An Integrated Approach to Inverse Kinematics and Path Planning for Redundant Manipulators

Dominik Bertram[1,2] James Kuffner[2] Ruediger Dillmann[1] Tamim Asfour[1]

[1]*Institute of Computer Science and Engineering*
University of Karlsruhe
Haid-und-Neu-Straße 7, 76131 Karlsruhe, Germany
bertramd@ira.uka.de

[2]*The Robotics Institute*
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA, 15213, USA
{dbertram,kuffner}@cs.cmu.edu

***Abstract*— We propose a novel solution to the problem of inverse kinematics for redundant robotic manipulators for the purposes of goal selection for path planning. We unify the calculation of the goal configuration with searching for a path in order to avoid the uncertainties inherent to selecting goal configurations which may be unreachable because they currently lie in components of the free configuration space disconnected from the initial configuration. We adopt workspace heuristic functions that implicitly define goal regions of the configuration space and guide the extension of Rapidly-exploring Random Trees (RRTs), which are used to search for these regions. The algorithm has successfully been used to efficiently plan reaching and grasping motions for a humanoid robot equipped with redundant manipulator arms.**

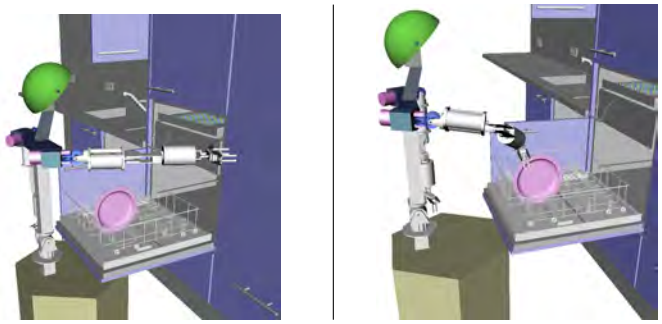

Fig. 1. Motion planning for a 7-DOF manipulator: Taking a plate out of a dishwasher. (Experiment *Dishwasher*)

I. Inverse Kinematics and Path Planning

Service robots and especially humanoid robots are expected to perform complex manipulation tasks in dynamic environments. This precludes the use of preprogrammed trajectories, and instead necessitates general and flexible techniques for autonomous *manipulation planning*. Solving a manipulation planning problem involves computing some sequence of grasping, regrasping, and manipulation operations applied to a set of movable objects [1]–[4].

In this paper, we focus on the *reaching* subtask, which involves computing a trajectory for the manipulator arm to move from some initial configuration to a goal configuration with the end-effector in a position to grasp the object. Reaching subtasks have traditionally been further subdivided into the problems of *grasp selection*, *arm configuration selection*, and *arm trajectory planning*. This division of the computation exists for both historical and practical reasons. Conventional path planning algorithms require a specific goal configuration as input. Thus, a method for calculating the joint angles that correspond to the desired workspace posture of the end-effector is needed. This is the classic *inverse kinematics* (IK) problem, which has a long history in the robotics literature.

Apart from special cases, there currently exist no known analytical methods for solving the inverse kinematics of a general redundant mechanism (greater than six degrees of freedom) [5]. Iterative, numerical techniques based on the calculation of the pseudo-inverse of the Jacobian J^+ [6], [7] are typically used instead. These methods have several drawbacks:

- Iterative methods can suffer from poor performance or non-convergence depending upon the quality of the initial guess, the distribution of singularities in the mechanism configuration space, or a combination of these effects.
- While there usually is a wide range of possible workspace end-effector poses for grasping an object, existing IK algorithms typically require selecting exactly one such pose in order to compute a corresponding arm joint configuration.
- When a given workspace position and orientation admits a continuous range of solutions in the joint space, iterative methods are usually only able to return a single solution, as opposed to multiple solutions or a family of solutions.

The conventional path planning problem formulation involves searching the configuration space (C-space) of a robotic system for a collision-free path that connects a start configuration q_{init} to a goal configuration q_{goal} [8]. For a complete treatment of motion planning, the reader is referred to [9], [10]. Approaches to the path planning problem can be divided into the two classes of *single-query* and *multiple-query* planning. For single-query planning, it is assumed that a single planning problem should be solved quickly, without any pre-processing. The class of multiple-query planning problems encompasses cases where many path planning problems are to be solved in the exact same environment, making extensive pre-processing viable (e.g. [11]). Since humanoids and other service robots are intended to operate in dynamic environments, most motion

planning problems for such robots can be assumed to fit into the class of single-query planning.

