Single- and Dual-Arm Motion Planning with Heuristic Search

Benjamin Cohen
Grasp Laboratory
University of Pennsylvania
Philadelphia, PA 19104
bcohen@seas.upenn.edu

Sachin Chitta
Willow Garage Inc.
Menlo Park, CA 94025
sachinc@willowgarage.com

Maxim Likhachev
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
maxim@cs.cmu.edu

Abstract

Heuristic searches such as A* search are a popular means of finding least-cost plans due to their generality, strong theoretical guarantees on completeness and optimality, simplicity in implementation and consistent behavior. In planning for robotic manipulation, however, these techniques are commonly thought of as impractical due to the high-dimensionality of the planning problem. In this paper, we present a heuristic search-based approach to motion planning for manipulation that does deal effectively with the high-dimensionality of the problem. The paper presents a summary of the approach along with applications to single-arm and dual-arm motion planning with upright constraints on a PR2 robot operating in non-trivial cluttered spaces. An extensive experimental analysis in both simulation and on a physical PR2 shows that, in terms of runtime, our approach is on par with other most common sampling-based approaches and due to its deterministic cost-minimization, the computed motions are of good quality and are consistent, i.e. the resulting plans tend to be similar for similar tasks. For complete details of our approach, please refer to (Cohen, Chitta, and Likhachev 2013).

Introduction

Many planning problems in robotics can be represented as finding a least-cost trajectory in a graph. Heuristic searches such as A* search (Hart, N. J. Nilsson, and Raphael 1968) have often been used to find such trajectories. There are a number of reasons for the popularity of heuristic searches. First, most of them typically come with strong theoretical guarantees such as completeness and optimality or bounds on suboptimality (Pearl 1984). Second, the generality of heuristic searches allows one to incorporate complex cost functions and complex constraints and to represent easily arbitrarily shaped obstacles with grid-like data structures (Likhachev and Ferguson 2009). Finally, heuristic searches provide good cost minimization and consistency in the solutions. Consequently, heuristic search-based planning has been used successfully to solve a wide variety of planning problems in robotics.

Despite the wide popularity of heuristic searches, they typically have not been used for motion planning for high-DOF robotic manipulators. The main reason for this is the high-dimensionality of the planning problem. In this paper, we present a heuristic search-based planner for manipulation that combats effectively this high dimensionality by exploiting the following three observations. First, we use a *manipulation lattice graph* to represent the planning problem. A manipulation lattice graph is a sparse representation in which the states correspond to the configuration of the robot while the edges represent short motion primitives. Any path in the graph is kinematically feasible for the robot. This representation was designed to specifically handle the complexities of manipulation through the use of *static* and *adaptive* motion primitives and by decoupling the problem when appropriate by varying the dimensionality of the lattice. Second, while finding a solution that is *provably* optimal is expensive, finding a solution of *bounded suboptimality* can often be drastically faster. To this end, we employ an anytime heuristic search, ARA* (Likhachev, G. Gordon, and Thrun 2003), that finds solutions with provable bounds on suboptimality and improves these solutions until allotted time for planning expires. Third, the solutions found in a low-dimensional simplification of the workspace can serve as highly informative heuristics and can therefore efficiently guide the search despite the kinematic constraints that are inherent to manipulation.

Alternative approaches include sampling-based and trajectory optimization methods. Sampling-based motion planners (Kavraki et al. 1996; Kuffner and LaValle 2000) have been shown to consistently and very quickly solve impressive high-dimensional motion planning problems. However, these methods do not offer any cost minimization, theoretical completeness or sub-optimality bounds. RRT* (Frazzoli and Karaman 2010) is a recently proposed sampling-based method that does provide guarantees on solution optimality and completeness. Our experimental analysis confirms, the random nature of sampling-based planners including RRT* makes it difficult to generate good quality solutions fast. CHOMP (Ratliff et al. 2009), works by creating a naive initial trajectory to the goal, and then running a modified version of gradient descent on the cost function. CHOMP offers many advantages over sampling-based approaches, however like sampling-based planning, CHOMP relies on randomization to help deal with environmental complexities. One of the strengths of our approach is consistency in the solutions which help make the robot's motions more predictable.

Algorithm

Heuristic search-based planning has four major components - graph construction, an informative heuristic, the cost function and the actual search itself. Their design is critical to the planner's performance, solution quality and feasibility. In the example of motion planning for manipulation, we define the goal of the graph search itself as the search for the least cost path in the constructed graph from a state that corresponds to the initial configuration of the manipulator to a state for which the pose of the end-effector satisfies a *goal constraint* in work space.

