Robotics and Autonomous Systems 62 (2014) 355–365

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Manipulation planning using learned symbolic state abstractions

Richard Dearden*, Chris Burbridge

School of Computer Science, University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK

HIGHLIGHTS

- Robotic manipulation planning system.
- Goals specified symbolically so different geometric solutions can be found.
- Learned two-way mapping between symbolic and geometric states.
- Mapping allows a symbolic state to be extracted from a scene.
- Allows a plan to be translated into a sequence of geometric configurations.

ARTICLE INFO

Article history:
Available online 11 October 2013

Keywords:
Intelligent robots
Supervised learning
Automatic planning
Symbolic reasoning

ABSTRACT

We present an approach for planning robotic manipulation tasks that uses a learned mapping between geometric states and logical predicates. Manipulation planning, because it requires task-level and geometric reasoning, requires such a mapping to convert between the two. Consider a robot tasked with putting several cups on a tray. The robot needs to find positions for all the objects, and may need to nest one cup inside another to get them all on the tray. This requires translating back and forth between symbolic states that the planner uses, such as stacked (`cup1,cup2`), and geometric states representing the positions and poses of the objects. We learn the mapping from labelled examples, and importantly learn a representation that can be used in both the forward (from geometric to symbolic) and reverse directions. This enables us to build symbolic representations of scenes the robot observes, but also to translate a desired symbolic state from a plan into a geometric state that the robot can achieve through manipulation. We also show how such a mapping can be used for efficient manipulation planning: the planner first plans symbolically, then applies the mapping to generate geometric positions that are then sent to a path planner.

1. Introduction

The vast majority of robots are currently controlled using hand-written programs. These programs are typically highly specialised to a single task, and include human-specified geometric information about the objects and behaviours being performed. This greatly limits the opportunities for reuse of the programs, and also makes creating new ones time consuming as it involves a mixture of writing code and recording geometric states of the robot and the objects the robot is interacting with. In this paper, we investigate how we can use parts of existing hand-written programs as atomic actions to produce a higher-level interface to the robot—one where a user specifies desired goals and the robot reassembles the parts of existing programs to generate a new program to achieve the goal. In particular, we are interested in robot manipulation tasks; we will use tidying a house as a motivating example throughout the paper.

To allow the user to create new robot behaviours by specifying goals, we use an automatic planning approach. We represent the purpose of a fragment of one of the hand-built programs in terms of the *preconditions* that must be true to execute the code fragment and the *effects* it has on the state of the world. We refer to this representation as a *planning action*. The planner then searches for a sequence of these actions that will achieve the user-specified goal.

Planning is challenging for robotic manipulation tasks because they contain a mixture of symbolic and geometric constraints. For example, we can plan to tidy a table by placing cups and plates on a tray and then moving them to the kitchen to be cleaned, but to accomplish the task we need to reason symbolically—how to achieve that no objects are on the table—and geometrically—where should I put each object, and how should I move the robot arm to achieve that. Many manipulation planning approaches (see Chapter 7 of [1] for an overview) assume that the task can be treated entirely as a geometric problem, with the challenge being to place all the objects in their desired positions. However, this

* Corresponding author. Tel.: +44 121 414 6687.
E-mail addresses: richard.dearden@gmail.com, RDearden@slb.com (R. Dearden), c.j.c.burbridge@cs.bham.ac.uk (C. Burbridge).

0921-8890/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.robot.2013.09.015

approach means that we are essentially providing part of the plan in advance. In our example, if one cup must be nested in another to get them all on the tray, this must be specified in the goal. By planning symbolically as well as geometrically, we make fewer assumptions about the domain information available to us, and can find a much richer set of plans for a task.

Planning with a mixture of symbolic and geometric states requires a set of symbolic predicates that correspond to geometric relationships. For example, we need to be able to express that a cup is touching the tray, which we write symbolically as `touching(cup,tray)`, but this predicate corresponds to a set of geometric configurations of the two objects, and some way must be found to represent this if we are to generate geometries for which the predicate is true. Previous approaches (see, for example, [2]) have manually built functions to do this. However, since we have the example programs available to us, we take the approach of learning the mapping between symbolic predicates and geometric states. By simulating the example programs we can generate a number of geometric states, and through a combination of comments in the code and knowledge of the program semantics we can label them with the predicates that hold in each one. This removes the burden on users to create the mapping themselves. By selecting the right representation, we can also use it both to translate from geometric to symbolic states (so as to determine what the symbolic representation of the current state is) and vice versa (to generate a geometric state corresponding to some symbolic predicate in a plan). Because the training data come from the existing robot programs, the learned mapping will also naturally reflect any implicit constraints in the programs, for example ensuring that no objects are placed too close to the edge of a table.