When only a limited set of goals (as in [12]) or only a single q_{goal} is computed, the following problems can occur when attempting to plan a path:

1) q_{goal} may not be collision-free with respect to obstacles in the environment.
2) q_{goal} may not represent the best choice available in terms of joint distance or planned path length from the current arm configuration.
3) q_{goal} may be unreachable from the current arm configuration (i.e. no collision-free path exists), causing the planner to fail, even when easily reachable, collision-free alternative inverse kinematic solutions exist.

In section II, we will present an example where these problems occur using the traditional approach to path planning and inverse kinematics.

II. Intuition Regarding Configuration Space

The configuration space of a redundant manipulator exists in a high-dimensional space corresponding to the number of degrees of freedom of the manipulator minus the constraints imposed on the motion of the end-effector.

Figure 2 shows a simple redundant robot arm that has three revolute joints with parallel axes, each with a range of motion of $(-\pi, \pi)$ radians. This system has three degrees of freedom (DOF). Because the end-effector of the robot can only reach positions in a plane, the arm is redundant in this plane. The C-space can be visualized as a cube with edges of length 2π. Figure 2 shows two IK solutions (a) and (b) plotted both in the workspace and in the configuration space for a given position of the end-effector.

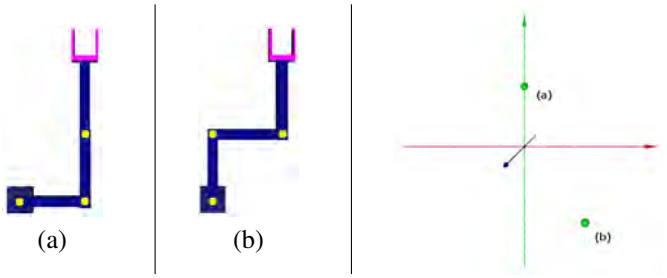

Fig. 2. Two inverse kinematics solutions for the same end-effector posture.

Obstacles in the workspace map to regions in the C-space called C-obstacles, which represent the set of all joint configurations of the robot that cause the geometry of the robot to intersect (overlap) the geometry of the obstacle. These C-obstacles can cause a complete disconnection between different regions of the free configuration space. In this case, there will exist no valid path between two joint configurations lying in two disconnected components of the free space. A simple example for such a C-space is given in Figure 3: The C-obstacle forms a "wall" of complex shape that is parallel to the θ_2-θ_3-axis.

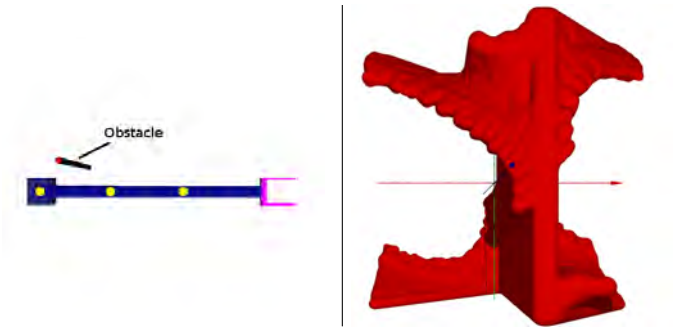

Fig. 3. 3-DOF robot and configuration space with a C-obstacle creating two disconnected components of free space.

In the context of inverse kinematics, this implies that there may be multiple candidate solutions which lie in disconnected components of the free space, and may be *unreachable* from the current initial joint configuration of the robot. Adding the obstacle from Figure 3 to the workspace shown in Figure 2 causes one of the two IK solutions to become disconnected from the initial configuration of the robot as shown in Figure 4. Using a traditional approach to IK, configuration (b) may be selected as the goal. Unfortunately, this blind selection of an IK goal without consideration of its reachability guarantees that the subsequent path planning search will fail.

