Combining Planning and Motion Planning: An Extended Abstract

Jaesik Choi and Eyal Amir
Department of Computer Science
University of Illinois at Urbana Champaign
Urbana, IL, 61801 USA
{jaesi,eyal}@illinois.edu

Abstract

Robotic manipulation is important for real, physical world applications. General Purpose manipulation with a robot (eg. delivering dishes, opening doors with a key, etc.) is demanding. It is hard because (1) objects are constrained in position and orientation, (2) many non-spatial constraints interact (or interfere) with each other, and (3) robots may have multi-degree of freedoms (DOF). In this paper we solve the problem of general purpose robotic manipulation using a novel combination of planning and motion planning. Our approach integrates motions of a robot with other (non-physical or external-to-robot) actions to achieve a goal while manipulating objects. It differs from previous, hierarchical approaches in that (a) it considers kinematic constraints in configuration space (C-space) together with constraints over object manipulations; (b) it automatically generates high-level (logical) actions from a C-space based on a motion planning algorithm; and (c) it decomposes a planning problem into small segments, thus reducing the complexity of planning. In conclusion, we summarize the future research directions.

1. Introduction

Algorithms for general purpose manipulations of daily-life objects are still demanding (e.g. keys of doors, dishes in a dish washer and buttons in elevators). However, the complexity of such planning algorithm is exponentially proportional to the dimension of the space (the degree-of-freedom (DOF) of the robot and the number of objects) (Canny 1987). It was shown that planning with movable objects is P-SPACE hard (Chen and Hwang 1991; Dacre-Wright, Laumond, and Alami 1992; Stilman and Kuffner 2005). Nonetheless, previous works examined such planning in depth (Likhachev, Gordon, and Thrun 2003; Kuffner and LaValle 2000; Kavraki et al. 1996; Brock and Khatib 2000; Alami et al. 1998; Stilman and Kuffner 2005) because of the importance of manipulating objects. The theoretical analysis gave rise to some practical applications (Alami et al. 1998; Cortés 2003; Stilman and Kuffner 2005; Conner et al. 2007), but general purpose manipulation remains out of reach for real-world-scale applications.

Motion planning algorithms have difficulty to represent non-kinematic constraints despite of its strength in planning with kinematic constraints. Suppose that we want to let a robot push a button to turn a light on. CSpace[1] can represent such constraints. However, the CSpace representation could be (1) redundant and (2) computationally inefficient because CSpace is not appropriate for compact representations. It could be redundant, because it always considers the configurations of all objects beside our interests (i.e. a button and a light). Moreover, mapping such constraints into CSpace would be computationally inefficient, because mapping a constraint among n objects could take $O(2^n)$ evaluations in worst case. Thus, most of motion planning algorithms assume that such mappings in CSpace are encoded.

AI planning algorithms and description languages (e.g. PDDL (McDermott 1998)) have difficulty to execute real-world robots despite of its strength in planing with generalized logical constraints. Suppose that we have a PDDL action for 'push the button' which makes a button pushed and a light turned on. However, the PDDL description could be (1) ambiguous and (2) incomplete (require details). Given a robot with m joints, it is ambiguous how to execute the robot to push the button, because such execution is not given in the description. Instead, it assumes that there is a predefined action which makes some conditions (e.g. a button pushed) satisfied whenever precondition is hold and the action is done.

Both methods solve this problem in different ways. Motion planning algorithms use abstractions to solve this problem. AI plannings use manual encodings. Although abstraction provides solutions in a reasonable amount of time in many applications, abstraction lose completeness. Thus, it has no computational benefit in worst cases. Although AI plannings have no need to search the huge CSpace, it requires manual encodings which are not only error-prone but also computationally inefficient.

We minimize manual encodings using the reachability of objects (Choi and Amir 2007; 2009b; 2009a). That is, logical actions are extracted from a tree (planned by a motion planning algorithm), if the actions change the reachability of objects (i.e. a switch can be reachable by opening a door).

