Master Thesis

Nicola Covallero

April 17, 2016

Contents

Chapter 1

Introduction

1.1 Problem Approach

In this section the approach to resolve the planning problem is described. For the table clearing task the main actions the robots has to use in order to interact with the objects are:

- Grasping
- Pushing

Grasping is the most important action since it lets to take an object from the pile of objects and drop it somewhere, for example in a bin, clearing in this way the table. There exist different works facing the same task by focusing only in grasping. The planning becomes useful considering the problem that two adjacent objects could not be grasped if they are so close such that the robot's gripper, when attempting to grasp an object is going to collide with the other object, making such an object ungraspable. From this consideration is necessary the pushing action, in order to separate adjacent objects which mutually exclude themselves to be grasped. And in order to face the problem in elegant way a planning system is used, instead of a heuristic approach for the action decision making stage.

Chapter 2

State of the Art Manipulation Planning

cite David's works

still to add reference to the ones in the pushing.odt file

Many manipulation planning approaches (see [24] for an overview) assume that the task can be treated as a geometric problem, with the goal to place the objects in their desired positions. Planning is essentially done with a mixture of symbolic and geometric states, this requires a set of symbolic predicates that correspond to geometric relationships. In particular to the best of my knowledge, all the planners for cluttered scene are based on a mixture of symbolic and geometric predicates, or based only on geometric ones.

Dogar and Srinivasa [14] proposed a framework for planning in clustter scenes using a library of actions inspired by human strategies. They designed a new planner that decides which objects to move, the order to move them, where to move them, and it chooses the manipulation actions to use on these objects, and accounts for the uncertainty in the environment all through this process. The planner first attempts to grasp the goal object, than if it is not possible it identifies what is the object that obstacles the action, then such an object is added to a list of the objects that have to be moved. They moved an object in whatever position such that makes the goal feasible. Despite this, such a planning strategy can be used only in the design of a new planner, which is something we want to avoid.

To grasp they used the Push-grasping action [13] , which is a robust way of grasping objects under uncertainty. It is a straight motion of the hand parallel to the pushing surface along a certain direction, followed by closing the fingers. For the pushing directions they used a resolution of $\pi/18$ rad (i.e. 36 different directions) and use a predefined set of 9 hand aperture values. However their cluttered scenes is intended to be a scene with separated and well known objects.

In [17] the authors proposed to apply a linear temporal logic (LTL) planner to a manipulation planning task in cluttered scenes, but it suffers from poor runtime and the LTL specification is complex. Moreover to the scope of this work the temporal specification is not needed since it is indirectly encoded in the clutter scene composition.

An interesting planner which mixes symbolic and geometric predicates is aSyMov [8]. It is probably the most well known planner which a task to move objects considering how their location will then affect the possible robot's paths. To do that they combine a symbolic planner with a probabilistic roadmap [35] for geometric planning.

A recent alternative proposed by Mösenlechner and Beetz [30] 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 could be look promising to the scope of this thesis but still implies the design of a new planner.

In [26], Lonzano-Peréz et al. They use a simple task-level planner, in which operators are described with two types of preconditions: symbolic and geometric. They proposed a strategy for integrated task and motion planning based on performing a symbolic search for a sequence of high-level operations, such as pick, move and place, while postponing geometric decisions, based on the CSP (Constraint Satisfaction Problem) technique. **Their technique allow working directly with state of the art planners.**

In [9] the authors address a problem similar to the one of this thesis. Pushing presents itself as a more intricate problem than grasping, a for grasping, after the object has been grabbed, it becomes a matter of solving a geometric problem. However, pushing an object carries uncertainty to the resulting state of the object. It is, nonetheless, important to be able to incorporate these two actions in the planning. For pushing they used a similar strategy to [29]. The planning algorithm also incorporates grasping, which, together with push actions, can perform a vaster array of tasks. It uses the concept of **reachability** [36] for each kind of action to quickly exclude impossible poses of the gripper of the planning stage, creating a reliable plan suitable for real-time operation. The authors model the planning problem through MDP (Markov Decision Process), discretezing the world in grid cells and assigning to each one a push and grasp vector definied by the reachability concept. Moreover each object is also classified through a simple primitive shape in order to have a proper map of pushing and grasping for each kind of considered object.

The majority of the state of the art for task planning in cluttered scene is focused on designing a new planner, why in this thesis state of the art planners ready to use want to be used.