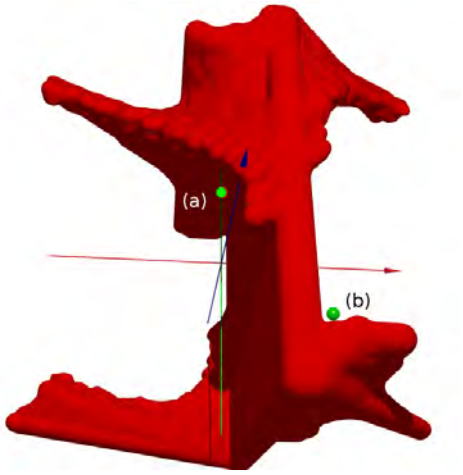

Fig. 4. Two IK solutions (a) and (b) lying in disconnected components of the configuration space.

As mentioned in section I, another problem is that even if it were possible to compute and plan for all possible IK solutions to a single end-effector posture, there may actually be a wide continuous range of end-effector positions which allow solving the task. Consider the task of grasping a cylindrical object. The two configurations shown in Figure 2 are possible solutions but in addition, there are infinitely many more configurations that enable the robot to grasp the object. This set of configurations corresponds to a symmetric region in the C-space illustrated in Figure 5.

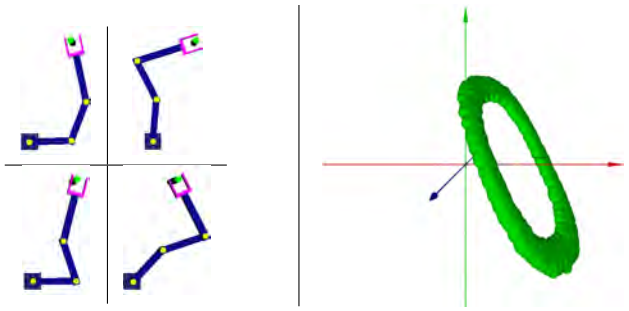

Fig. 5. Solution region for grasping a cylindrical object.

III. An Alternative Method: Integrated Planning and Inverse Kinematics

We propose to avoid these difficulties by integrating the search for inverse kinematics solutions directly into the planning process. Currently, an efficient path planning method based on Rapidly-exploring Random Trees [13] is used to compute collision-free paths. We made the following modifications to the RRT search algorithm [14]:

- No explicit goal configuration is computed. Instead, the planner evaluates workspace goal criteria for configurations generated during the search. This allows for the possibility of discovering any valid goal joint configuration (inverse kinematic solution) that is part of the goal region.
- We grow only a single tree in the configuration space rooted at the current (initial) arm joint configuration, since there is no explicit goal configuration from which a second tree could be grown.
- Appropriate distance metrics and heuristics have been developed for the workspace goal criteria in order to naturally increase the probability of finding solutions that are easily reachable from the current arm configuration.
- Workspace obstacle distance information is used to improve overall performance.

IV. Algorithm

A. Overview

The algorithm consists of several parts which are shown in Figure 6. The robot initial joint configuration q_{init} is given as input to initialize the RRT search tree. For every configuration q added as a node to the RRT, obstacle distance information is used to ensure that the new branch is collision-free.

Progress towards the goal is measured via a heuristic workspace goal function $\Gamma(q) \mapsto \Re$ based on the input workspace goal position and orientation along with any additional grasping constraints. If the goal is reached, then a solution trajectory connecting $q_{init}$and the discovered IK goal configuration is returned.

B. Heuristic Workspace Goal Functions

To implicitly define the region of goal configurations in the C-space, we use a heuristic workspace metric $\Gamma(q) \mapsto \Re$ that maps the pose of the end-effector to a scalar value representing

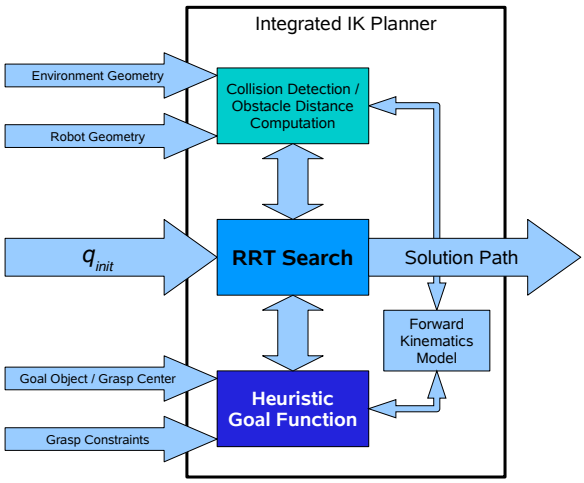

Fig. 6. Overview of Integrated Inverse Kinematics and Planning Algorithm.

