Planning clearing actions in cluttered scenes by phasing in geometrical constraints [1]

Nicola COVALLERO [a] David MARTÍNEZ [a] Guillem ALENYÀ [a,2], and Carme TORRAS [a]

[a] *Institut de Robotica i Informàtica Industrial, CSIC-UPC*

Abstract. Manipulation planning of cluttered objects involves a mixture of symbolic and geometric constraints which makes such planning very time consuming and often unsuitable for real applications. We propose to divide the geometric restrictions in two groups. The ones in the first group are used to generate a set of symbolic states used for planning. The evaluation of the ones in the second group is delayed after planning, and only relevant ones are evaluated when necessary. We demonstrate our proposal in a simple but effective implementation using pushing and grasping actions.

1. Introduction

In this paper we explore how to combine symbolic planning with geometrical restrictions to perform robot table clearing tasks. This is a challenging problem because the robot will not be able to grasp directly most of the objects, as the possible trajectories will be blocked by other objects. In this scenario non-prehensile actions are necessary in order to move the objects and grasp the desired ones.

We consider a real robot system with non-deterministic perceptions and imprecise pushing and grasping actions. To solve this problem we propose to use manipulation planning, which merges manipulation skills and planning to find the sequences of actions to attain the goal. The difficulty of these problems lays on the computational costs of considering a mixture of symbolic and geometric restrictions, as the latter are very time-consuming. Therefore we propose an approach in which we plan symbolically to find the best sequence of actions given a cost function. As planning symbolically is fast, but not all geometrical constraints can be tackled within symbolic predicates, we delay the geometrical evaluation after the plan is computed.

[1]This work has been supported by the MINECO project RobInstruct TIN2014-58178-R and the ERA-Net CHIST-ERA project I-DRESS PCIN-2015-147.

[2]Corresponding Author: IRI, Llorens i Artigas 4-6, 08028 Barcelona, Spain; E-mail: galenya@iri.upc.edu.

Many manipulation planning approaches, like aSyMov [1], assume that the task can be treated as a geometric problem with the goal to place the objects in their desired positions. Generally, hybrid planners that consider a combination of symbolic predicates and geometric features usually require too much computing time even for slightly complex problems and could not be adapted to real applications.

Similarly to ours, other approaches use a symbolic planner to tackle more complex problems, and include geometrical restrictions separately. A recent alternative proposed by Msenlechner and Beetz [2] is to plan symbolically but evaluate the plan geometrically with a simulator. The cost of this approach in cluttered scenes can be very high.

Dogar and Srinivasa [3] use geometric planning to rearrange the objects that are surrounding a target object. They can only consider goals with a single object; in contrast, we provide a more complete symbolic planning approach that can find the optimal sequence of actions to complete goals that involve a set of objects.

Recently, Laskey *et al.* [4] proposed a reactive strategy to push objects where robot motion is learned using Learning from Demonstration. Using a different approach but obtaining also complex pushing actions, King *et al.* [5] proposed to embed the physical model into the motion planner to solve complex rearranging problems. Our system can potentially use motions learned this way.

In this paper we propose a new system to compute the most convenient plan (given a cost function) taking into account geometrical restrictions. To this end, the states contain the symbolic information and the geometrical restrictions that are easy to compute, and then the costly geometrical restrictions are only considered in a lazy way. The system decides which objects to move or grasp, the most convenient order and it handles the uncertainty of the outcomes by replanning after each executed action. We also show that by combining pushing and grasping actions we can solve complex tasks involving scenes with cluttered objects, where finding a suitable plan is challenging. The performance of the method is demonstrated by means of experiments with a real robot.

2. Planning with relational and reachability constraints

We propose to divide the geometric constraints in two groups: *relational* and *reachability. Relational* geometric constraints are those generated between objects when executing grasping or pushing operations. They are computed by simulating pushing actions and checking all the collisions between objects, and also simulating grasping actions and checking collisions between the robot and the objects. *Reachable* constraints are generated when computing the robot motion path, possibly taking into account obstacle avoidance.

The planning sequence is described in Alg. 1. We propose to embed the *relational* and *reachability* constraints in the state definition and thus are tackled naturally by the planner (see Sec. 2.1). After finding an initial plan without limits regarding the *reachability* (line 1) the *reachability* constraints are evaluated. If they can be fulfilled (line 4), the next action is executed (line 10); otherwise, the state is updated accordingly and replanning is triggered (lines 5-6) to find

Algorithm 1 Planning iteration

```
Input: state
plan ← planning(state);
if hasSolution(plan) then
    action ← IK(plan[0]); {Compute the IK of the first action of the plan}
    while ¬isFeasible(action) and hasSolution(plan) do
        state ← updateState(action);
        plan ← planning(state);
        action ← IK(plan[0]);
    end while
    if isFeasible(action) and hasSolution(plan) then
        execute(action);
    end if
end if
```

an alternative solution. In our implementation we validate only the robot inverse kinematic (IK) for the next action as a proof-of-concept, but the same scheme holds for more elaborated strategies, like using heuristics to compute only some key actions [7], compute all actions [8], or use complex robot motion planners [9].