We learn a kernel density estimate [3,4] from the training data for each predicate. This allows us not only to label unseen geometric states (the forward direction for the mapping), but also, via a hill-climbing search in the probability density function, to find geometric states that make predicates true with high probability (the backwards direction). We extend this to find geometric states for conjunctions of symbolic predicates and also – for backtracking – to find multiple, significantly different, geometric states corresponding to a symbolic state (this work was first published in [5]).

Because of the interactions between the symbolic and geometric parts of the planning problem, the ideal planning approach would be to interleave the two, with geometric planning potentially causing symbolic backtracking when it fails, and vice versa. Unfortunately, generating paths and geometric states is very time consuming [6], so this *hybrid* planning approach is often unacceptably slow due to the many geometric states and robot path plans generated during the search for a plan. To overcome this, we take a different approach, preferring to generate a complete plan at a purely symbolic level, and then we translate that plan into a geometric one and generate paths to achieve the geometric configurations. If this process fails, we allow the system a limited amount of purely geometric backtracking – proposing new positions for the objects moved by the robot – before giving up and backtracking at the symbolic level to generate a different plan.

The structure of this paper is as follows. In the next section, we discuss related work on both mapping between symbolic and geometric representations and on planning for robotic manipulation tasks. In Section 3, we describe the symbolic to geometric mapping in detail, and in Section 4 we present the planning algorithm that uses it. We present experiments to demonstrate the effectiveness of the approach in Section 5, and present our conclusions and future work in Section 6.

2. Related work

A number of planning systems have been developed that operate in a mixture of geometric and symbolic states and therefore need to map between them. Probably the best known is aSyMov [7], which solves tasks very similar to ours using a planner that combines a symbolic planner with a probabilistic roadmap [8] for geometric planning. In common with most of these approaches, the authors assume that they are given a mapping from geometric to symbolic states. In addition, they restrict the objects to certain fixed world positions and only consider a single symbolic–geometric predicate "on", so the translation from symbolic to geometric states is trivial.

Kaelbling et al. [2] have proposed an alternative approach to manipulation planning which does full hybrid planning, but, to reduce the complexity of the problem, uses a hierarchical planner and interleaves planning with execution. As with the above approach, it again only seems to use a single symbolic predicate for a geometric concept. Their mapping from symbolic to geometric states is handled by *geometric suggesters* which are hand-built functions for generating geometric states. Similar approaches have been proposed by Wolfe et al. [9] and Karlsson et al. [10], which both use a hierarchical task network planner to constrain the search space to make full hybrid planning more tractable. In [10], the authors use a uniform sampling approach to generate geometric states when backtracking, and in [6] the same authors extend this by using constraints to further reduce the amount of backtracking needed to find plans.

Other recent "fully hybrid" approaches include work on *semantic attachment* by Dornhege et al. [11,12], and on *planning modulo theories* by Gregory et al. [13]. In the semantic attachment approach, the symbolic planning language includes calls to external functions that check preconditions of actions or calculate their effects and may update a number of state variables. For a manipulation domain, as described in [12], the RRT planner used to compute arm motions, and the geometric grasp planner used to check if an object can be grasped, would be linked to the symbolic planner through semantic attachments. Similarly, the planning modulo theories approach uses specialised solvers for specific theories – path planning might again be an example – to assist the symbolic planner. In both cases, it is unclear if it would be possible in a general way to use the specialised solvers to update our learned geometric predicates.

The more traditional approach to manipulation planning is surveyed in Chapter 7 of [1]. These approaches treat planning as a completely geometric problem of searching in the set of reachable geometric configurations to reach a goal. However, this requires a geometric goal to be specified, which means that alternative geometric solutions that would solve the same symbolic goal cannot be generated—for example placing two cups on a tray or nesting the two cups on the tray.

A recent alternative proposed by Mösenlechner and Beetz [14] is to specify goals symbolically but evaluate the plan geometrically. The idea is to use a high-fidelity physics simulation to predict the effects of actions and a hand-built mapping from geometric to symbolic states. Planning is conducted by a forward search, and the effects of actions determined by simulating them, and then using the mapping to update the symbolic state. This has advantages in terms of robustness, since multiple simulator runs can be performed, but is potentially very expensive for complex plans.

The idea of learning a mapping from geometric to symbolic states has been considered before in the literature. However, what sets our approach apart from others is the idea of using the learned mapping in reverse, generating geometry from a symbolic description to enable symbolic planning operators to be utilised to produce geometric effects.