proximity to the goal. Forward kinematics is used to compute the relevant information about the end-effectors posture from a configuration q. Our current implementation uses a weighted sum of several factors to achieve a simple characterization of the goal region. The main factor is the difference d of the Euclidean distance between the origin of the end-effector's coordinate system and the center of the target object $\|G-H\|$.

Depending on the target object, different penalty terms are added to constrain the orientation of the end-effector. These terms typically consist of the dot-product of one or more of the coordinate system axes x_H, y_H, z_H and a vector v to which it should be aligned either parallel or vertical, within a tolerance specified by the weight w_1 of the term:

$$w_1 \cdot \big|\,|x_H \cdot v| - 1\big|$$

One common special penalty term aligns the vector $G - H$ to x_H, thus causing the end-effector to point towards the center of the target object. Another possibility is to make $G - H$ vertical to one of the axes of the target object coordinate system x_G, y_G, z_G, thus causing the end-effector to be aligned parallel to one of the coordinate planes:

$$w_2 \cdot \Big(\big|(G-H)\cdot x_H\big| - 1\Big)$$
$$w_3 \cdot \big|(G-H)\cdot y_G\big|$$

Using these components, goal function templates for a variety of target objects, such as cylindrical, box-shaped, spherical and planar shapes, have been defined. For example, a goal function for a spherical object as depicted in Figure 7 might be defined as follows:

$$\phi_{sphere} = \|G-H\| \quad + \quad 50\,\big|(G-H)\cdot x_H - 1\big|$$

To model the goal region, a threshold value g needs to be defined for the goal function. A value below g implies that the given configuration is a part of the goal region.

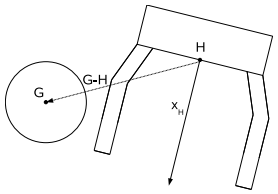

Fig. 7. Grasping a spherical object

While goal functions following the scheme presented above are easily defined and produced good results in our experiments, careful tuning of the penalty weights is necessary to ensure that the resulting function is largely free of local minima. Determining alternative methods for modeling the goal region that are local-minima free is likely to improve the planning algorithm.

C. Search Tree Branch Scaling

In [15], a method is proposed for computing collision-free bubbles around configurations of manipulators with revolute joints. This method essentially gives an upper bound for the distance any point on the geometry of the manipulator can travel for a given change in the configuration. This upper bound is a weighted sum of the differences of the joint angles, with the weights r_i corresponding to the positional change that the i-th joint can effect in the workspace. It can be regarded as a metric $\delta(q_1, q_2) \mapsto \Re$ in the C-space. For a given configuration q, the bubble is then defined as follows:

$$\mathcal{B}(q) = q^* : \delta(q, q^*) \leq d$$

where d is the distance from the manipulator in configuration q to the nearest obstacle in the workspace.

When adding a branch extending from a configuration q to a new configuration q_{new} to the RRT, it has to be guaranteed to be collision-free. To this end, the direction vector representing the branch $q_{new} - q$ has to be scaled so that it lies completely within the collision-free bubble of its parent configuration. A modified scaling method from [16] is used to achieve this.

Let q_d denote the normalized direction vector of the new branch. A scale factor s is needed such that $\delta(q, q_{new}) = d$, with $q_{new} = q + sq_d$. Because

$$\delta(q, q + sq_d) = \sum_{i=1}^{n} r_i |sq_d| \qquad \text{it follows that}$$

$$s = d / \sum_{i=1}^{n} r_i |q_d| \qquad \text{or}$$

$$s = d/\delta(0, q_d)$$

The resulting branch is guaranteed to be collision-free, without the need to run a collision check to confirm. This approach to collision avoidance significantly improves performance. Only in cases where the bubble is so small that a branch inside it would only result in insignificant changes in the end-effector's posture, the branch is extended beyond the boundaries of the bubble and then checked for collision. This is done by enforcing a minimum for the scale factor s:

$$s = \begin{cases} d/\delta(0, q_d) & \text{if } d/\delta(0, q_d) > s_{min} \\ s_{min} & \text{otherwise} \end{cases}$$