Manipulation Lattice Graph Representation

We employ a non-uniform resolution lattice search space with non-uniform dimensionality whose edges correspond to a predefined set of actions, *static motion primitives*, as well as *adaptive motion primitives*, or actions that are generated at runtime. We call this novel approach, a *manipulation lattice graph*. Unlike a standard lattice graph, our approach is capable of decoupling the search space when appropriate and permits the use of a coarser discretization without sacrificing the ability to satisfy an arbitrary goal constraint. Our approach remains true to standard lattices, in that it is a sparse representation where every path in the lattice represents a kinematically feasible motion.

Let us use the notation $G = (S, E)$ to denote the graph G we construct, where S denotes the set of states of the graph and E is the set of transitions between the states. The states in S are the set of possible (discretized) joint configurations of the joints in the arm we are planning for. The transitions in E correspond to two types of short, *atomic*, 100 ms duration actions we call motion primitives, *static* and *adaptive*. We define a state s as an n-tuple $(\theta_0, \theta_1, \theta_2, ..., \theta_n)$ for a manipulator with n joints. It is important to note that the graph is constructed dynamically by the graph search as it expands states since pre-allocation of memory for the entire graph would be infeasible for an n DOF manipulator with any reasonable n. Each motion primitive, or mp, is a single vector of joint velocities, $(v_0, v_1, v_2, ..., v_n)$ for all of the joints in the manipulator. The set of primitives is the set of the smallest possible motions that can be performed at any given state. We refer to the pre-defined set of actions that can be performed at any given state, as *static motion primitives*. These actions are chosen before the search begins and their purpose is to uniformly explore the space for a valid path.

Adaptive Motion Primitives. Many problems in manipulation may require the robot's end effector to achieve a very specific goal configuration, e.g. if the goal of a motion is to ultimately grasp an object. Such precise positioning may be difficult to achieve on a lattice that is derived with discretization. A fine discretization will make the search more computationally expensive but a coarse discretization may make it difficult to achieve any arbitrary goal constraint. To that end, we limit the effects of discretization through the use of continuous solvers that compute the edges to connect any given state to the goal state. We call these motions a type of *adaptive* motion primitives, or primitives that are dynamically generated on-the-fly. Adaptive motion primitives are actions that don't belong to the static set and that are created as needed during a state expansion, given that the state meets a certain set of pre-defined criteria. In the motivating example of an adaptive motion to reach the goal, we say that when the state, s, that is being expanded represents an end effector position that is close to the goal, a motion primitive is generated connecting s to s_{goal}. By *snapping* to the goal pose, the search is shortened and it is capable of satisfying any arbitrary goal constraint despite the discretization.

One example of a continuous solver-based adaptive motion primitive, or *amp*, that we use is *inverse kinematics-based*. When a state s is expanded whose end-effector position, $ef_{xyz}(s)$, is within a pre-defined distance to the goal end-effector position, d_{ik}, we use an inverse kinematics (IK) solver to generate an additional mp, $amp_{ik}(s, s_{goal})$ for state s. If IK succeeds, we then construct $amp_{ik}(s, s_{goal})$ as an interpolated path from s to the solution returned by IK and also check it for collisions.

Non-uniform Dimensionality. Planning in a high dimensional lattice is computationally expensive and can require a lot of system resources. An important observation, however, is that when planning for a high dimensional arm, not all of the available DOF may be needed to find a safe path to the goal region or even to the goal position itself. Frequently, using a subset of the joints is fully adequate in computing a feasible path to the vicinity of the desired end-effector pose. Once it does get close to the desired end-effector pose, changing additional joints may become necessary in order to satisfy orientation constraints and to maneuver the end-effector in cluttered spaces.

This observation motivated us to generate a set of static motion primitives that varies in its dimensionality. A subset of this full set of motion primitives can be used to quickly search for a path to the goal region. These motion primitives are chosen such as to result in a lower-dimensional state-space. Once the search enters a potentially cluttered goal region, the planner uses the complete set of full dimensional primitives to search for a path to the goal pose in a full-dimensional state-space.

We define MP_{lowD} to be a subset of the predefined set of primitives that can change only a subset of joints. This means that in the regions where only the motion primitives from MP_{lowD} are used, the state-space is lower-dimensional (its dimensionality is the number of joints that are in the subset). MP_{fullD} is the complete set of primitives that are capable of changing all of the joints, creating a full dimensional state-space.

Non-uniform Resolution. The motion primitives we used are *multi-resolution* as well as multi-dimensional. All $mp \in MP_{lowD}$ are larger motions, allowing the search to get to the general goal region quicker. MP_{fullD} contains shorter motion primitives to allow the search to find a motion to the goal more precisely. Thus, MP_{lowD} and MP_{fullD} are two different sets.