Learning in Task Planning Machine learning techniques also have been applied to task planning. To plan complex manipulation tasks, robots need to reason on a high level. Symbolic planning, however, requires knowledge about the preconditions and effects of the individual actions. In [7] the authors proposed a practical approach to learn manipulation skills, including preconditions and effects, based on teacher demonstrations. It is difficult to solve most real world manipulation tasks by reasoning purely in terms of low-level motions due to the high-dimensionality of the problems. Instead, robots should reason on a symbolic level and appropriately chain the learned actions to solve new tasks. Such a planning step, however, requires knowledge of the important preconditions and effects of the actions. With few demonstration the authors proposed a method to teach the robot the precondition and effects of individual actions.

Their method furthermore enables the robot to combine the learned actions by means of planning to solve new tasks that are more complicated than the learned individual actions. They focused their work on a class of manipulation actions where the preconditions and goals depend on spatial or geometrical constraints between the involved objects. They describe a scene as a collection of features and during the action demonstration the algorithm looks for the features that have a small change to identify the ones that can be predicates or effects of the considered action. They use an heuristic metric to define the variation of the features. The problem of this approach is that it moves the problem of identifying the right predicates to describe the problem in identifying right features that let a correct learning of the predicates. **We could consider this as further work**.

In [12] [11] the authors propose an approach for planning robotic manipulation tasks which uses a learned mapping between geometric states and logical predicates. The planner first plans symbolically, then applies the mapping to generate geometric positions that are then sent to a path planner. They try to fit a probability distribution function to the geometric states to map them to a symbolic state. This work sounds promising but the cluttered scene this thesis is going to treat is quite complex to be treated with such a method.

Interpreting the scene Focusing on symbolic planning, the research group of Artificial Intelligence and Robotics Laboratory of Istanbul Technical University, published some interesting researches suitable to the aim of this thesis. In [15] [31] [16] the authors propose a system which combines 3D recognition and segmentation results to create and maintain a consistent world model involving attributes of the objects and spatial relations among them. Unknown objects are modelled by using the segmentation output to determine their sizes and considering similarities with existing models to determine their shapes and colors. Then, these models are also stored as templates to be used for recognition along with the extracted attributes. They focused on the extraction of size, shape and color attributes as well as the following unary and binary spatial relations: on, on ground/on table, clear and near for object manipulation scenarios. These predicates are generated and updated over time to maintain a consistent world model. **For the aim of this thesis this paper is of particular interest because let to compute nicely some useful symbolic predicates** The *on* relation for a pair of objects is determined by checking whether their projections on the table plane overlap. For each pair of objects the *near* relation is determined by comparing the distance between the centers of these objects in each dimension with the sum of the corresponding sizes in that dimension.

Check fro backtracking techniques in planning, way to evaluate a plan

2.1 Segmentation

[22] interact with pile of objects with the aim to complement the segmentation algorithm for table clearance tasks.

2.2 Review Pushing

Review of pushing techniques

[19] object singulation

[20] object interactive perception and the same author did a work similar [21]

[27]

2.3 Review Grasping

Review of grasping techniques

In [37] they presented a uniform deterministic sequence and sets of sample over 2-sphere and SO(3).

In [10] paper of grasping

ROS object_manipulator

2.4 Usefull things

Temporal filtering to reduce the noise of the kinect [31].

2.5 Usefull Concepts

Closed world assumption [33] can be defined as having complete knowledge about the world, that is, the numbers and the attributes of all objects are known apriori.

2.6 What we have

- A naive pushing approach able to understand which object blocks the pushing action of another one. For the pushing action we could try to create a classifier in order to understand what it is the most likely pushing direction using geometric features. In this case we can evaluate different directions and wwith the feasible ones we could choose the best one. To push we could use or ProMP or DMP. When we have more possible pushing directions choose the one the moves the nearer the center of the robot's workspace. **Can we learn the pushing action? With a learning algorithm, not a classifier. Maybe we could create a classifier based on learning procedure.**
- Understand when an object is on top of another one
- Grasping? we miss how to decide if an object is graspable and if it is possible and, when it is not, understand if it is fault of an adjacent object. Strategy to resolve that:
  - Haf algorithm (very expensive) first on the object considering adjacent objects and then only on the object, so we can compare and understand if that was feasible.

  - HAF and then checking collision with the environment modelling in a simpler way the gripper, and detecting what are the objects that obstacle it

    With haf we also should test different rotations, this is too much expensive
  - AGILE: (not too expensive, but still more or less 2-3 seconds for whatever point cloud) and it selects just randomly the grasping poses. It is not a good choice doing that except using a very large number of sample, in that case we also have to use a point cloud reconstruction to resolve the problems seen in my work.
  - Naive approach: considering grasping it in the principal direction, it is computationally very cheap, and detecting for collision.
  - For objects that are not sorrounded by any other objects we should use a more powerfull algorithm in order to assure a good grasping pose detection.