D. Search Tree Heuristics

The general extension algorithm of the RRT, as described in [14], is biased towards exploring the empty regions of the C-space. While this is a desirable property, since it guarantees that the tree converges to uniform coverage of the entire space, it is also desirable to partly bias the search towards the goal region. To achieve this, an alternate extension procedure that implements a best-first search strategy was developed. The function *EXTEND* is replaced by the new Procedure `EXTEND_HEURISTIC` in a certain fraction of the extension steps during the search.

Procedure `EXTEND_HEURISTIC`(T, q_{rand})
Extends RRT T from the highest ranked node, in the direction of configuration q_{rand}.

Data: $ranking$: list of configurations, sorted by their chance of reaching the goal region;
f: extension failure threshold
Result: q_{new}: the new configuration;
q_{near}: if extension unsuccessful

$q_{near} \leftarrow ranking$.`front()`;
if `NEW_CONFIG`(q, q_{near}, q_{new}) $\wedge$ (`GOAL_DIST`(q_{new}) $<$ `GOAL_DIST`(q_{near})) **then**
  T.`add_branch`(q_{new});
  return q_{new};
end
`INCREASE_FAILURECOUNT`(q_{near}, 1);
if `FAILURECOUNT`(q_{near}) $> f$ **then**
  $ranking$.`remove`(q_{near});
  $q_{parent} \leftarrow$ `PARENT`(q_{near});
  `INCREASE_FAILURECOUNT`(q_{parent}, f);
end
return q_{near};

Procedure `EXTEND_HEURISTIC` has the following key features:

- A ranking of all nodes in the RRT is created according to their chance of extending into the goal region. The measure used by the ranking is a weighted sum of the goal distance and the distance to the nearest obstacle. The obstacle distance component typically receives a negative weight, since a low goal distance combined with a high clearance is the optimal state for a node in the search tree.
- Instead of selecting a random configuration and trying to extend in its direction from the nearest node of the RRT, the heuristic extension algorithm selects the node with the best ranking and tries to extend in a random

direction from there. The new branch is only added if it achieves a lower goal distance than the parent node. This heuristic implements a best-first search strategy and can therefore get stuck in local minima, e.g. if it continuously tries to directly extend towards a part of the goal region that is obstructed by an obstacle.

- To detect and resolve local minima situations, extension failures are counted for nodes, similar to the method proposed in [17]. When a node exceeds a certain failure threshold, it is removed from the ranking and will therefore not be selected for heuristic extension again. It can, however, still be used in random extension.
- The failure count of a node is incremented when an attempt to add a new branch fails because the new branch is not collision-free or does not yield a lower goal distance than the parent node.
- A node's failure count is set to the maximum when one of its child nodes is removed from the ranking. This rule is necessary to prevent the heuristic from continuously extending into the same local minimum region. Setting the node's failure count to the maximum ensures that it will be removed from the ranking as soon as it produces another failure. This allows nodes that are not in the vicinity of the local minimum to get the best ranking, causing the tree to avoid the region.
- Whenever an extension is made that reduces the goal distance, the tree continues to extend the same direction until the goal distance stops improving. This rule is similar to the RRT-Connect heuristic proposed in [14].

The RRT-Connect search algorithm has been shown to be probabilistically complete in [14]. Since our new algorithm shares the randomized portion of RRT-Connect, it is also probabilistically complete. Because of the randomness of extension directions and the wide variety of situations that can arise in the C-space, it is difficult to make any accurate prediction regarding the rate of convergence. Applying the heuristic extension algorithm to about 50 percent of all extension attempts, however, has been observed to yield good performance for a variety of planning tasks.

V. Experiments

The proposed algorithm was used to solve a number of grasping problems of varying difficulty. All experiments were conducted using a simulation of the humanoid robot ARMAR [18], which is equipped with two 7-DOF manipulator arms, in a kitchen environment. All tests were run 100 times on a *AMD Athlon 2600+* clocked at 2.0 GHz. The *Proximity Query Package* [19] was used for obstacle distance computation.