Anytime Search

Any standard graph search algorithm can be used to search the graph G that we construct. Given its size, however, optimal graph search algorithms such as A* (Hart, N. J. Nilsson, and Raphael 1968) are infeasible to use. While finding a solution that is *provably* optimal is expensive, finding a solution of *bounded suboptimality* can often be drastically faster. To this end, we employ an anytime heuristic search, ARA* (Likhachev, G. Gordon, and Thrun 2003), that quickly finds an initial and possibly sub-optimal solution and repairs it while deliberation time allows, efficiently reusing its previous efforts. The algorithm guarantees completeness for a given graph G and provides a bound ϵ on the sub-optimality of the solution at any point of time during the search, w.r.t. to the set of motion primitives used and the resolution of the configuration space.

The cost function we used is designed to minimize the path length while maximizing the distance between the arm and nearby obstacles along the path. The cost of traversing any transition between states s and s' in graph G can therefore be represented as $c(s, s') = c_{cell}(s') + c_{action}(s, s')$. The action cost, c_{action}, is the cost of the mp which is generally determined by the user and c_{cell} is the soft padding cost.

Informative Heuristics

For a heuristic function to be most informative, it must capture the key complexities associated with the overall search, such as mechanism constraints or the environment complexities. A common approach for constructing a heuristic is to use the results from a simplified problem (e.g. from a lower-dimensional search problem where some of the original constraints have been relaxed). We use a single heuristic that guides the search towards achieving the (x, y, z) component of the goal constraint. We use a 3D breadth first search (BFS) to compute the cost of the least-cost path from a given cell to the cell that corresponds to the goal position (x, y, z) while avoiding obstacles. More details are provided in the following section.

Applications

Single-arm Planning

We construct an n dimensional statespace when planning for an arm with n joints. Each state is represented by an n element vector whose elements correlate to actual joint positions. Each motion primitive is an n element vector of velocities.

In addition to the adaptive motion primitive, $amp_{ik}(s, s_{goal})$ that we described earlier, we use another continuous solver-based amp, which we call an *orientation solver-based* primitive, or $amp_{os}(s, s_{goal})$. When a state s is expanded whose end-effector position satisfies the position constraint of the goal, $ef_{xyz}(s_{goal})$, we use an orientation solver to generate an additional motion primitive, amp_{os} for that state. The orientation solver computes the proper motions necessary to satisfy the orientation constraint, $ef_{rpy}(s_{goal})$ (roll, pitch, yaw angles of the desired end-effector pose), without moving the end effector out of its position, $ef_{xyz}(s)$.

To make the heuristic we presented in section more representative of the actual search, we represent the end-effector using its inner sphere. In our implementation, this implies that we are effectively adding an extra padding to the obstacles equal to the radius of this sphere when running the 3D breadth first search to compute heuristics.

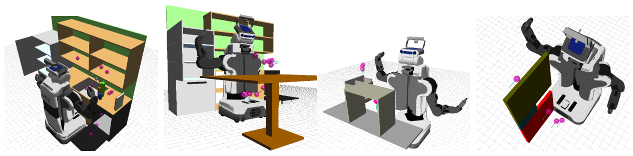
Figure 1: The four scenarios: kitchen, tabletop, industrial and narrow passageway. The pink spheres with cyan arrows indicate the desired 6D goal poses for the right end-effector.

To measure the performance of our planner, we performed a set of experiments on the right arm of the PR2 robot in four different simulated environments. The set of planners we compared our approach to are RRT* (Frazzoli and Karaman 2010) and RRT-Connect (Kuffner and LaValle 2000). The cost function for RRT* is the distance traveled in joint space. The environments we used are shown in Figure 1. In each environment, multiple goal locations are defined for the end-effector of a robot and 30 experiments are run in each one. More information on the details of all of the experiments in this paper can be found in (Cohen, Chitta, and Likhachev 2013).

Table 1 shows the performance benchmarks for all the environments in Figure 1. The sampling-based planners are very fast but our approach is certainly competitive in most environments. In general, we found that the solutions generated by ARA* are noticeably shorter in path length. Note that in these experiments both ARA* and RRT* are only run until the first solution. ARA* is initialized with $\epsilon = 100$.

For many planning problems, the consistency of the motions is important as it helps make the actions of the robot more predictable for a human interacting with it. Planning with heuristic searches is typically very consistent, meaning that similar inputs generate similar outputs. To compare the consistency of the motions generated by the planners, we performed an experiment in which all three of the planners are called to plan from a single configuration of the robot to multiple goal poses that are within a close vicinity of each other. In these experiments, the right arm of the PR2 is extended over the tabletop, and below the table are 27 6D goal poses, all of which are contained within a $10cm$ cube. The average lengths of these paths can be found in Table 2.

same start $\rightarrow$ different goals	ARA*	RRTC	RRT*
length (joint space) *(mean, rad)*	8.828	23.565	22.259
length (joint space) *(std. dev., rad)*	2.758	14.705	17.438
length (wrist) *(mean, meters)*	1.921	2.831	2.831
length (wrist) *(std. dev., meters)*	0.255	1.929	2.365
length (elbow) *(mean, meters)*	1.133	1.769	1.703
length (elbow) *(std. dev., meters)*	0.155	1.050	1.469
variance (wrist) *(total, meters^2)*	11.721	124.085	104.662
variance (elbow) *(total, meters^2)*	10.128	55.716	44.023

Table 2: Results from 27 consistency experiments.