The most of state of the art dealing with problems similar to this one manage the grasping action as unfeasible for the part of object that obstacle the arm, not the gripper, since they are usually dealing with uncluttered scenarios, where the objects are all separated among themselves. The idea here is to relax the grasping checking in order to speed up the translation process into symbolic level. Then to perform real grasping a reliable algorithm is used in order to get a good grasping, and checked for collision. If it is not possible the grasping we can deduce that the robot is not able to find grasp it, and take off from the goal such object and all the objects. Considering the case that such an object is on top of another one also the bottom object cannot be included in the goal, so it should be not included into the planner's goal.

2.7 Planners

2.8 Materials

2.8.1 Collision Detection Libraries

There exists several collision detection libraries, usally done for videogames such as: OZCollide [23], Bullet (available also for ROS framework)[6], Flexible Collision Library [32]. OZCollide has a poor documentation, try bullet or FCL.

- Classical planners
- Hierachical Planners
- Probabilistic Planners

2.9 Problem Description

2.10 Segmentation

2.11 Predicates

2.11.1 Block Predicates

2.11.2 On Top Predicates

2.11.3 Block Grasp Predicates

2.12 Pushing Action

2.13 Grasping Action

2.14 Real Robot

Chapter 3

Task Planner

In this chapter the general framework adopted is discussed, proposing a suitable task planner. After the review of the current state of the art of task planners, a proper planner is chosen and then a suitable description to the table clearing problem is discussed.

3.1 State of the Art

As already seen in chapter 1 there exist different kind of planners and they can be grouped in three main categories:

1. classical planners
2. hierarchical planners
3. probabilistic planners

Classical planners, as suggested by the name, are the more classical and easy to use. They are characterized by environments which are fully observable, deterministic, finite and static (changes happen only when the agent acts) and discrete (in time, actions, objects..) [34]. A deterministic problem is generally formulated as a 6-tuple $\Pi = \langle S^d, s_o^d, G^d, A^d, T^d, c^d \rangle$ [25], where:

- S^d is a finite set of states;
- $s_o^d \in S^d$ is an initial state;
- $G^d \in S^d$ is a goal state;
- $A^d(s)$ is a set of applicable actions for each $s \in S^d$;
- $T^d(a, s) \in S^d$ is a deterministic transition function for all actions $a \in A^d(s)$ and states $s \in S^d$;
- $c^d(a)$ is the cost to apply action a.

The solutions, or trajectorys, τ_i are sequences of actions applicable from the initial state until the goal state. The cost of a trajectory $C(\tau_i)$ is the sum of the cost of the actions of the trajectory $\sum_{a \in \tau} c^d(a)$. The optimal solution is the solution with less cost: $\tau^* = \min_{\tau_i} c^d(\tau_i)$. A very well known classic planner is the Fast Downward planner [18].

Hierarchical planning, also called *Hierarchical Task Network*(HTN), works in a similar way to how it is believed that human planning works [28]. It is based on a reduction of the problem. The planner recursively decomposes tasks into subtasks, stopping when it reaches primitive tasks that can be performed directly by planning operators. In order to tell the planner how to decompose nonprimitive tasks into subtasks, it needs to have a set of methods, where each method is a schema for decomposing a particular kind of task into a set of subtasks [1]. For this kind of planning technique a well known planner is JSHOP2 [4]. **Probabilistic planning** is a planning technique which consider that the environment is not deterministic but probabilistic. So the actions have a certain probability to obtain a certain state, and given an initial and final state the planner finds the solution path with the highest probability. A well known example on which this kind of planners are build on is the Markov Decision Process. A probabilistic problem is generally formulated as a 6-tuple $\Pi = \langle S, s_o, G, O, T, A \rangle$ [25], where:

- S is a finite set of states;
- $s_o \in S$ is the initial state;
- $G \in S$ is a goal state;
- O is the set of outcomes, the probability of $o \in O$ is $Pr(o)$;
- $T(o, s) \in S$ is a (total) deterministic transition function for all outcomes $o \in O$ and states $s \in S$;
- $A(s)$ is a set of applicable actions for each $s \in S$, coupled to a function $out(a) \subseteq O$ mapping each action to a set of outcomes in such a way that
  - each outcome $o \in O$ belongs exactly to one action $act(o)$;
  - $\sum_{o \in out(a)} Pr(o) = 1$ for all a.

In this case the optimal solution is the one with the highest probability. In this category two famous probabilistic planners are Gourmand [2] and PROST [3].