The first two tasks involve grasping a sphere and a plate floating in the workspace, without any additional obstacles. Note, however, that the target object itself is frequently the most difficult obstacle in a grasping problem. The workspaces of the experiments *Sphere* and *Plate* are shown in Figure 8. The average computation time was 0.4 seconds for grasping the sphere and 2.1 seconds for the plate.

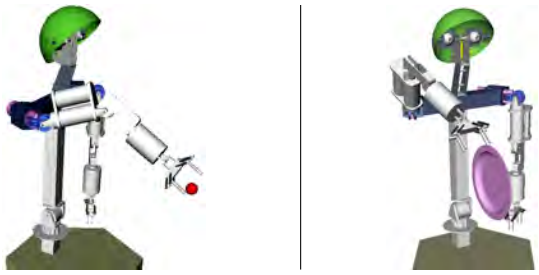

Fig. 8. Experiments *Sphere* and *Plate*

The third workspace, shown in Figure 9, consists of the kitchen environment; the task is to grasp a spherical object in a drawer. The average computation time was 0.7 seconds. The second workspace shown in Figure 9 shows a difficult grasping problem: An object is to be grasped in the confined space of a cabinet. Note that the initial configuration has the end-effector of the robot's arm placed under the cabinet, so that the arm has to fold to get around the cabinet. The average computation time for 100 trials in this experiment was approximately 17.6 seconds.

Another difficult workspace is shown in Figure 1. Here the task is to take a plate out of the dishwasher. The dishwasher basket has highly complex geometry, consisting of about $30,000$ triangles, which makes distance computation expensive. This task was solved in an average of 4.3 seconds. Table I summarizes the results of all of our experiments.

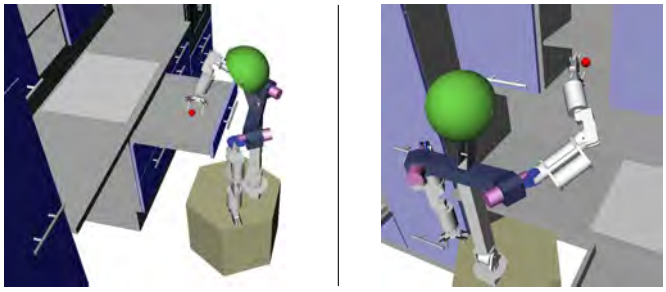

Fig. 9. Experiments *Drawer* and *Cabinet*

Experiment	DOF	triangles	Avg. nodes in tree	Avg. comp. time (seconds)
Sphere	7	5578	71	0.44
Plate	7	7016	268	2.09
Drawer	7	53138	167	0.74
Dishwasher	7	54576	311	4.29
Cabinet	7	53138	2396	17.58

TABLE I
Summary of experiments.

The examples presented and the fact that all test runs converged demonstrate that our new algorithm is already capable of solving typical planning problems with high reliability. When comparing the performance to other planning systems, it is important to keep in mind that our approach solves the

problem of grasp selection in addition to path planning. The computation time needed for inverse kinematics is usually disregarded when analyzing the performance of traditional planners. While such a planner might yield better performance when a reachable IK solution is known, this knowledge is unnecessary in our approach.

VI. Conclusion

In the preceding we have proposed a novel, integrated approach to inverse kinematics and single-query motion planning for manipulation tasks. In contrast to traditional approaches, it can consider any reachable configuration as the goal configuration during the search. Our approach is inspired by an intuitive understanding of the structure of the C-space, consisting of disconnected goal and obstacle regions. It is based on exploring a connected free component of the configuration space with a single Rapidly-exploring Random Tree (RRT). Candidate configurations are evaluated by a heuristic workspace metric that measures the manipulator's ability to achieve a desired pose of the target object. This goal distance, as well as obstacle distance information, is used to guide the search of the configuration space.