Environment →	Kitchen			Tabletop			Industrial			Narrow Passageway		
Planner →	ARA*	RRTC	RRT*	ARA*	RRTC	RRT*	ARA*	RRTC	RRT*	ARA*	RRTC	RRT*
planning time *(mean, sec)*	0.31	0.01	0.87	0.98	0.01	0.03	0.14	0.01	6.06	0.74	0.66	3.90
planned length *(mean, rad)*	9.52	13.13	12.90	10.97	10.20	10.19	5.76	12.12	12.33	17.33	25.66	23.24
simplified length *(mean, rad)*	6.93	9.81	9.30	7.37	8.14	7.71	4.09	8.81	6.88	9.54	13.56	12.60
success rate	100%	100%	87%	100%	100%	100%	100%	100%	80%	100%	100%	100%

Table 1: Performance comparison of three planners for single-arm manipulation in the scenarios shown in Figure 1.

Dual-arm Planning with an Upright Constraint

Many dual-arm tasks come with a natural requirement that the object be kept upright throughout the entire path, such as carrying a tray with food or drink on it. The act of holding an object with two hands naturally implies a constraint where the two end-effectors of the arms have to maintain a relative configuration with respect to each other.

Instead of a 14D state space for a robot with two 7 DOF arms (i.e. PR2), we can exploit the natural dimensionality reduction that stems from the two constraints we mentioned above. Given the global pose of the object and the positions of one DOF in each arm, we can compute the complete configuration of each arm. That is, there is a one to one mapping between the 14D joint space of the two arms and the 6D space represented by the object pose and two free angles (one for each 7 DOF arm), represented as $(x, y, z, roll, \theta_1, \theta_2)$. The 6D states can be mapped back to the full 14D space whenever required, e.g. for collision checking and feasibility checking.

Given the upright constraint, we can modify the heuristic to make it more informative. Instead of modeling the object as a sphere when performing the 3D BFS, we instead model it as a cylinder, or a stack of cylindrical discs, because we are constraining the object from rolling or pitching. Modeling the object as a cylinder is significantly more informative than using an inner sphere when the object's dimensions are not similar along each axis, e.g. a tray which is very wide and flat.

Results from a set of 12 experiments are presented in Table 3. Photos of the robot during the experiments on the PR2 can be seen in Figure 2.

Time until First Soln. (s)	Expands. until First Soln.	ϵ_{final}	Expands. until Final Soln.
0.31	182	3	8,161
0.15	76	3	7,584
0.33	182	3	6,265
2.01	544	5	5,021
1.07	379	4	7,991
0.98	432	4	6,445
14.88	6,773	100	6,785
0.56	31	3	6,714
0.57	34	3	5,960
1.06	322	5	4,932
0.14	62	3	7,344
0.13	68	3	6,437

Table 3: Results from 12 simulated trials.

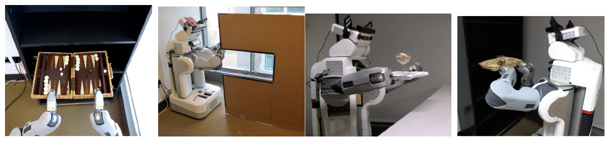

Figure 2: Four of the dual-arm experiments.

References

Cohen, B.; Chitta, S.; and Likhachev, M. 2013. Single-and dual-arm motion planning with heuristic search. *The International Journal of Robotics Research* 0278364913507983.

Frazzoli, E., and Karaman, S. 2010. Incremental sampling-based algorithms for optimal motion planning. *Int. Journal of Robotics Research*.

Hart, P. E.; N. J. Nilsson; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics* SSC-4(2):100–107.

Kavraki, L.; Svestka, P.; Latombe, J.-C.; and Overmars, M. H. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12(4):566–580.

Kuffner, J., and LaValle, S. 2000. RRT-connect: An efficient approach to single-query path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 995–1001.

Likhachev, M., and Ferguson, D. 2009. Planning long dynamically-feasible maneuvers for autonomous vehicles. *International Journal of Robotics Research (IJRR)*.

Likhachev, M.; G. Gordon; and Thrun, S. 2003. ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in Neural Information Processing Systems (NIPS) 16*. Cambridge, MA: MIT Press.

Pearl, J. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.

Ratliff, N.; Zucker, M.; Bagnell, J. A.; and Srinivasa, S. 2009. Chomp: Gradient optimization techniques for efficient motion planning. In *IEEE International Conference on Robotics and Automation*.