3.2 Planner

The problem this thesis is facing could be resolved by several approaches by using planners from all the categories. We have already seen that such a problem involves geometric constraints and those cannot be considered directly by the planner using a ready to use state of the art planner, that would imply a designing of a new planner. Since the aim of this work is not to design a new planner but to resolve the table clearing task through already existing planners the problem has to be cast in a way to be manipulated by existing planners. This easy way involves working with symbolic predicates, symbolic predicates are predicates which can be true or false, and they will be introduced more in detail in the next sections.

The problem moreover involves a big amount of uncertainty due to the interaction of the robot with the environment. When the robot will interact with the pile of objects it is very hard to predict correctly the position of the object in the next frame, that is the next state, this is a crucial problem which should be considered. With a probabilistic planner, the planner will take into account what object has been moved and it will update the state obtaining a set of states, each one with a certain probability, and the returned plan, or solution, is the one with the highest probability. The probability also has to be modelled and it is highly depending on the form of the object, which is also hard to predict. An other way to face the problem is to replan at each frame, that is after each interaction of the robot with the pile of objects, or whenever the current state deviates from the expected one, generating a new trajectory from the current state to the goal. The plan's actions are considered deterministic, and the only useful action is actually the first one of the plan, after its execution the system replans again. Little et al. discussed in [25] the problem of when is more useful the probabilistic planning with respect a simple replanning system. They defined a the concept of *Probabilistic Interesting Problem* with the following definition:

> A probabilistic planning problem is considered to be *probabilistically interesting* if and only if it has all of the following structural properties:
>
> - there are multiple goal trajectories;
> - there is at least one pair of distinct goal trajectories, τ and τ', that share a common sequence of outcomes for the first $n-1$ outcomes, and where τ_n and τ'_n are distinct outcomes of the same action; and
> - there exist two distinct goal trajectories τ and τ' and outcomes $o \in \tau$ and $o' \in \tau'$ of two distinct actions $a = act(o)$ and $a' = act(o')$ such that executing a strictly decreases the maximum probability of reaching a state where a can be executed.

They assert that unless a probabilistic planning problem satisfies all of the structural conditions in this definition, then it is inevitable that a well-written replanner will outperform a well-written probabilistic planner. Moreover the authors do not negate the possibility that a deterministic replanner could perform optimally even for probabilistically interesting planning problems.

To conclude, a replanner would make more probabilistic problem practically solvable. In the other hand it suffers to be less robust than a probabilistic one.

Taking into account such considerations, in order to face simply such a difficult problem, the problem has been thought to be solved thanks a deterministic replanner although it is *probabilistic interesting*. The choice also was guided by the difficulty to model the probability distribution of the actions which depends on the particular shape of the objects the robot has to interact with.

Between a classic planner and an heuristic one there is not much difference for the aim of this work. The planner chosen to develop the work is the **Fast Downward** planner [18][5], a very well know classic one. The advantage of this planner is its wide documentation with respect other planners, which makes it easier to use and it also has a wide community.

3.3 PDDL

Bibliography

[1] Description of the shop project https://www.cs.umd.edu/projects/shop/description.html .

[2] Gourmand Home Page.

[3] PROST Home Page.

[4] SHOP Home Page.

[5] Fast Downward Home Page.

[6] Bullet `http://wiki.ros.org/bullet`, 2015.

[7] Nichola Abdo, Henrik Kretzschmar, Luciano Spinello, and Cyrill Stachniss. Learning manipulation actions from a few demonstrations. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1268–1275. IEEE, 2013.

[8] Stéphane Cambon, Fabien Gravot, and Rachid Alami. A robot task planner that merges symbolic and geometric reasoning. In Ramon López de Mántaras and Lorenza Saitta, editors, *ECAI*, pages 895–899. IOS Press, 2004.

[9] Rui Coelho and Alexandre Bernardino. Planning push and grasp actions: Experiments on the icub robot.

[10] Hao Dang and Peter K. Allen. Semantic grasping: Planning robotic grasps functionally suitable for an object manipulation task. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)*, pages 1311–1317. IEEE, October 2012.

[11] Richard Dearden and Chris Burbridge. An approach for efficient planning of robotic manipulation tasks. In *ICAPS*. Citeseer, 2013.

[12] Richard Dearden and Chris Burbridge. Manipulation planning using learned symbolic state abstractions. *Robotics and Autonomous Systems*, 62(3):355 – 365, 2014. Advances in Autonomous Robotics — Selected extended papers of the joint 2012 {TAROS} Conference and the {FIRA} RoboWorld Congress, Bristol, {UK}.

