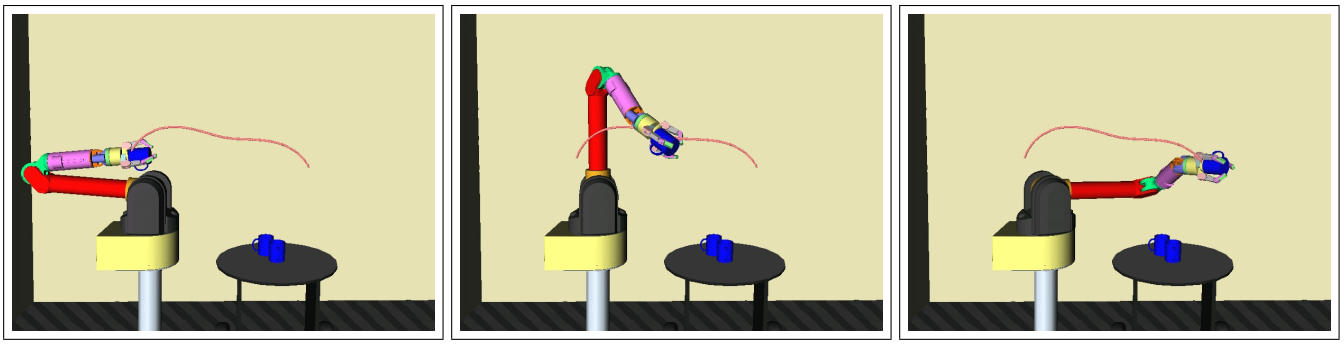


Fig. 3. The Jacobian transpose provides a mapping from joint space to workspace that enables the manipulator to approximately traverse a workspace path. This mapping can be used to extend our configuration space search tree towards a goal specified in workspace.

that tolerance.

For most robotic path planning problems, constructing this desired goal configuration q_{goal} is straightforward: we know where we would like the robotic system to end up, and we can figure out where that location corresponds to in the robot's configuration space. However, when dealing with redundant manipulators, such as our 7 link robotic arm, solving for this mapping from workspace to joint space known as the manipulator's *inverse kinematics* is extremely challenging. In fact, for manipulators with more than six links there is no closed-form solution to this problem [9], and even for manipulators with six or fewer links there may be an infinite number of configuration space states that result in the same workspace state.

Researchers have investigated various ways of getting around this problem. Classically, the most common approaches use numerical approximation to compute a solution to the inverse kinematics (IK) problem [10]. However, as described in [6], these approaches can fail to converge to a valid solution and are usually limited to finding only one of the potentially infinite number of solutions. This can be particularly disadvantageous for manipulator planning in environments containing obstacles because it is likely that the single configuration space goal returned may be unreachable. Even if no obstacles are present, this goal still may not be feasible given the limitations of the mechanism.

To overcome the limitations of these numerical approximation-based techniques, Bertram et al. [6] developed a nice extension to the RRT algorithm that removes the need for an inverse kinematics solution and can handle a goal specified in the workspace of the manipulator end effector. In their approach, rather than selecting the goal configuration as the sample point to extend the tree towards (with some probability p_g), they select the configuration in the search tree that is closest to the workspace goal *using a workspace distance metric* and then extend out from this configuration in a random direction. This algorithm has a number of nice properties relative to the numerical IK approximations and standard RRT approach. First, no explicit inverse kinematics is required for planning. Secondly, all configurations reached during the search are valid, so there is no problem of planning to invalid goal configurations based on inaccurate inverse kinematics. Finally, because the workspace goal is used to influence the growth of the tree, convergence is typically much faster than with the standard RRT algorithm.

The algorithm we present in the following sections shares the same motivation as that of Bertram et al.'s, namely, it attempts to remove the need for explicit solutions to the inverse kinematics problem without sacrificing the efficiency of the resulting planning process. However, our approach tries to exploit the workspace goal even further to improve convergence. Rather than randomly extending the search tree out from the node closest to the workspace goal, it computes the best extension from this node towards the goal. In the following section we describe how this extension can be calculated.

III. Exploiting the Jacobian

Given a robot arm configuration $q \in \mathcal{Q}$ and a desired end-effector goal $x_g \in \mathcal{X}$, where $\mathcal{X}$ is the space of end-effector positions $\mathbb{R}^3$ (or poses $SE(3)$), we are interested in computing an extension in configuration space from q towards x_g. Unfortunately, the mapping from $\mathcal{Q}$ to $\mathcal{X}$ is usually nonlinear and very expensive to compute. However, its derivative, called the Jacobian, is a linear map from the tangent space of $\mathcal{Q}$ to that of $\mathcal{X}$, is expressed as $J\dot{q} = \dot{x}$, where $x \in \mathcal{X}$ is the end-effector position (or pose) corresponding to q, and can be computed quickly.

