Asymptotically-optimal Path Planning for Manipulation using Incremental Sampling-based Algorithms

Alejandro Perez  Sertac Karaman  Emilio Frazzoli  Seth Teller  Matthew R. Walter

I. INTRODUCTION

Most manipulation platforms involve robots with multiple degrees of freedom, which results in motion planning problems in high-dimensional configuration spaces. Moreover, planning problems involving grasping and manipulation of complex objects require invoking computationally-expensive collision checking procedures several times. Consequently, existing planners can not achieve high quality solutions, e.g., in terms of a cost function, in reasonable amount of computation time.

In this talk, we present an algorithm that overcomes these difficulties by augmenting the asymptotically-optimal RRT∗ algorithm with a sparse sampling procedure, called the Ball Tree algorithm, and a memoization technique that speeds up the collision checking procedure. The proposed algorithm is specifically tailored for anytime computation. That is, a feasible solution is identified quickly, and the solution is refined towards an optimal one if the algorithm is allowed more computation time.

The algorithm is evaluated in a series of Monte-Carlo simulation studies involving seven, twelve, and fourteen degree-of-freedom manipulation planning problems using a realistic simulation environment. Simulation results suggest that the proposed algorithm provides significant improvements in both the quality of the first solution found as well as the final path that is executed by the robot, while incurring no substantial computational cost when compared to the RRT algorithm. The algorithm is also tested on the PR2 platform for single-arm and dual-arm planning problems. A more elaborate discussion of the algorithms and the results presented in this talk is given by Perez et al. [1].

II. ALGORITHM

The RRT∗ algorithm, introduced by Karaman and Frazzoli [2], is an incremental sampling-based motion planning algorithm with the asymptotic optimality guarantee, i.e., almost-sure convergence to globally optimal solutions, which the RRT algorithm lacks, without incurring substantial computational overhead when compared to the RRT.

We implement the RRT∗ algorithm by delaying calls to the collision checking procedure until absolutely necessary. During the extension phase of the RRT∗ algorithm, we first compute the cost of each path, sort them in the order of increasing cost, and check the paths for collision in this ordering until a collision-free path is found. During the rewiring phase of the RRT∗ algorithm, we invoke the collision checking procedure only if the cost of the rewiring path is low enough to improve the cost of the rewiring vertex. (see [1]). Although in the worst case, this approach will result in checking all trajectories for collision, the authors have found in the experimental studies that on average only a few paths are checked for collision, significantly improving the running time of the RRT∗ algorithm in problems in which collision checking is computationally expensive.

The Ball Tree algorithm, presented by Shkolnik and Tedrake [3], is a sampling-based method similar to the RRT that approximates connected regions of free space with balls instead of points. Treated as sets of reachable points, the algorithm uses these balls to perform rejection sampling, resulting in trees that are sparser than those of the standard RRT while maintaining probabilistic completeness.

We propose a manipulation planning algorithm that offers two compelling advantages. Firstly, it is noticeably faster than conventional planners at identifying an initial, low-cost, feasible path to the goal in configuration space. Secondly, the algorithm is able to take advantage of available computation time to refine this solution towards an optimal one. We achieve these characteristics by combining the Ball Tree algorithm, which maintains sparse trees to efficiently reach the goal, the RRT∗ algorithm, which provides the anytime refinement of the tree, and a memoization method, which speeds up the collision checking procedure.

The proposed algorithm is called the BT+RRT∗ in this text. The details of this algorithm are given by Perez et al. [1].
III. RESULTS

We evaluate the effectiveness of our algorithm through both simulation as well as through experiments on the PR2 robot. We first perform a Monte Carlo study to analyze the algorithm’s performance on two different planning problems for the PR2 robot. The first involves finding an obstacle-free path through configuration space that brings a single, seven degree of freedom arm to a pre-grasp pose. In the second scenario, we consider jointly planning trajectories for both arms (see Figure 1). The experiments were performed using the OpenRAVE simulation environment [4].

A. Single-Arm Scenario (Seven Degrees of Freedom)

The results for the seven degree-of-freedom single-arm planning scenario are summarized in Figure 2.

![Graph showing solution cost as a function of computation time, averaged over the set of single-arm Monte Carlo simulations for the four planning algorithms.](https://example.com/graph1.png)

**Graph 2. Solution cost as a function of computation time, averaged over the set of single-arm Monte Carlo simulations for the four planning algorithms. Vertical bars indicate standard deviation over the 100 runs while the open circles denote the average completion time. The bottom figure presents an inset view that compares the mean behavior of the three algorithms that utilize the RRT**

B. Dual-Arm Scenario (Fourteen Degrees of Freedom)

The results for this scenario are summarized in Table I. Allowing a maximum of 10,000 iterations, the RRT was able to find a solution in 59 of the runs and the RRT was successful in only 25. The BT/RRT planner identified a solution in all but one run while our algorithm found a trajectory every time. Much like the seven and twelve degree of freedom simulations, the BT+RRT and BT/RRT* Ball Tree planners return an initial solution much sooner than the RRT and RRT*. On average, our algorithm takes longer than the BT/RRT* to isolate an initial solution, though with the benefit of a significant improvement in cost that resembles trajectories every time. Much like the seven and twelve degree of freedom simulations, the BT+RRT and BT/RRT* return an initial solution much sooner than the RRT and RRT*. On average, our algorithm takes longer than the BT/RRT* to isolate an initial solution, though with the benefit of a significant improvement in cost that resembles trajectories every time. Much like the seven and twelve degree of freedom simulations, the BT+RRT and BT/RRT* return an initial solution much sooner than the RRT and RRT*.

### Table I

<table>
<thead>
<tr>
<th></th>
<th>BT+RRT*</th>
<th>RRT*</th>
<th>BT/RRT</th>
<th>RRT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success Rate (100 runs)</td>
<td>100%</td>
<td>95%</td>
<td>100%</td>
<td>75%</td>
</tr>
<tr>
<td>First Solution</td>
<td>Cost (rad)</td>
<td>3.82 (1.97)</td>
<td>4.20 (2.57)</td>
<td>5.65 (2.21)</td>
</tr>
<tr>
<td>Final Solution</td>
<td>Cost (rad)</td>
<td>3.84 (1.97)</td>
<td>4.20 (2.57)</td>
<td>5.65 (2.21)</td>
</tr>
<tr>
<td>Time per Solution (sec)</td>
<td>5.90 (1.97)</td>
<td>6.34 (1.97)</td>
<td>7.12 (1.84)</td>
<td>8.04 (1.97)</td>
</tr>
</tbody>
</table>

### ACKNOWLEDGEMENTS

The authors are grateful to Professors L.P. Kaelbling and T. Lozano-Perez and to Willow Garage Inc. for providing access to the PR2 experimental platform.

### REFERENCES