[13] Mehmet Dogar and Siddhartha Srinivasa. Push-grasping with dexterous hands: Mechanics and a method. In *Proceedings of 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, October 2010.

[14] Mehmet Dogar and Siddhartha Srinivasa. A framework for push-grasping in clutter. In Nick Roy Hugh Durrant-Whyte and Pieter Abbeel, editors, *Robotics: Science and Systems VII*. MIT Press, July 2011.

[15] Mustafa Ersen, Melodi Deniz Ozturk, Mehmet Biberci, Sanem Sariel, and Hulya Yalcin. Scene interpretation for lifelong robot learning. In *Proceedings of the 9th international workshop on cognitive robotics (CogRob 2014) held in conjunction with ECAI-2014*, 2014.

[16] Mustafa Ersen, Sanem Sariel Talay, and Hulya Yalcin. Extracting spatial relations among objects for failure detection. In *KIK@ KI*, pages 13–20, 2013.

[17] Keliang He, Morteza Lahijanian, Lydia E. Kavraki, and Moshe Y. Vardi. Towards manipulation planning with temporal logic specifications. In *2015 IEEE Intl. Conf. Robotics and Automation (ICRA)*, pages 346–352, Seattle, WA, May 2015. IEEE.

[18] Malte Helmert. The fast downward planning system. *J. Artif. Intell. Res.(JAIR)*, 26:191–246, 2006.

[19] Tucker Hermans, James M. Rehg, and Aaron F. Bobick. Guided pushing for object singulation. In *IROS*, pages 4783–4790. IEEE, 2012.

[20] Dov Katz. *Interactive perception of articulated objects for autonomous manipulation*. University of Massachusetts Amherst, 2011.

[21] Dov Katz, Moslem Kazemi, J. Andrew (Drew) Bagnell, and Anthony (Tony) Stentz . Clearing a pile of unknown objects using interactive perception. In *Proceedings of IEEE International Conference on Robotics and Automation*, March 2013.

[22] Dov Katz, Arun Venkatraman, Moslem Kazemi, J Andrew Bagnell, and Anthony Stentz. Perceiving, learning, and exploiting object affordances for autonomous pile manipulation. *Autonomous Robots*, 37(4):369–382, 2014.

[23] Igor Kravtchenko. OZCollide `http://www.tsarevitch.org/ozcollide/`, 2015.

[24] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, May 2006.

[25] Iain Little, Sylvie Thiebaux, et al. Probabilistic planning vs. replanning. In *ICAPS Workshop on IPC: Past, Present and Future*, 2007.

[26] Tomás Lozano-Pérez and Leslie Pack Kaelbling. A constraint-based method for solving sequential manipulation planning problems. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 3684–3691. IEEE, 2014.

[27] Kevin M Lynch and Matthew T Mason. Stable pushing: Mechanics, controllability, and planning. *The International Journal of Robotics Research*, 15(6):533–556, 1996.

[28] Bhaskara Marthi, Stuart J Russell, and Jason Wolfe. Angelic semantics for high-level actions. In *ICAPS*, pages 232–239, 2007.

[29] Tekin A Mericli, Manuela Veloso, and H Levent Akin. Achievable push-manipulation for complex passive mobile objects using past experience. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 71–78. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[30] Lorenz Mösenlechner and Michael Beetz. Using physics- and sensor-based simulation for high-fidelity temporal projection of realistic robot behavior. In *AIPS*, 2009.

[31] Melodi Ozturk, Mustafa Ersen, Melis Kapotoglu, Cagatay Koc, Sanem Sariel-Talay, and Hulya Yalcin. Scene interpretation for self-aware cognitive robots. 2014.

[32] Jia Pan, Sachin Chitta, and Dinesh Manocha. Fcl: A general purpose library for collision and proximity queries. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3859–3866. IEEE, 2012.

[33] R. Reiter. A logic for default reasoning. In M. L. Ginsberg, editor, *Readings in Nonmonotonic Reasoning*, pages 68–93. Kaufmann, Los Altos, CA, 1987.

[34] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education, 2 edition, 2003.

[35] T. Siméon, J.-P. Laumond, and C. Nissoux. Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics*, 14(6):477–493, 2000.

[36] Nikolaus Vahrenkamp, Tamim Asfour, and Rudiger Dillmann. Robot placement based on reachability inversion. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1970–1975. IEEE, 2013.

[37] Anna Yershova and Steven M. LaValle. Deterministic sampling methods for spheres and SO(3). In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA 2004, April 26 - May 1, 2004, New Orleans, LA, USA*, pages 3974–3980, 2004.