Our algorithm provides a path of a robot given following inputs: configurations of a robot and objects; constraints be-

Dagstuhl Seminar Proceedings 10081
Cognitive Robotics

[1] CSpace is the set of all possible configurations

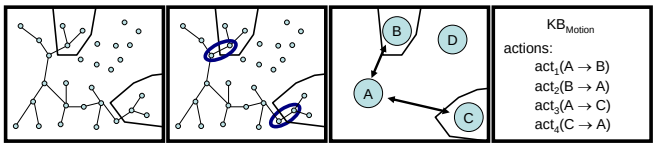

Figure 1: This figure illustrates a process to encode a motion plan into KB_M. The process is follows: (1) a motion plan (a tree) is built by a motion planning algorithm; (2) actions which changes the states of objects are found; (3) propositions are generated (and grouped) based on the found actions; and (4) a KB_M is created. Here, we assume that we have a mapping function which provides discrete states of objects given the configuration of an object in finding actions (2).

tween objects; an initial state; and a goal condition.[2] We use logical expressions to represent both spatial constraints in C-space (e.g. collision) and constraints in state space. We automatically build a set of actions from a motion planner, while it was done by hands in previous works.

In detail, our algorithm unifies a general purpose (logical) planner and a motion planner in one algorithm. It is composed of three subroutines: (1) extracting logical actions from a motion planner, (2) finding an abstract plan from the logical domain, and (3) decoding it into C-space. It extracts PDDL (McDermott 1998) actions KB_{Motion} from a tree constructed by a motion planner in C-space. ((Choi and Amir 2009a) provides a formulation for Situation Calculus.) Then, it combines extracted actions with a given KB_{Object} (Knowledge Base) that has propositions, axioms (propositional formulas) and abstract PDDL actions. In the planning step, an abstract plan is found by a PDDL planner (e.g. factored planning algorithms (Amir and Engelhardt 2003; Brafman and Domshlak 2006)). Then, a concrete path is found from the abstract plan by another motion planner.

The complexity of the planning problem is bounded by the treewidth of the encoded KB. One may think some analogy between the treewidth of KB in this paper and the number of mutually-interfering objects in the motion planning literature. However, the treewidth is more general expression because KB has more expressive power than the conventional C-space. In addition, this work proposes two improvements in terms of efficiency. One improvement is to use a factored planning algorithm for the decomposed domain. The other is to encode actions on behalf of workspace which is much smaller than C-space.

2. Related Works

Here, we review the related works in two aspects: (1) using logical representation in robot planning; and (2) modifying the motion planning algorithm to achieve complex task (eg. manipulating objects). One may see the former way as top-down and the latter way as bottom-up.

(Alami et al. 1998) presents a well-integrated robot architecture which controls multiple robots. It uses logical representations in higher level planners and C-space based motion planners in lower-level planning. (Conner et al. 2007) provides an improved way to combine the *Linear Temporal Logic (LTL)* to control continuously moving cars in the simulated environment. Its model is a nondeterministic automata, while our model is deterministic. Due to the intractability of nondeterministic model, the representation is restricted to a subset of LTL to achieve a tractable (polynomial time) algorithm. These AI planners still requires manual encodings between two layers.

[2]For each object, we use a function which maps from a configuration to discrete states (labels) of objects, if discrete states are required for the provided constraints of objects (KB_{Object}).

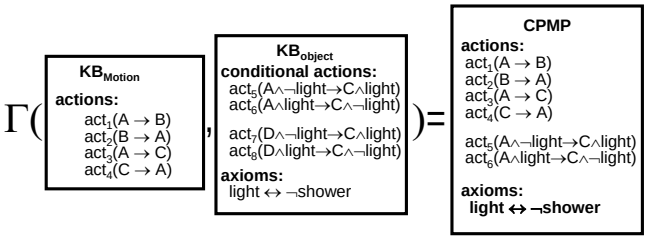

Figure 2: This figure shows an operation (or algorithm) to combine the extracted KB_M (KB_{Motion}) with pre-existing KB_O (KB_{Object}). KB_O is independently given in a general form to a robot. Thus, KB_O can be reusable for robots with different configurations space. Meanwhile, KB_M is specific to a robot. Thus, some actions (e.g. act_7 and act_8) in KB_O are invalidated by the KB_M.