Ideally, to drive the end-effector to a desired configuration x_g, we could compute the error $e = (x_g - x)$ and run a controller of the form $\dot{q} = kJ^{-1}e$, where k is a positive gain. In the absence of any obstacles, internal collisions, or joint limits, this simple controller is guaranteed to reach the goal. Unfortunately, in the absence of a closed form solution, the computation of the inverse of the Jacobian must be done numerically at each time step. There have been several numerical methods suggested for computing the inverse, ranging from efficient coordinate descent techniques[11], [12], [13], [14], to function approximators which learn an approximation of the inverse mapping[15], [16]. These techniques are, however, orders of magnitude slower than just computing the Jacobian.

An alternate approach, first presented in [17], is to use the transpose of the Jacobian instead of the inverse. This results in a control law of the form $\dot{q} = kJ^{T}e$. The controller eliminates

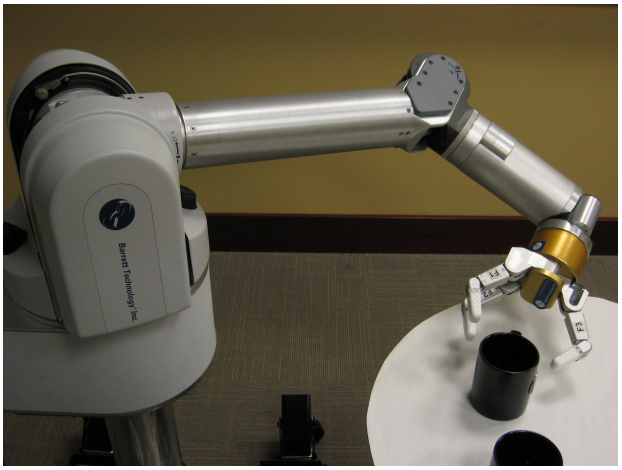


Fig. 4. The WAM hitting 6 of the 7 joint limits reaching for a mug. The joint limits cause the Jacobian Transpose controller to get stuck in a local minimum.

the large overhead of computing the inverse by using the easy-to-compute Jacobian instead. It is easy to show that, under the same obstacle-free requirements as the Jacobian inverse controller, the Jacobian transpose (JT) controller is also guaranteed to reach the goal. A rigorous proof is given in [17], but the intuition is as follows. The instantaneous motion of the end effector is given by $\dot{x} = J\dot{q} = J(kJ^T e)$. The inner product of this instantaneous motion with the error vector is given by $e^T \dot{x} = ke^T JJ^T e \geq 0$. Since this is always positive, under our assumptions about obstacles, the controller is guaranteed to make forward progress towards the goal.

The JT controller makes the crucial assumption that any commanded joint velocity direction is achievable. This assumption is broken in the presence of configuration space obstacles[1]. These obstacles can be in the form of workspace obstacles, self-collisions, or joint limits. Figure 4 provides an example where the Jacobian transpose is trying to direct a manipulator towards a goal and the arm is unable to follow this direction because of its joint limits. These constraints reduce the effectiveness of the approach as a standalone solution for generating trajectories for redundant manipulators. However, the JT control loop is very fast to compute and will provide a direct action for leading the end effector towards its goal. These properties make it an ideal candidate for a local extension operation within a planning algorithm such as RRTs.

IV. Exploring in Configuration Space and Focusing in Workspace

We can use the Jacobian transpose controller to provide a powerful goal directed action within a sampling-based planner such as RRTs. The idea is to replace the configuration space goal bias used in the original RRT algorithm with the Jacobian transpose-based workspace goal bias. To do this, instead of selecting with probability p_g the node closest in configuration space to the configuration space goal g_{goal} and then extending

