

Search-based Manipulation Planning for the PR2

Benjamin Cohen, Michael Phillips, Sachin Chitta and Maxim Likhachev

I. INTRODUCTION

Motion planning for manipulation is generally thought to be too complex of a problem to solve using the same graph search algorithms that are commonly preferred in navigation planning. Graph searches are favored in the domain of navigation because they provide strong theoretical guarantees without sacrificing fast planning times. However, the high dimensionality of the manipulation planning problem generally slows down these types of searches tremendously. Instead, sampling-based algorithms are commonly used to quickly generate motion plans at the expense of the quality of the solution. Sampling based approaches rely on randomization to dodge local minima and are therefore incapable of any form of cost minimization, bounds on the sub-optimality of the solution and don't provide a guarantee that the solution will be found. Also, a major complaint with trajectories generated by these approaches is that they are highly inconsistent. Given the same input parameters, the solutions can be very different.

Consistency in a motion planner is important in achieving our goal to put service robots in the home. People become uncomfortable when they can not anticipate and understand a machine's behavior. It is our job to assure that our robots behave in a consistent and predictable fashion. Search-based approaches to path planning are deterministic. This means that not only if given the same input, the same output will be returned, but also, given similar input - similar output is expected. Such consistency in a robot's motions, both with its base and its arms, will help foster a human's ability to trust it in their home.

In our approach to planning for manipulation, we are interested in providing the same guarantees that are expected in navigation planning, such as completeness, a path will be found if one exists, and a bounds on the suboptimality of the solution. Our goal is to also provide consistency and solutions of minimal cost. To accomplish this, we use an anytime graph search with some clever modifications to generate safe trajectories quickly, more often than not in less than one second.

B. Cohen is with the GRASP Lab, University of Pennsylvania, USA bcohen@seas.upenn.edu

M. Phillips and M. Likhachev are with the Robotics Institute, Carnegie Mellon University, USA mlphilli@andrew.cmu.edu, maxim@cs.cmu.edu

Sachin Chitta is a research scientist at Willow Garage Inc., Menlo Park 94025, USA sachinc@willowgarage.com

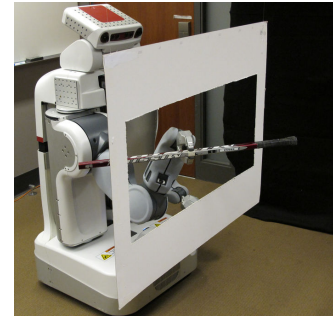


Fig. 1. Manipulating objects in cluttered environments is the primary motivation of this work.

II. APPROACH

A. Motion Planning

Our approach is based on constructing a motion primitive-based graph and searching this graph for a low-cost solution [1]. The graph structure we use is a lattice representation. A lattice is a discretization of the c-space into a set of states, and connections between these states, where every connection represents a feasible path. The set of states are the set of possible discretized joint configurations and the transitions in the graph are a set of kinematically feasible motion primitives. A motion primitive is the difference in the global joint angles of neighboring states. We define a state as an n -tuple $(\theta_1, \theta_2, \dots, \theta_n)$ for a manipulator with n joints. A motion primitive is a short atomic action defined as a vector of joint velocities, (v_1, v_2, \dots, v_n) for a subset or for all n joints.

The planner minimizes a cost function with the goal of minimizing the length of the path taken by the end effector while keeping the arm far away from obstacles. For the heuristic function, we use the 3D distance from the end effector to the goal while accounting for obstacles. The distance is computed in the form of a single 3D Dijkstra's search and it is guaranteed to be consistent. To search the graph, we use an anytime variant of A* called Anytime Repairing A* [2]. ARA* finds a highly suboptimal solution quickly and then improves it as time permits.

In [3], we showed that with the use of adaptive motion primitives we are able to find solutions with much fewer state expansions, leading to shorter planning times.

