A planning algorithm for G-TAMP

Beomjoon Kim

January 1, 2019

1 Introduction

Motivation:

- TAMP planners integrate discrete symbolic reasoning with continuous geometric reasoning to plan robot motions to achieve a high-level objective.
- While TAMP planners need make decisions at different levels of abstractions, for a large class of TAMP problems, we need to eventually solve a sequence of reaching-and-packing problems. For instance, consider the task of cooking a meal. We may, at a purely symbolic level, make a plan that says the target raw food should be washed first, and then placed on a stove on top of a frying pan, without performing any geometric reasoning. At a more concrete level, however, the problem eventually comes down to finding the motions for reaching and placing it in the region determined in the task plan. All the while, if there are any obstacles that are blocking the path of the reaching-and-packing motion, the planner also need to identify these objects and find motions to clear them.
- What makes such geometric reasoning challenging is the intricate relationship among individual reaching-and-packing problems: how we clear obstacles for reaching-and-placing the raw food can influence how we reach-and-place the frying pan.
- We define such problems as geometric TAMP problems. Our goal is to first formally formulate this problem, and identify the structures of these problems to propose a planning algorithm that exploits such structures to solve them efficiently and optimally. Our algorithm can be used as an interface layer used to check the feasibility of the task-plan, much like how existing TAMP planners use motion planners such as RRT to check the feasibility of individual pick and place operator instances.
- We focus on mobile-manipulation problems where, feasibility check might come down to calling a path planner with an infeasible problem instance, rather than simply checking existence of an inverse kinematics solution and a collision at the end-configuration which in general takes faster than reporting failure with a sampling-based path planner.

Informal problem formulation:

- Specifically, given a task-plan that consists of the sequence of abstract and discrete operators which satisfies the task-level goal, our objective is to find the continuous decision variables such as grasps, base poses, placements, and collision-free paths that correspond to the given abstract operator sequence, as well as any obstacles that are in the way, and the associated continuous variables to clear them, such that the solution optimizes the objective function.

- This is a long-horizon problem with an infinite branching factor and a set of hybrid decision variables. We could, in principle, use uniform-random sampling to sample operator instances and heuristic-forward search that reconsiders past nodes, to find a solution. However, this algorithm is inefficient in that it cannot determine which branch to re-explore, or which node to back-track to. Moreover, as we will show, uniform sampling cannot find an optimal solution.

Algorithmic contribution:

- We offer an anytime planning algorithm, which, in similar spirit to Monte-Carlo Tree Search, uses the online estimate of the heuristic to decide which branch to re-explore in a principled manner. Our algorithm operates on different levels of hierarchy which effectively shortens the planning horizon, and allows efficient back-tracking. To determine the obstacles to clear, we use the swept-volume of reaching and packing motions for the key objects, rather than random sampling. To optimize the heuristic function, we propose a novel sampling strategy that not only finds a feasible solution but also an optimal solution under mild assumptions.

- Specifically, our algorithm operates at three different levels of hierarchy. In the highest level, we have the task-planner that outputs the abstract task plan. This task plan is then passed on to the next layer. In the second layer, we need solve the *reaching* and *packing* problems in which we decide how we will fetch the given object, by finding grasps and pick base pose, as well as how to pack an object into a region, by finding object placement, placement base pose, and motions from the pick configuration to the place configuration, ignoring any obstacles in the way. The swept-volume of the fetching and packing motion is then passed into the layer below, which then solves an instance of the *navigation-among-movable-obstacles (NAMO)* problem. At this stage, the abstract plan becomes completely concrete, and the planner outputs the sequence of configurations that clears obstacles and fetches and packs the target object. We then move onto the next abstract operator in the task plan.