[1]Non-integrable velocity constraints (nonholonomic constraints) will also break the assumption, but they are less frequently encountered in robot arms.

```
GrowRRT()
2  Q_new = {q_start};
3  while (DistanceToGoal(Q_new) > distanceThreshold)
4    p = RandomReal([0.0, 1.0]);
5    if (p < p_g)
6      Q_new = ExtendTowardsGoal();
7    else
8      Q_new = ExtendRandomly();
9    if (Q_new ≠ ∅)
10     AddNodes(Q_new)
ExtendTowardsGoal()
11   q_old =ClosestNodeToGoal();
12   repeat
13     J^T =JacobianTranspose(q_old);
14     δ_x =WorkspaceDelta(q_old, x_goal);
15     δ_q = J^T · δ_x;
16     q_new = q_old + δ_q;
17     if (CollisionFree(q_old, q_new))
18       Q_new = Q_new ∪ q_new;
19     else
20       return Q_new;
21     q_old = q_new;
22   while (DistanceToGoal(q_new) > distanceTrheshold)
23   return Q_new;
```

Fig. 5. The JT-RRT Algorithm.

from this node towards q_{goal} using configuration space metrics, we select the node closest in *workspace* to the workspace goal x_{goal} and then extend from this node towards x_{goal} using the Jacobian transpose construction.

This allows us to bias the search towards the desired workspace goal while also exploring efficiently through configuration space. The resulting approach shares all of the advantages of Bertram et al.'s approach over previous techniques. Namely, no explicit inverse kinematics is required for planning (nor is the approach limited by a restricted sample set of the IK solutions), all solutions computed are feasible, and its workspace goal bias results in much faster convergence than with standard sampling-based algorithms. However, its ability to use the Jacobian transpose to compute extensions that lead directly towards the goal results in much more efficient goal biasing than the algorithm of Bertram et al., which computes random extensions from the closest nodes. This important difference provides a significant improvement in performance, as shown in the following section.

The Jacobian-transpose directed RRT algorithm is provided in Figure 5. In this pseudocode, the RRT is grown until one of the new nodes added to the tree (i.e. in the set Q_{new}) is within the desired distance threshold of the goal. During the extension stage of the algorithm, with probability p_g the tree is grown towards the goal and with probability $1 - p_g$ it is grown randomly in configuration space (exactly as in the goal-directed RRT algorithm). When growing towards the goal (the **ExtendTowardsGoal** function), the node q_{old} in the tree with the shortest workspace distance to the goal x_{goal} is

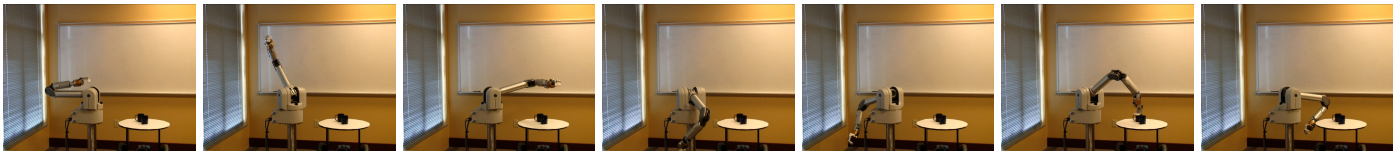


Fig. 6. The example planning problems used in our results. In each case, the manipulator was tasked with planning a path to get its end effector to a desired (x, y, z) location in space. The leftmost image shows the initial arm configuration, the other images show sample arm configurations that satisfied each respective scenario.

Approach	Scenario	Successful	Time (s)	Nodes	Random Extensions	Goal Extensions	Collision Checks	Joint Limits
Bertram et al.	1	50/50	1.542	8583	138	138	22560	N/A
JT-RRTs	1	50/50	0.450	861	28	26	5731	116
Bertram et al.	2	50/50	0.981	5663	90	89	14254	N/A
JT-RRTs	2	50/50	1.761	3189	130	131	23768	532
Bertram et al.	3	31/50	10.926	53074	991	990	140566	N/A
JT-RRTs	3	50/50	0.866	2800	118	120	8671	290
Bertram et al.	4	1/50	5.396	27198	421	416	75753	N/A
JT-RRTs	4	50/50	2.304	6856	322	324	23851	1497
Bertram et al.	5	50/50	1.810	9853	160	157	26133	N/A
JT-RRTs	5	47/50	1.840	7282	72	74	12979	5559
Bertram et al.	6	6/50	16.011	56313	2858	2832	129833	N/A
JT-RRTs	6	50/50	4.152	16854	989	979	36756	1912

TABLE I

RESULTS FROM 7 DOF MANIPULATOR PLANNING

selected and the Jacobian transpose is computed for this node. The workspace vector from the end effector position at q_{old} to x_{goal} is used to calculate a workspace delta δ_x that the Jacobian transpose is multiplied by. Because this gives (only) the instantaneous direction of movement towards the goal it is important that only a small step is taken along this direction before the Jacobian transpose is re-evaluated. The resulting configuration space delta δ_q is added to q_{old} to compute a new configuration q_{new} that resides towards the goal x_{goal} in workspace. If this configuration is reachable from q_{old} without colliding with any configuration space obstacles or violating joint constraints, it is put in the set of nodes Q_{new} to be added to the tree.