Our proposed approach has several key advantages over traditional motion planning algorithms:

- No explicit inverse kinematics computation is needed for planning.
- Only reachable, collision-free IK solutions are discovered during the search. Solutions that cause collisions or lie in disconnected components of the C-space are disregarded.
- Given a suitable goal function, our approach yields reliable planning performance that is probabilistically complete.
- Due to the purely local extension of the RRT, generated paths tend to belong to the same topological class as the shortest possible path.

Although our new planning algorithm has been shown to be capable of solving complex planning problems, several possible improvements have been identified:

- The rate of convergence greatly depends on the accuracy of the goal function. Since we used a relatively simple model, a more analytical approach to modeling the distance to the goal region may speed up convergence significantly.
- Using more efficient collision detection / minimum distance computation algorithms, such as the one proposed in [20] will improve performance, as distance computation contributes a sizable portion of the computation time per RRT node.
- Using an efficient approximate nearest neighbor algorithm for the random extension of the RRT should significantly reduce the computation time needed for RRTs containing a large number of nodes.
- Dynamically modifying the parameters of the algorithm according to the state of the search might enable more consistent behavior across different planning problems.

Exploring these implementation issues, and conducting further analysis forms the basis of our future work.

Acknowledgments

We thank the InterACT program [21] for making this joint research project possible. James Kuffner thanks Steve LaValle, Shintaro Yoshizawa and Yutaka Hirano for helpful discussions, and for partial support from NSF grants ECS-0325383, ECS-0326095, and ANI-0224419.

References

[1] G. Wilfong, "Motion panning in the presence of movable obstacles," in *Proc. ACM Symp. Computat. Geometry*, 1988, pp. 279–288.

[2] R. Alami, T. Simeon, and J.-P. Laumond, "A geometrical approach to planning manipulation tasks," in *5th Int. Symp. Robot. Res.*, 1989, pp. 113–119.

[3] R. Alami, J.-P. Laumond, and T. Simeon, *Two manipulation planning algorithms*. Wellesley, MA: A.K. Peters, 1997, ch. Algorithms for Robotic Motion and Manipulation.

[4] T. Simon, J.-P. Laumond, J. Corts, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *International Journal of Robotics Research*, vol. 23, no. 7, pp. 729–746, 2004.

[5] J. J. Craig, *Introduction to Robotics : Mechanics and Control*. Addison-Wesley, 1989.

[6] C. Klein and C. Huang, "Review on Pseudoinverse Control for Use with Kinematically Redundant Manipulators," *IEEE Transactions on System, Man and Cybernetics*, vol. 13, no. 3, pp. 245–250, 1983.

[7] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2001.

[8] T. Lozano-Perez, "Spatial planning: a configuration space approach," *IEEE Trans. Comput.*, pp. 108–120, 1983.

[9] J. C. Latombe, *Robot Motion Planning*. Boston, MA: Kluwer Academic Publishers, 1991.

[10] S. M. LaValle, *Planning Algorithms*. Cambridge University Press (also available at http://msl.cs.uiuc.edu/planning/), 2006, to be published in 2006.

[11] L. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration space," *IEEE Trans. on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[12] Y. Hirano, K. Kitahama, and S. Yoshizawa, "Image-based object recognition and dexterous hand/arm motion planning using rrts for grasping in cluttered scene," in *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2005, pp. 3981–3986.

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

[14] J. Kuffner and S. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA'2000)*, San Francisco, CA, Apr. 2000, pp. 995–1001.

[15] S. Quinlan, "Real-time modification of collision-free paths," Ph.D. dissertation, Stanford University, 1994.

[16] R. Geraerts and M. H. Overmars, "On improving the clearance for robots in high-dimensional configuration spaces," in *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2005, pp. 4074–4079.

[17] P. Cheng, "Reducing RRT metric sensitivity for motion planning with differential constraints," Master's thesis, Iowa State University, 2001.

[18] T. Asfour and R. Dillmann, "Human-like motion of a humanoid robot arm based on a closed-form solution of the inverse kinematics problem," in *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2003.

[19] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, "Fast proximity queries with swept sphere volumes," Department of Computer Science, University of N. Carolina, Chapel Hill, Tech. Rep., 1999.

[20] K. Steinbach, "Improved sphere tree construction and minimum distance computation for 3d meshes," Master's thesis, University of Karlsruhe, 2005.

[21] "interACT center," http://www.is.cs.cmu.edu/.