B. Learning

Many tasks in manipulation are somewhat repetitive and have similar solutions like moving objects on a tabletop,

opening doors, or loading a dishwasher. Our approach exploits previous solutions to make future queries faster. We do this by caching previous paths and building an efficient roadmap out of them. When the search finds the roadmap we allow it to "jump" to another location on the roadmap that is "closer" to the goal (by heuristic). This allows us to shortcut a lot of the search. We also have a heuristic which attracts the search to the roadmap so we can more easily connect to the roadmap quickly. The strength of this attraction to reuse old information also determines the sub-optimality bound on the solution.

III. EXTENSION TO TWO ARMS

For the past few months we have been researching methods of extending the approach mentioned above to planning for two arm manipulation. Our goal is to generate safe motion plans for two arms that are gripping the same object such as a box or a tray. We assume the grips will stay fixed throughout the motion. Extending our previous approach as is to two arms, each with seven joints, would lead to the tedious task of searching a graph with 14 DOF. So instead of having states in the graph represent the set of possible joint configurations, the states now represent the position and orientation of the object in space as well as the redundancy of each arm. The new 8D representation, $(x, y, z, roll, pitch, yaw, \theta_{left}, \theta_{right})$, simply adds one dimension to the statespace instead of doubling it. This representation has other benefits as well. The motion primitives are defined as movements in space instead of joint velocities, which are more difficult to comprehend and create. The new representation is also better suited to easily track the path suggested by the heuristic function.

Extending our search-based approach to planning for two arm manipulation is a work in progress. We plan on submitting this work to ICRA 2012. More details will be available after the work is submitted.

IV. EXPERIMENTAL RESULTS

For an extensive set of statistics on over a hundred randomly generated tests in different types of commonly found planning scenarios please refer to the results in [3].

A. Cluttered Environments

Our main focus is to develop algorithms for manipulation in cluttered environments. The heuristic function mentioned above allows us to efficiently avoid local minima caused by obstacles. In the attached video, you will find a clip showing the PR2 safely pass a long rod through a window and then back out. The window is marginally wider than the length of the rod. More videos of manipulation in cluttered environments are immediately available on demand.

B. Grasping Pipeline

We also demonstrated the utility of our motion planner with more common scenarios such as simple tabletop manipulation. We recently integrated our motion planner with the Grasping Pipeline developed and released by Willow Garage.



Fig. 2. Graspy is tired from autonomously manipulating objects in a safe and consistent way for over two hours. Keep it up Graspy!

We successfully ran the grasping pipeline autonomously for two hours, allowing it to constantly move two different objects from one side of the table to the other. The state machine made hundreds of planning requests for each arm to the planner and there was only one recorded planning failure during that time. Footage of the first 53 minutes is available upon request. In the attached clip, you will find a short snippet from that video. It's important to notice the consistency of the trajectories throughout the video.

Tabletop manipulation is a great example of a repetitive task when the planner can greatly benefit from learning. To compare the planner with and without learning, we ran the grasping pipeline in autonomous mode for twenty minutes. During that time, the planner was called over 40 times for each arm. The average planning time of the planner without learning is 0.20 seconds and 0.02 seconds with learning, lending a speedup of ten times on average.

C. Two Arms

We don't yet have results for two arm manipulation. We expect to have statistics and videos by the ICRA 2012 deadline. Our goal demonstration is to have the PR2 manipulate a tray between different shelves of a catering cart.

V. PROPOSED DEMONSTRATION

In conclusion, we are proposing a demonstration in which the PR2 will manipulate objects through a cluttered environment using our one arm planner. With the cooperation of the grasping pipeline, we can have the robot manipulate large objects on a tabletop such as a plunger.

If we are successful in completing our two arm planner before the ICRA deadline, we will also have the PR2 move a waiter's tray between different shelves on a catering cart.

REFERENCES

- [1] B. J. Cohen, G. Subramanian, S. Chitta, and M. Likhachev, "Search-based Planning for Manipulation with Motion Primitives," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.
- [2] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," in *Advances in Neural Information Processing Systems (NIPS) 16*. Cambridge, MA: MIT Press, 2003.
- [3] B. J. Cohen, S. Chitta, and M. Likhachev, "Planning for Manipulation with Adaptive Motion Primitives," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.