Although left out of the pseudocode for clarity, a number of extensions can be made to this basic version of the algorithm for improved performance. In particular, because the Jacobian transpose can cause the manipulator to come up against its joint limits, when computing the new configuration q_{new} it is usually much more efficient to bound the value of each joint by its respective limits (line 16). If all the joints are at their limits then we terminate the extension operation (since we can get no further using the Jacobian transpose). Additionally, when choosing nodes for goal extension (line 11) we avoid nodes that have already been used in an extension step (since the Jacobian transpose will produce the same extension, resulting in duplicate nodes).

V. RESULTS

We compared the performance of the JT-RRT algorithm against Bertram et al.'s RRT extension over a range of different planning scenarios involving our 7 DOF manipulator. In each scenario, the task was to reach a different end-effector (x, y, z) position in space, while avoiding obstacles in the environment. Figure 6 shows the initial configuration of the arm and a sample goal configuration for each scenario. Because one of the motivations of our current research is coordination between a manipulator arm and a mobile robotic Segway, we included a Segway as one of the environmental obstacles, along with cups placed on top of it. The walls, floor, and base of the manipulator are also included in the environment for collision-checking purposes.

Both approaches were implemented in C++ using the OpenRAVE simulator, originally developed at Carnegie Mellon University, and the runtime results are for a Centrino Core2Duo 2.3 GHz processor. For each scenario we ran 50 different planning runs and recorded the number of successful runs (Successful), the total time taken (Time (s)), the number of nodes added to the tree (Nodes), the number of attempted random extensions (Random Extensions), the number of attempted extensions towards the goal (Goal Extensions), the number of collision checks (Collision Checks), and the number of times any of the joint limits were hit during the Jacobian transpose extension operation (Joint Limits). All values are with respect to the successful runs. Averages for all these values are included in Table I. For both approaches we used a goal bias p_g of 0.5 which was shown to be most effective for Bertram et al.'s approach [6]. We did not further tune this value for the JT-RRT approach. Each end-effector goal position needed to be reached within 0.15 meters for the goal threshold to be satisfied.

A sample trajectory from the third results scenario is shown in Figure 7, both in simulation and during execution on our physical manipulator arm.

From the results the JT-RRT algorithm typically requires

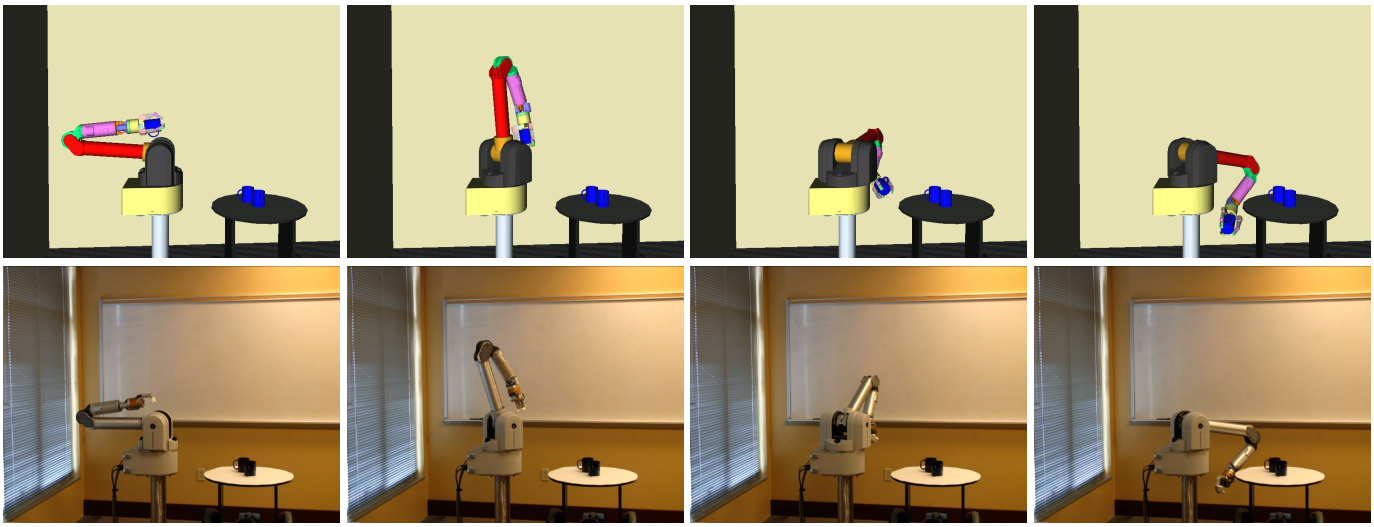


Fig. 7. The JT-RRT algorithm used for manipulation planning for a 7 DOF robotic arm. The top images show the trajectory executed in the openRAVE simulator; the bottom images show the trajectory executed on our robotic platform. This trajectory corresponded to the sixth problem from our set of results.