Motion planning research has the long-term goal of building a motion planning algorithm that finds plans for complex tasks (eg. manipulating objects). (Stilman and Kuffner 2005) suggests such a planning algorithm based on a efficient heuristic planner (Chen and Hwang 1991) which relocates obstacles to reach a goal location. Recently, it was extended to include constraints over object into the *C-space* (Stilman 2007). In fact, the algorithms conjunction with probabilistic roadmap method (Kuffner and LaValle 2000) are highly effective in manipulating objects.

Other works also present efforts in this direction to build a motion planning algorithm for complex tasks. (Plaku, Kavraki, and Vardi 2008) solves a motion planning problem focused on safety with logical constraints represented with LTL . (M. Pardowitz 2007) focuses on learning actions for manipulating objects based on the explanation based learning (Dejong and Mooney 1986). They use a classical hierarchical planner in planning.

3. An Experiment in Simulation

In the simulation, we build our algorithm for a task that pushes buttons to call numbers. There are 8 buttons in total. 4 buttons ($key1(P1)$, $key2(P2)$, $unlock(P3)$, and $lock(P4)$) are used to lock (and unlock) the buttons. Other 4 buttons ($\#A(P5)$, $\#B(P6)$, $\#C(P7)$ and $Call(P8)$) are used to make phone calls. Initially, the button is locked, the robot needs to push unlock buttons after pushing both key buttons ($P1$ and $P2$). Then, the robot can make a phone call with pushing the $Call$ button ($P8$) after selecting an appropriate number among $\#A(P5)$, $\#B(P6)$, and $\#C(P7)$. After a call, the buttons are automatically unlocked. We encode such constraints and action in KB_{Object}.

To build KB_{Motion}, we build a tree from a randomized

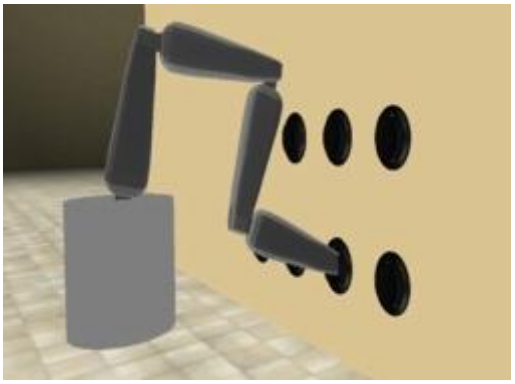

Figure 3: This is a capture of the motion of push button in the wall in experiments. The robot has 5 DOFs (rotational joints on the base and 4 revolute joints on the arm). We do experiment with increasing the number of joints from 2 to 9.

algorithm with 80000 points in C-space. With a labeling function that returned the states of buttons, we found 33 edges in the tree. They are encoded into 8 actions in KB_M for 8 buttons. Then, the combined KB ($CPMP$) is used to find a goal (calling all numbers ($\#A$, $\#B$, and $\#C$). The returned abstract actions are decoded into a path on the tree of motion plan. Figure 3 is a snapshot of the simulation.[3]

4. Conclusion and Future Research

We present an algorithm that combines the general purpose (logical) planner and a motion planner. Our planner is designed to manipulate objects with robot. To solve the problem, previous works used a hierarchical planner (high-level) and a motion planner (low-level). Most of them used manual encodings between two layers. That was one of technical hardness of this problem.

Theoretically, the combination of such planner is hard for the following reasons: (1) hierarchical planner is hard and not feasible sometime; and (2) direct combination of C-space and state space gives an doubly exponential search problem. Moreover, we can miss the geometric motion planning information, if we translate everything to PDDL (McDermott 1998) without a motion planner.