The planner used is the *Fast Downward* planner [10], a very well-known classic one. This planner is feature-wise complete, stable and fast in solving planning problems. The planning takes the state of the scene(Sec. 2.1) and uses the action model (Sec. 2.2) to compute a plan. In our current implementation, the whole plan until the goal is fulfilled is computed. If the plan cannot be found and replanning is not effective, the system cannot continue. To overcome this limitation, subgoals [6] could be used to find a feasible sub-plan that allows the robot to continue towards the goal.

It has been proved that for non-probabilistic interesting problems a well-written replanner outperforms a well-written probabilistic planner [11]; this may hold also for probabilistic interesting planning problems. Planning at a deterministic symbolic level makes the planning stage fast and this allows the system to work efficiently both with replanning and backtracking. If an unexpected effect happens it will be naturally considered in the next iteration, as we propose to replan every time the state is updated, as has been demonstrated to be effective for robotics applications [12].

2.1. State

The scene is described with symbolic predicates:

- `(removed obj0)`: object `obj0` has been grasped and removed from the table.
- `(on obj1 obj0)`: `obj1` is on top of `obj0`.
- `(block_grasp obj1 obj0)`: `obj1` prevents the robot to grasp `obj0` because the gripper would collide with `obj1`.
- `(block_push obj1 obj0 dir1)`: `obj1` prevents the robot to push `obj0` along the pushing direction `dir1` because `obj1` would collide either with `obj0` or with the end effector.

- `(ik_unfeasible_push obj0 dir1)`: either the IK of the action which pushes `obj0` along pushing direction `dir1` has no solution or `obj0` would be pushed outside the working space.
- `(ik_unfeasible_grasp obj0)`: the IK of the action which grasps `obj0` has no solution.

2.2. Action Model

The action model consists of a set of rules that represent the actions. For simplicity we have considered two actions with few parameters. This can be enlarged, for example, by considering different grasping poses to grasp the same object, as well as considering other actions like *pull*.

`(grasp obj0)`

The robot grasps `obj0` and then drops it into a bin. In our implementation, the robot grasps the object at its centroid and the orientation of the gripper is computed accordingly to the principal components of the object, but general grasping algorithms can be also considered instead [13].
Preconditions:

- no object stands on top of `obj0` (if `obj0` has objects on the top and the robot would grasp `obj0` the objects on the top would likely fall),
- no object collides with the gripper,
- the IK has solution.

Effects:

- `obj0` is removed,
- `obj0` no more impedes other objects to be pushed or grasped,
- `obj0` is not on top of other objects.

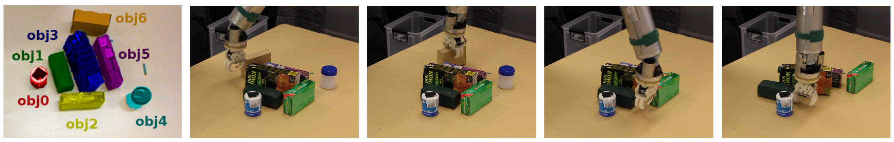
Cost: The cost for the grasping action is 1.

`(push obj0 dir1)`

The pushing action consists of translating `obj0` along pushing direction `dir1` until it can be grasped. We simplify our implementation by considering only the object's principal axis projected onto the table plane and the axis orthogonal to the principal one. In total there are 4 possible pushing directions per object, 2 per axis. The principal axis is computed using principal component analysis.
Preconditions:

- no object stands on top of `obj0` (otherwise the objects on the top would likely fall),
- `obj0` does not stand on top of other objects (otherwise `obj0` would likely fall),
- when `obj0` is pushed along `dir1` no object collides with `obj0` and with the gripper,
- the IK has solution.

Effects:



(a) (b) (c) (d) (e)

Figure 1. Table clearing experiment. (a) 7 objects and superimposed labels. (b)(c)(d) and (e) the robot is respectively executing the action `(push obj6 dir1)`, `(grasp obj6)`, `(push obj2 dir1)`, `(push obj1 dir1)`.

- `obj0` no more blocks other objects from being pushed or grasped,
- the other objects no more block `obj0` from being pushed or grasped,
- grasping `obj0` is feasible (It is possible that `obj0` cannot be grasped in a certain pose, because the IK has no solution, but it could be moved in a new pose in which it can be grasped).