much less computation and adds many fewer nodes to the search tree, particularly in the more challenging of the scenarios. It is also able to successfully generate solutions for almost every run[2]. For the single scenario where it was not able to produce solutions 100% of the time, it appears that the Jacobian transpose extension operation was coming into conflict with the joint limits of the manipulator (the number of times a joint limit was reached during extension was 5559). This is a side effect of using the Jacobian transpose: as noted earlier, it is oblivious to configuration space obstacles and joint limits. A simple extension to improve this would be to make sure each new node generated during the goal extension step brings the end-effector a non-trivial distance closer to the goal, to make sure the joint limits are not restricting the movement to be along useless directions. An even more promising improvement would be to use the Jacobian transpose as a guide for a more informed local search during goal extension so that these constraints can be taken into account and overcome.

However, in general the performance of the JT-RRT algorithm is very good in terms of both computation time and memory required. The reason that it is typically much more efficient than Bertram et al.'s approach is because it is better able to compute an extension operation that leads towards the desired goal. Rather than selecting the closest node to the goal and then just extending in a random direction, it is able to select this node and then extend directly towards the goal. This greatly improves the goal-directed portion of its search and enables it to satisfy very precise goal thresholds that would otherwise be untenable given random extensions.

VI. Conclusion

We have presented an extension to the RRT algorithm that is able to overcome the problem of inverse kinematics by exploiting the nature of the Jacobian as a transformation from configuration space to workspace. The resulting algorithm is able to harness the power of the RRT algorithm for exploring very high-dimensional configuration spaces while also being able to focus its search towards a desired goal in workspace. We have found it to be very effective for redundant manipulator path planning and have presented results from both simulation and a 7 DOF robotic arm.

We are presently working on a number of extensions to our current framework. Firstly, we are investigating using the Jacobian transpose as a guide for a more informed local search during goal extension operations. We are also looking at using heuristics to create better/faster solutions (as used in the Anytime RRT algorithm [5]) as the quality of standard RRT solutions can vary substantially. Finally, we are combining our arm-level planning with a grasp planner to perform complete manipulation and grasping tasks involving known and unknown objects.

[2]Each approach was limited to adding 100000 nodes to the search tree.

VII. Acknowledgements

The authors would like to express their gratitude to James Kuffner, Ross Diankov, and Dmitry Berenson, for valuable discussions and the development of the openRAVE simulator. Mike Vande Weghe is partially supported by the National Science Foundation under grant 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. 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.
[7] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.
[8] N. Nilsson, *Principles of Artificial Intelligence*. Tioga Publishing Company, 1980.
[9] J. Craig, *Introduction to Robotics: Mechanics and Control*. Addison-Wesley, 1989.
[10] 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.
[11] Y. Nakamura and H. Hanafusa, "Inverse kinematics solutions with singularity robustness for robot manipulator control," *Journal of Systems, Measurement, and Control*, vol. 108, pp. 163–171, 1986.
[12] C. W. Wampler, "Manipulator inverse solutions based on vector formulations and damped least-square methods," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 16, pp. 93–101, 1986.
[13] A. S. Deo and I. D. Walker, "Adaptive nonlinear least-squares for inverse kinematics," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1992.
[14] J. Zhao and N. I. Badler, "Inverse kinematics positioning using nonlinear programming for highly articulated figures," *ACM Transactions on Graphics*, vol. 13, pp. 313–336, 1994.
[15] M. I. Jordan and D. E. Rumelhart, "Forward models - supervised learning with a distal teacher," *Cognitive science*, vol. 16, pp. 307–354, 1992.
[16] A. D'Souza, S. Vijaykumar, and S. Schall, "Learning inverse kinematics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.
[17] W. A. Wolovich and H. Elliott, "A computational technique for inverse kinematics," in *IEEE Conference on Decision and Control*, vol. 23, 1984, pp. 1359–1363.