Our algorithm combines the C-space and state space in a KB, CPMP (Combining Planning and Motion Planning). This is a unique approach in a way that actions of AI planner are generated by a motion planner. Moreover, we provide the computational analysis. That is, we prove that the treewidth of CPMP determines the hardness of a manipulation task.

The combining planning and motion planning is a generalized framework. However, there are many research problems to be solved in the future research. First, an algorithm which learns the mapping function between two spaces would be required. Our algorithm assumes that there is a mapping function which provides the value of shared propositions given a configuration of C-Space. Thus, an algorithm which can detect the change of shared propositions with sensors would be promising. Second, the exploration steps may take long time due to the large cardinality of state space. Thus, an adaptive exploration algorithm which builds a tree or a graph in CSpace based on the constraints of state space would be useful.

[3]The details of encoded actions and movies are available at http://reason.cs.uiuc.edu/jaesik/cpmp/supplementary/.

References

Alami, R.; Chatila, R.; Fleury, S.; Ghallab, M.; and Ingrand, F. 1998. An architecture for autonomy. *International Journal of Robotics Research* 17(4):315–337.

Amir, E., and Engelhardt, B. 2003. Factored planning. In *IJCAI*, 929–935.

Brafman, R. I., and Domshlak, C. 2006. Factored planning: How, when, and when not. In *AAAI*.

Brock, O., and Khatib, O. 2000. Real-time replanning in high-dimensional configuration spaces using sets of homotopic paths. In *ICRA'00*, 550–555.

Canny, J. 1987. *The Complexity of Robot Motion Planning*. Cambridge, MA: MIT Press.

Chen, P., and Hwang, Y. 1991. Motion planning for a robot and a movable object amidst polygonal obstacles. In *ICRA'91*, 444–449.

Choi, J., and Amir, E. 2007. Factor-guided motion planning for a robot arm. In *IROS'07*, 27–32.

Choi, J., and Amir, E. 2009a. Combining motion planning with an action formalism. In *Commonsense*, 19–26.

Choi, J., and Amir, E. 2009b. Combining planning and motion planning. In *ICRA'09*, 238–244.

Conner, D. C.; Kress-Gazit, H.; Choset, H.; Rizzi, A.; and Pappas, G. J. 2007. Valet parking without a valet. In *IROS'07*.

Cortés, J. 2003. *Motion Planning Algorithms for General Closed-Chain Mechanisms*. Ph.D. Dissertation, Institut National Polytechnique de Toulouse, Toulouse, France.

Dacre-Wright, B.; Laumond, J.-P.; and Alami, R. 1992. Motion planning for a robot and a movable object amidst polygonal obstacles. In *ICRA'92*, volume 3, 2474–2480.

Dejong, G., and Mooney, R. 1986. Explanation-based learning: An alternative view. *Mach. Learn.* 1(2):145–176.

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

Kuffner, J. J., and LaValle, S. M. 2000. RRT-connect: An efficient approach to single-query path planning. In *ICRA'00*.

Likhachev, M.; Gordon, G.; and Thrun, S. 2003. ARA*: Anytime A* search with provable bounds on sub-optimality. In *NIPS'03*.

M. Pardowitz, R. Zollner, R. D. 2007. Incremental acquisition of task knowledge applying heuristic relevance estimation. In *IROS'07*.

McDermott, D. 1998. The planning domain definition language manual.

Plaku, E.; Kavraki, L. E.; and Vardi, M. Y. 2008. Hybrid systems: From verification to falsification by combining motion planning and discrete search. *Formal Methods in System Design*.

Stilman, M., and Kuffner, J. 2005. Navigation among movable obstacles: Real-time reasoning in complex environments. *International Journal of Humanoid Robotics* 2(4):479–504.

Stilman, M. 2007. Task constrained motion planning in robot joint space. In *IROS'07*.