Cost: Penalize actions with small collision free range that could result in collision between the robot arm and the objects. In our implementation, the cost is defined as a function of the minimum distance d_{min} between the end effector and the other objects along the path of the action

$$c = \lceil e^{k(n-d_{min})} \rceil$$

where $k = 100$ is the gain factor and $n = 0.05$ refers to the minimum distance, in metres, to consider the pushing action safe. When $d_{min} \geq n$ the cost is 1.

3. Experiments

The system designed has been implemented using ROS on a Barrett WAM arm. We present here a detail of the trace of one clearing trial for a scenario with 7 objects (Fig. 1) [3].

The setup of the experiment can be observed in Fig. 1a. It contains 7 objects in a very close position. The next couple of images show an example of the strategy of pushing an object to place it in a grasping-enabled position. The objective is to grasp object 6. Figure 1b corresponds to the execution of a push action applied to object 6. Observe that in its original position the object cannot be grasped because the gripper would collide with object 3 and object 5. But after the push action it becomes graspable (Fig.1c).

The next couple of images show a different strategy based on clear all the clutter around an object. The objective is to grasp object 0, but observe that the gripper can collide with object 1. The planner finds a solutions to that situation by first selecting a push action applied to object 2 (Fig 1d), and then a push action to object 1 (Fig.1e).

For the experiment in Fig. 1a we performed 3 runs with the goal to clear the table. The total average time spent in the planning and geometrical constraints

[3] Additional material: www.iri.upc.edu/groups/perception/phasinginplan

checking were of 82 seconds and 46 seconds respectively, and the task was solved with 12 actions (7 grasp and 5 push).

4. Conclusions

In this work, a symbolic system that plans at a deterministic semantic level and accounts for geometrical restrictions is proposed. Geometrical constraints are divided into *relational* (collisions between objects or with the robot) and *reachability* (unfeasible actions due to non-solvable IK or path planning failures). *Relational* constraints generate a set of symbolic states that are used by the planner to compute a plan. The evaluation of *reachability* constraints is delayed until the plan has to be executed. A simple implementation is presented showing that the planner can discover different combinations of pushing-grasping actions.

References

[1] S. Cambon, F. Gravot, and R. Alami, "A robot task planner that merges symbolic and geometric reasoning," in *ECAI*, vol. 16, 2004, p. 895.

[2] L. Mösenlechner and M. Beetz, "Using physics- and sensor-based simulation for high-fidelity temporal projection of realistic robot behavior," in *AIPS*, 2009.

[3] M. Dogar and S. Srinivasa, "A framework for push-grasping in clutter," in *Robotics: Science and Systems VII*, N. R. Hugh Durrant-Whyte and P. Abbeel, Eds. MIT Press, July 2011.

[4] M. Laskey, J. Lee, C. Chuck, D. Gealy, W. Hsieh, F. T. Pokorny, A. D. Dragan, and K. Goldberg, "Robot grasping in clutter: Using a hierarchy of supervisors for learning from demonstrations," in *IEEE Conf. on Automation Science and Engineering*, 2016.

[5] J. E. King, J. A. Haustein, S. S. Srinivasa, and T. Asfour, "Nonprehensile whole arm rearrangement planning on physics manifolds," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 2508–2515.

[6] D. Martínez, G. Alenyà, and C. Torras, "Relational reinforcement learning with guided demonstrations," *Artificial Intelligence*, 2016, to appear. [Online]. Available: http://www.iri.upc.edu/publications/show/1621

[7] T. Lozano-Prez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 3684–3691.

[8] F. Lagriffoul, D. Dimitrov, A. Saffiotti, and L. Karlsson, "Constraint propagation on interval bounds for dealing with geometric backtracking," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 957–964.

[9] I. A. Sucan and S. Chitta, "Moveit!" [Online] Available:http://moveit.ros.org.

[10] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, 2006.

[11] I. Little, S. Thiebaux *et al.*, "Probabilistic planning vs. replanning," in *ICAPS Workshop on IPC: Past, Present and Future*, 2007.

[12] D. Martínez, G. Alenyà, and C. Torras, "Planning robot manipulation to clean planar surfaces," *Engineering Applications of Artificial Intelligence*, vol. 39, pp. 23–32, 2015.

[13] M. Kopicki, R. Detry, M. Adjigble, R. Stolkin, A. Leonardis, and J. L. Wyatt, "One-shot learning and generation of dexterous grasps for novel objects," *The International Journal of Robotics Research*, vol. 35, no. 8, pp. 959–976, 2016.